OPENSIPS + ENVOY

# OpenSIPS + Envoy

## A Solution for TLS networking.

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

# 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

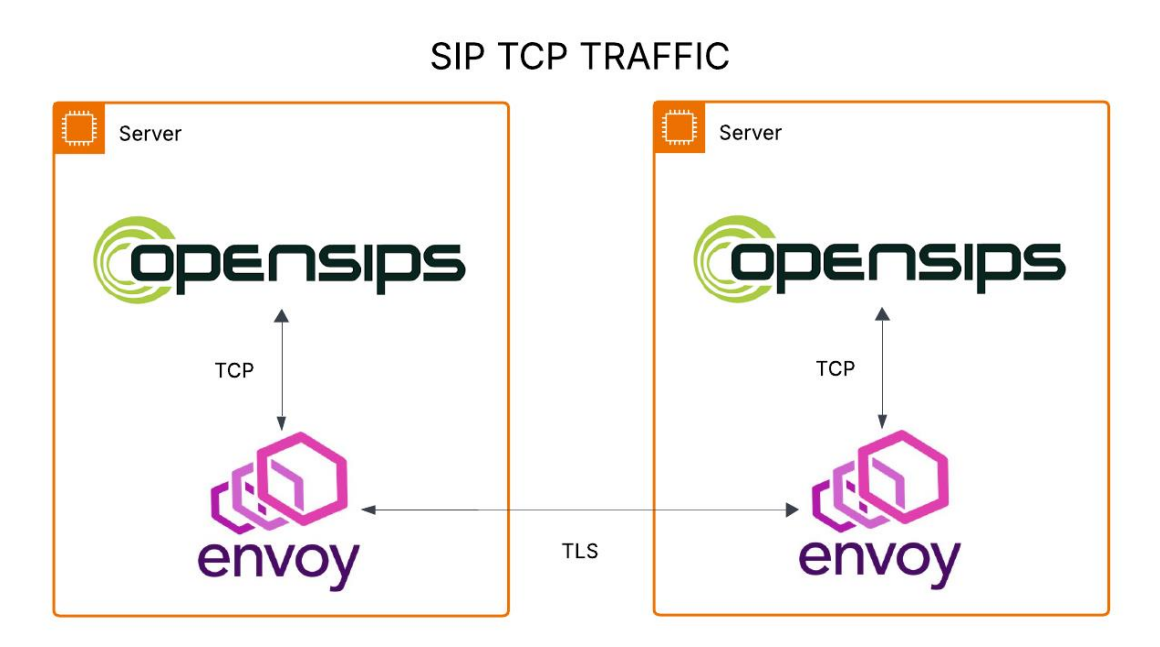
05 Demo

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

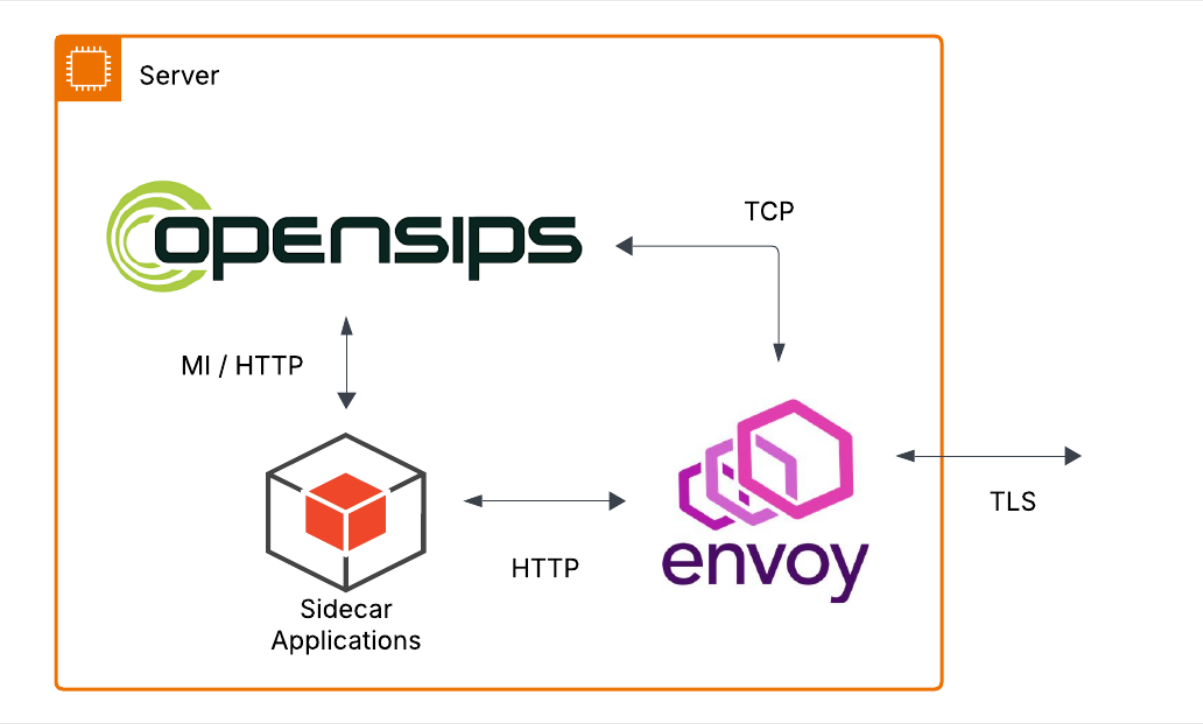
Egress

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

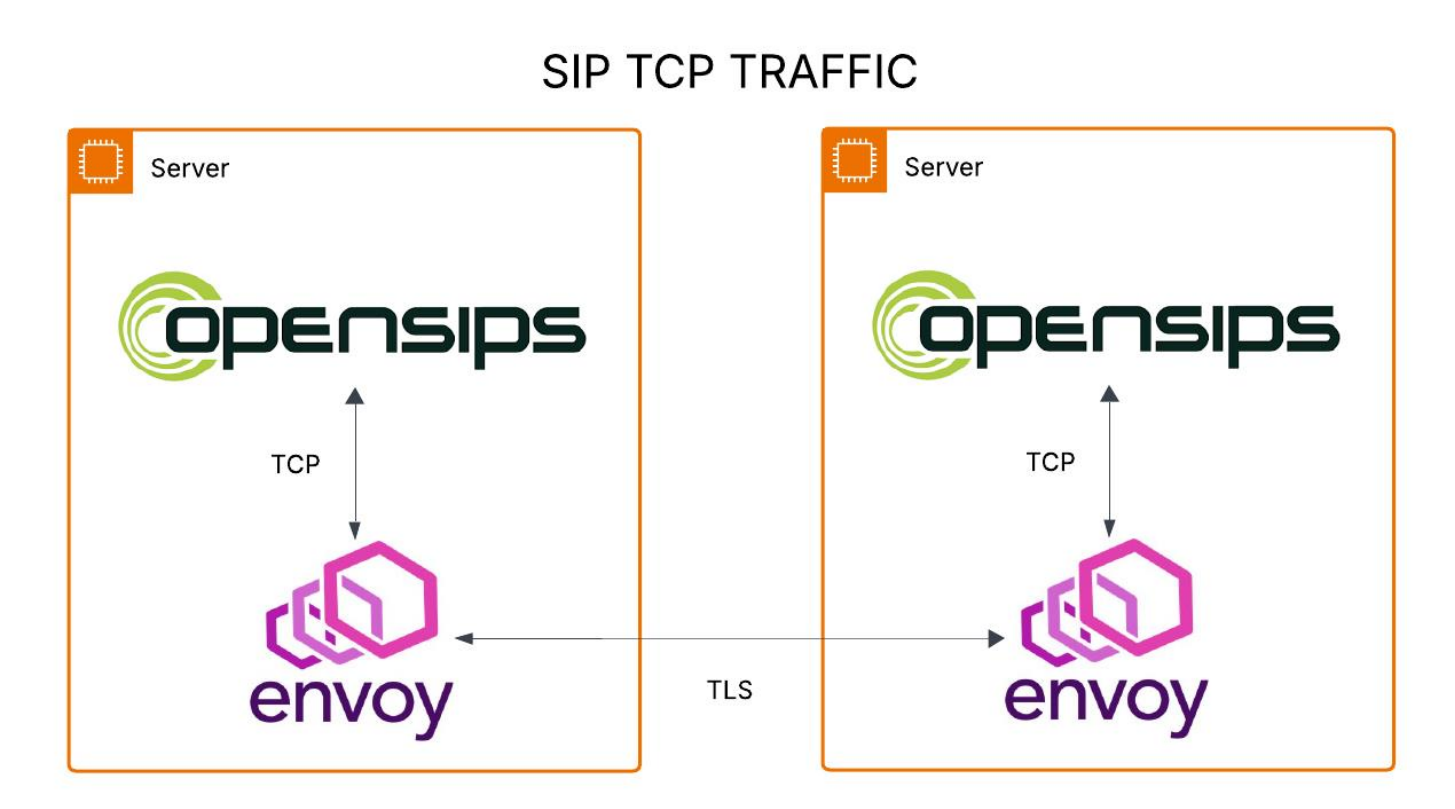
05 Demo

TLS PROXY

06 What's Next? Questions?

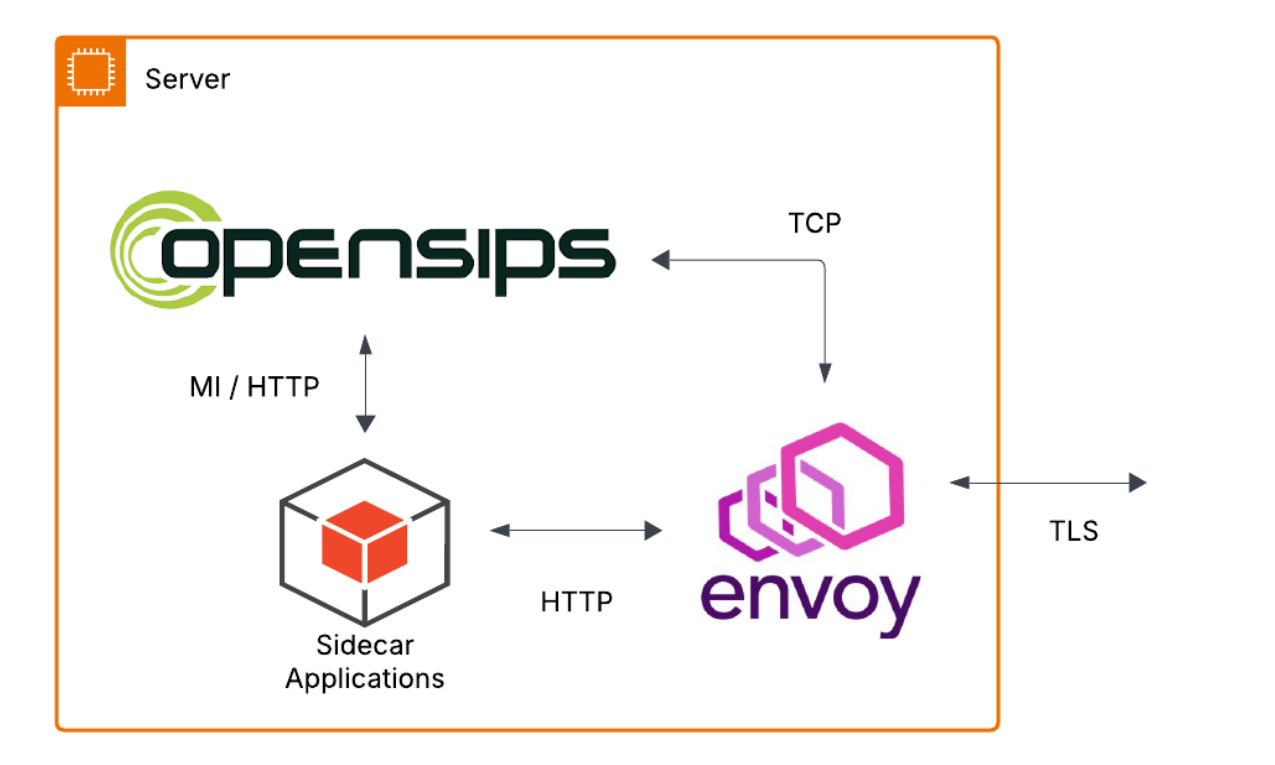
# Trusted Communication

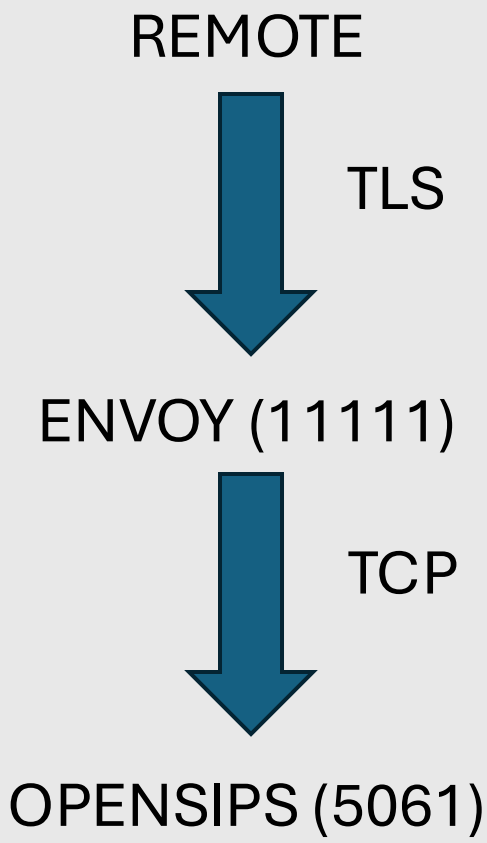
## Flow



# 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?

opensips

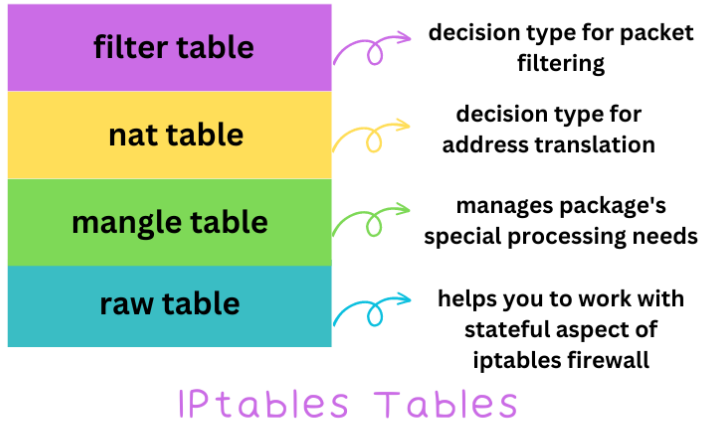
Port 5061

envoy

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**

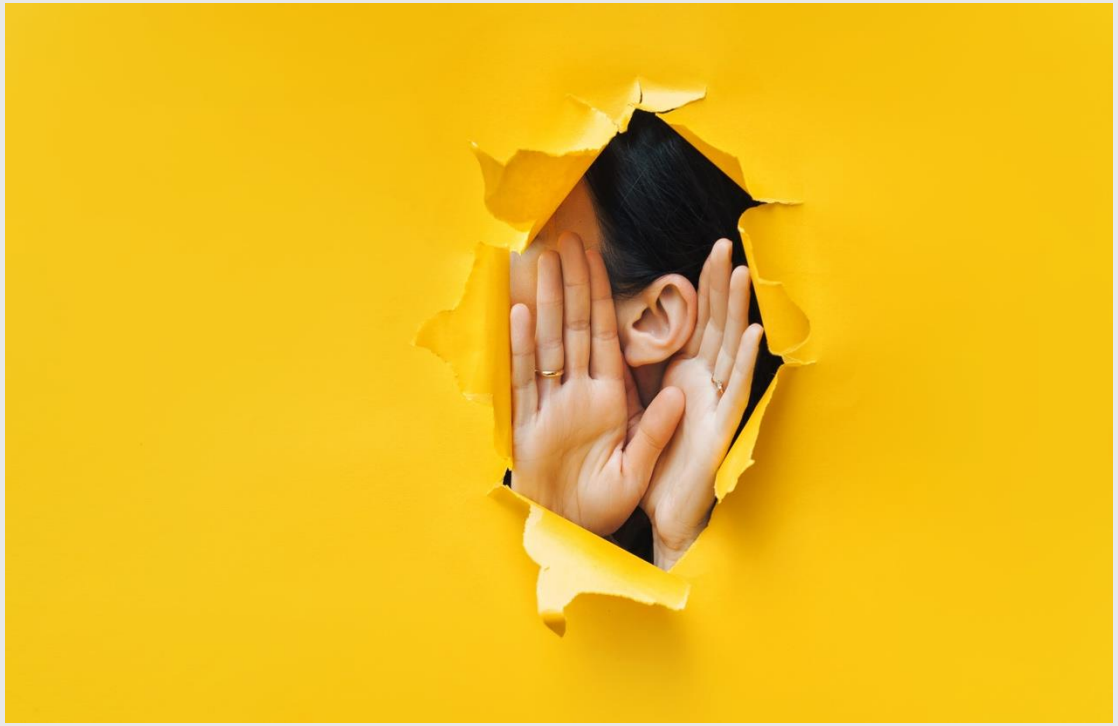
# Envoy Configuration - Ingress

```
static_resources:
  listeners:
  - name: ingress_listener
    address:
      socket_address:
        address: 0.0.0.0
        port_value: 11111
    listener_filters:
    - name: envoy.filters.listener.original_src
      typed_config:
        "@type": type.googleapis.com/envoy.extensions.filters.listener.original_src.v3.OriginalSrc
        mark: 123
    filter_chains:
    - filters:
      - name: envoy.filters.network.tcp_proxy
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.network.tcp_proxy.v3.TcpProxy
          stat_prefix: ingress_tcp
          cluster: ingress_cluster
      transport_socket:
        name: envoy.transport_sockets.tls
        typed_config:
          "@type": type.googleapis.com/
          envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
          common_tls_context:
            tls_certificates:
              - certificate_chain:
                  filename: "/etc/envoy/certs/server.crt"
                private_key:
                  filename: "/etc/envoy/certs/server.key"
  clusters:
  - name: ingress_cluster
    type: STATIC
    lb_policy: ROUND_ROBIN
    load_assignment:
      cluster_name: ingress_cluster
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: 127.0.0.1
                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**

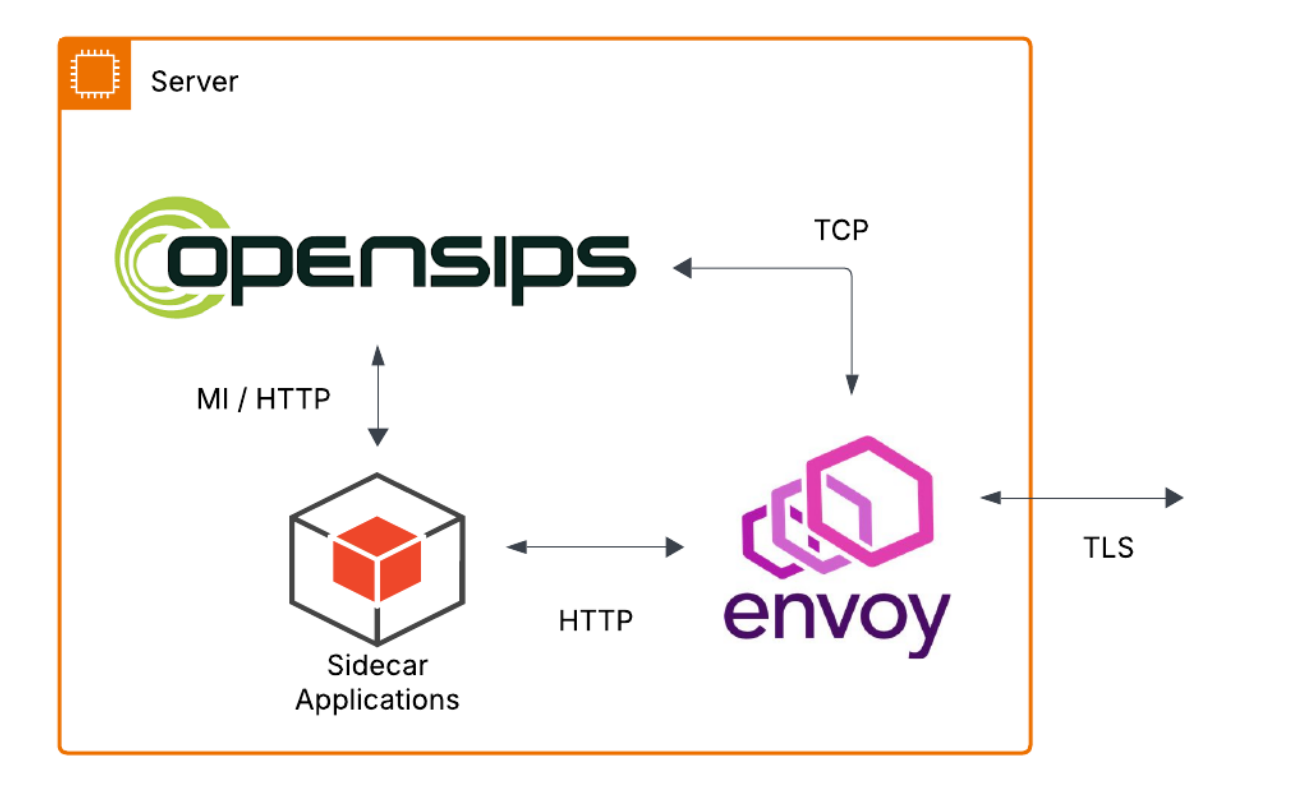
05 Demo

TLS PROXY

06 What's Next? Questions?

# Envoy Configuration

## Egress Flow




OPENSIPS + ENVOY

# OpenSIPS TLS via TCP

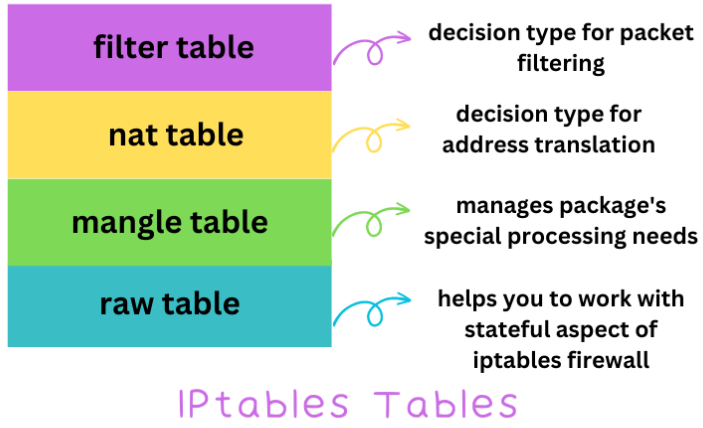
## Packet Marking

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

```
static_resources:
  listeners:
  - name: egress_listener
    address:
      socket_address:
        address: 0.0.0.0
        port_value: 22222
    listener_filters:
      - name: envoy.filters.listener.original_dst
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.listener.original_dst.v3.OriginalDst
    filter_chains:
    - filters:
      - name: envoy.filters.network.tcp_proxy
        typed_config:
          "@type": type.googleapis.com/envoy.extensions.filters.network.tcp_proxy.v3.TcpProxy
          stat_prefix: tcp_proxy
          cluster: egress_cluster
  clusters:
  - name: egress_cluster
    type: ORIGINAL_DST
    lb_policy: CLUSTER_PROVIDED
    transport_socket:
      name: envoy.transport_sockets.tls
      typed_config:
        "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.UpstreamTlsContext
        sni: "*"
        common_tls_context:
          validation_context:
            trusted_ca:
              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
```

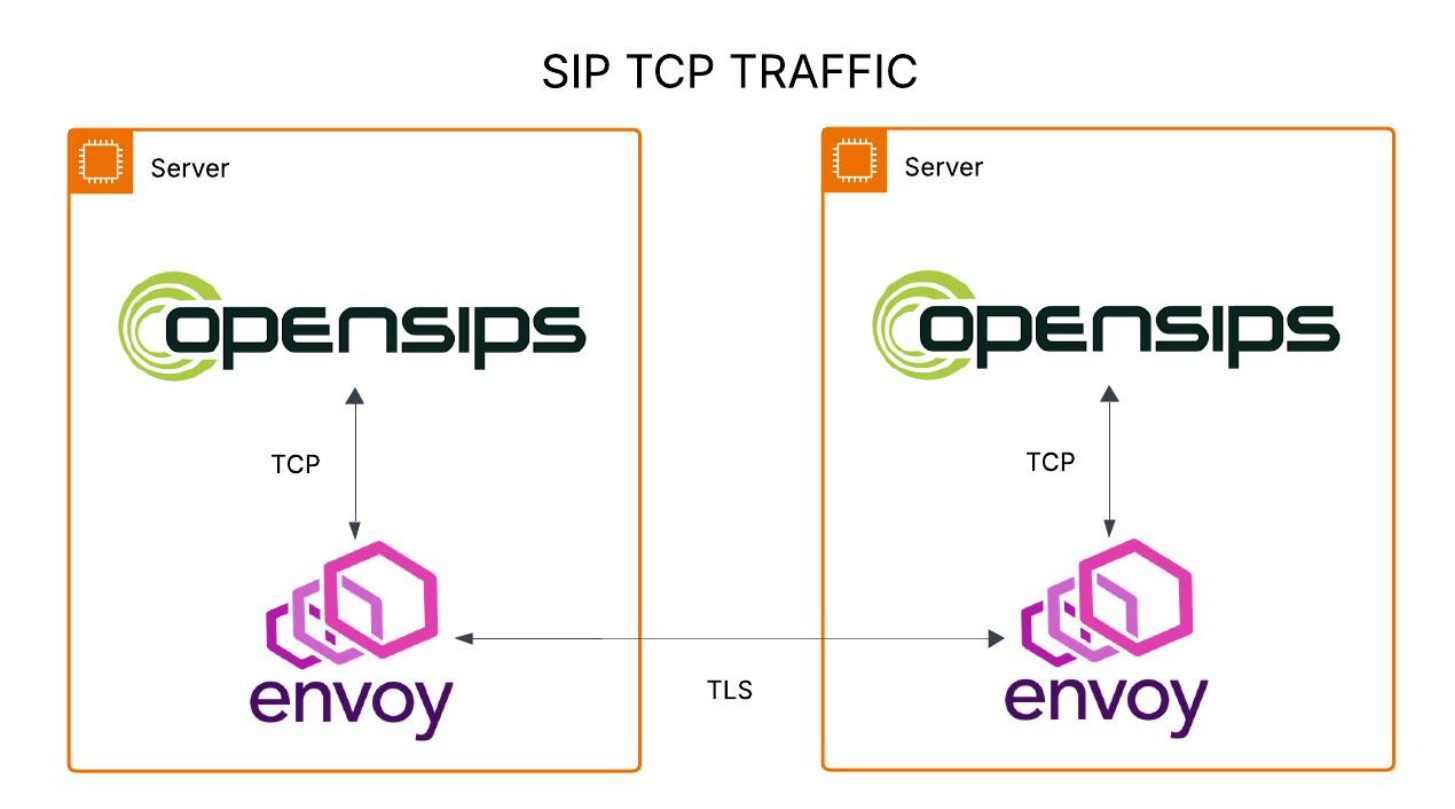
# Envoy Configuration

## 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

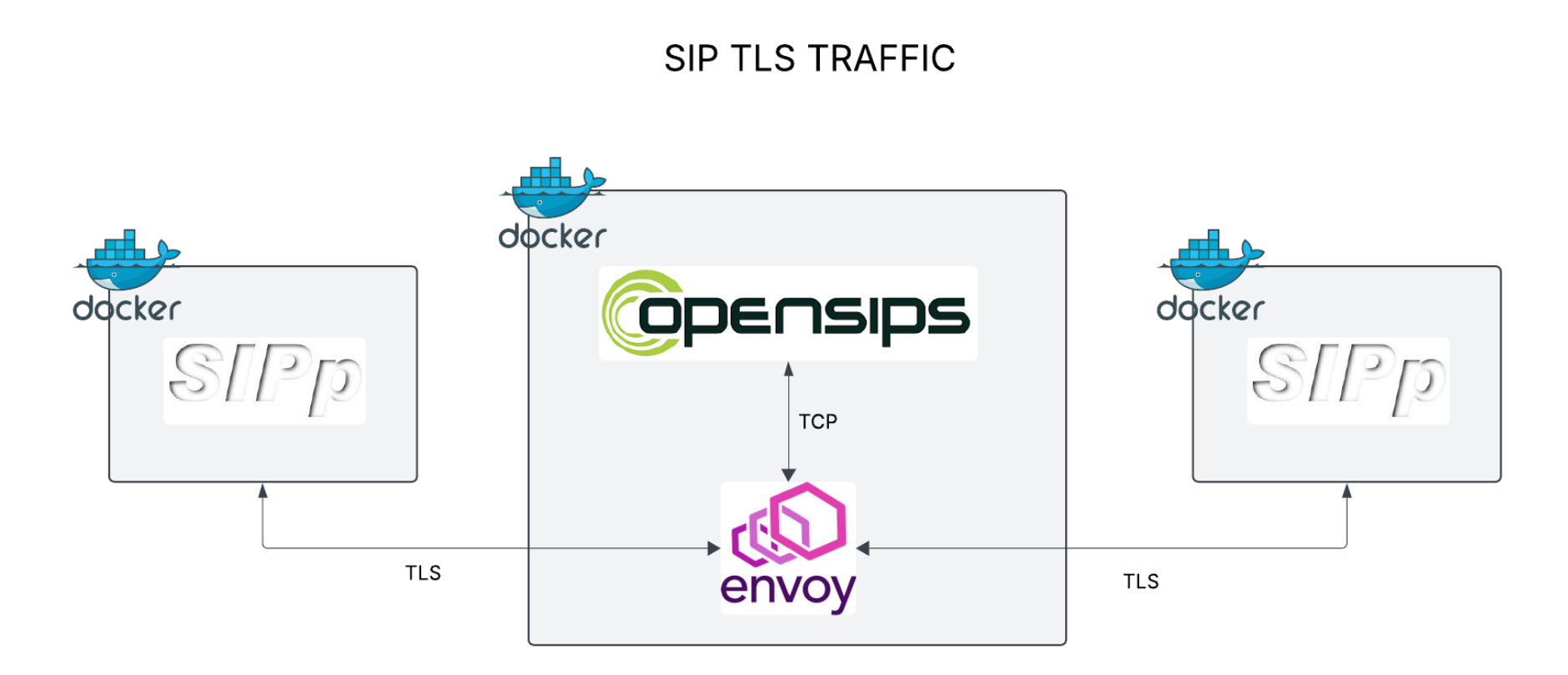
05 **Demo**

TLS PROXY

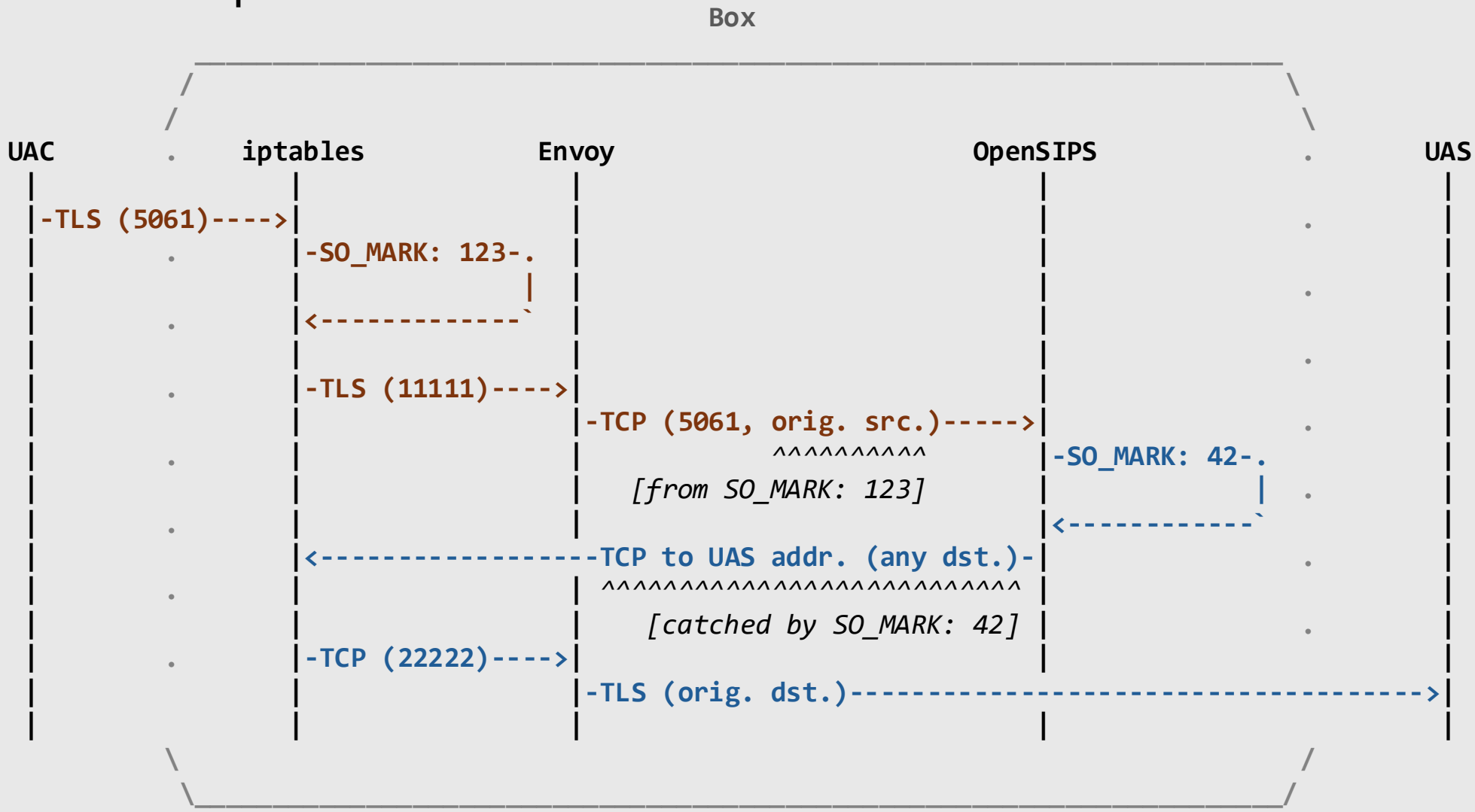
06 What's Next? Questions?

# A STEP FURTHER

## DEMO

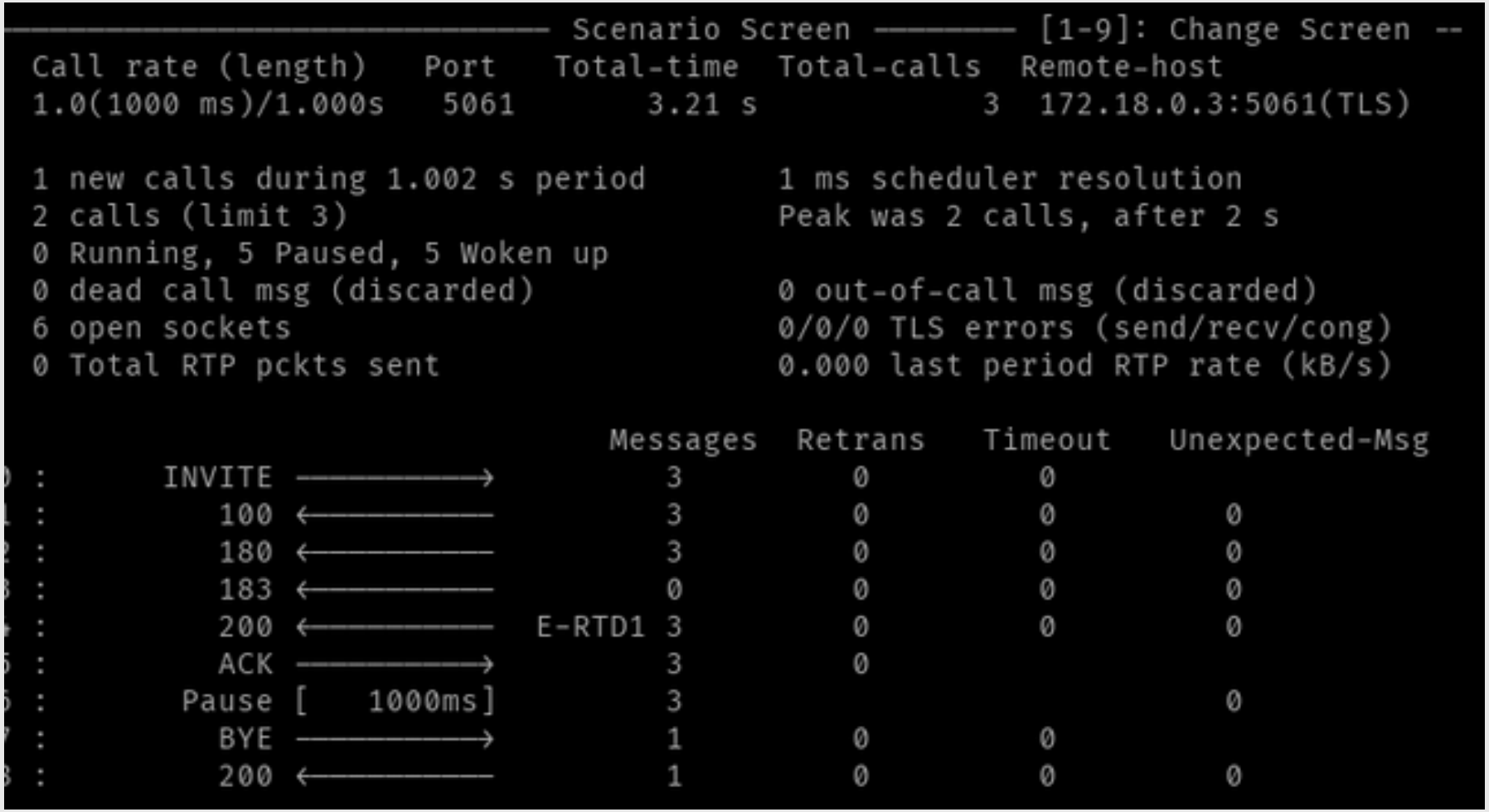


# Proof of Concept Demo




https://github.com/purecloudlabs/opensips-summit-2025

# Proof of Concept Demo: UAC Output



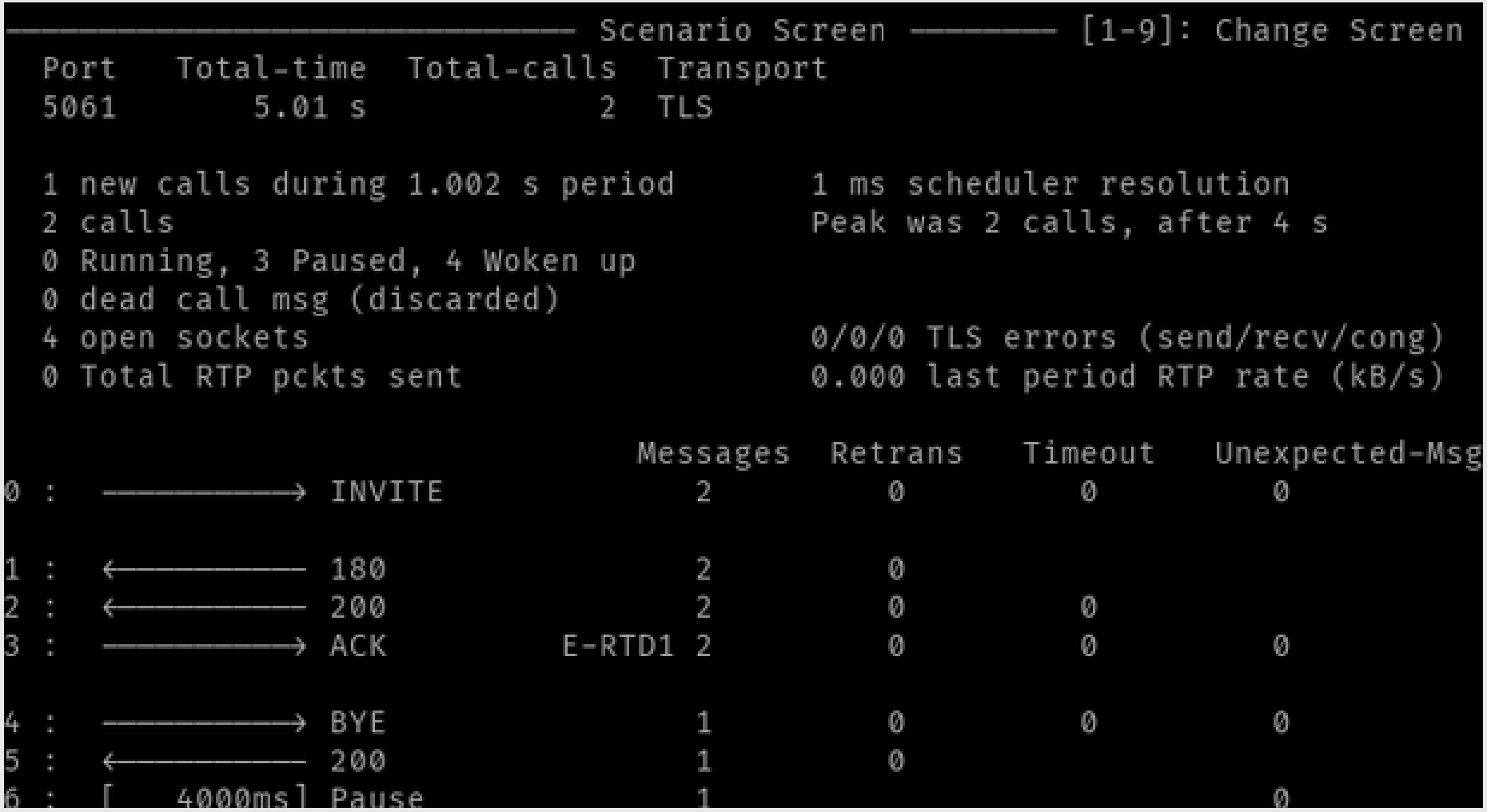
OPENSIPS + ENVOY

# Proof of Concept Demo: OpenSIPS + Envoy Output

```
Listening on
             tcp: 0.0.0.0 [0.0.0.0]:5061
Aliases:

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
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'.
```

```
for address 172.18.0.4:32921. Marking with 123

for address 172.18.0.4:39255. Marking with 123

for address 172.18.0.4:39675. Marking with 123
```

# Proof of Concept Demo: UAS Output



# 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.

# THANK YOU!