Introducing TLS Bumping for Integrating SASE functions with Service Mesh

Lei Zhang
Luyao Zhong

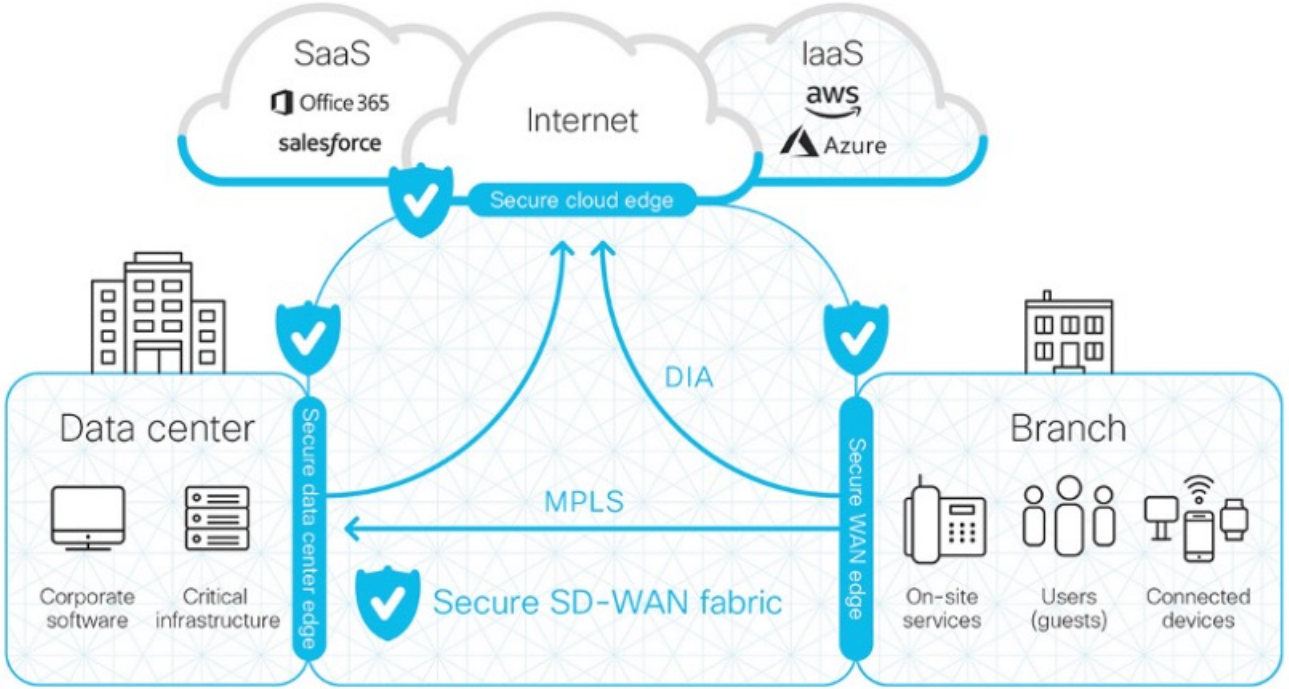
Agenda

- What is SASE
- Integration of service mesh and SASE
- TLS bumping
- Envoy* enhancements
- Future plan

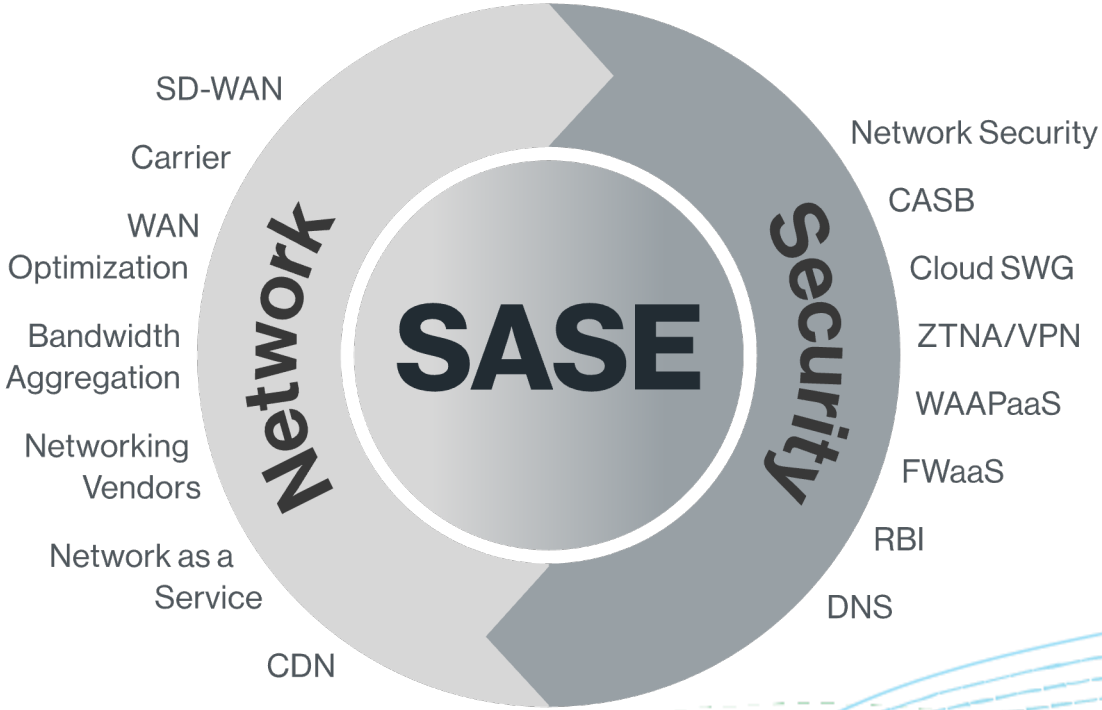
What is SASE

- Secure access service edge (SASE) is a technology used to deliver wide area network (WAN) and security controls as a cloud computing service directly to the source of connection (user, device, branch office, Internet of things (IoT) device) rather than a data center.

What is SASE



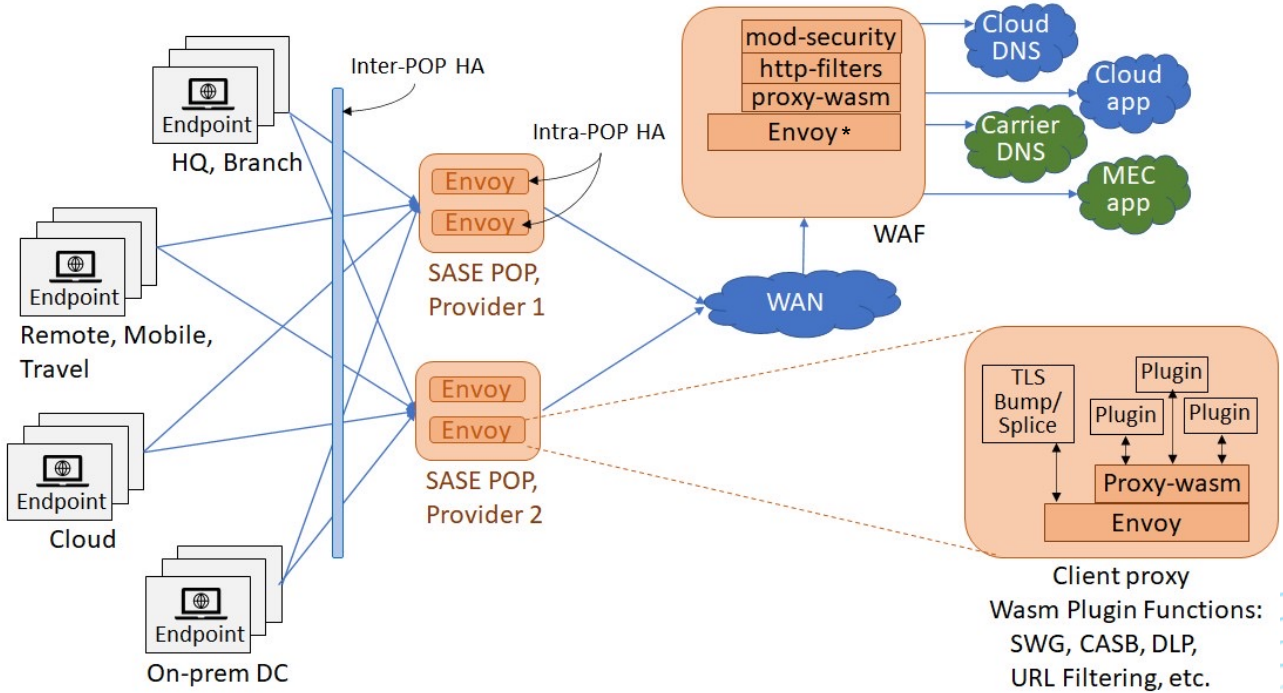
Key components



Integration of service mesh and SASE

- SASE security functions run in cloud edge
- Kubernetes* is popular in cloud edge
- Service mesh provides strong traffic management
- Can we combine them together?

Integration of service mesh and SASE



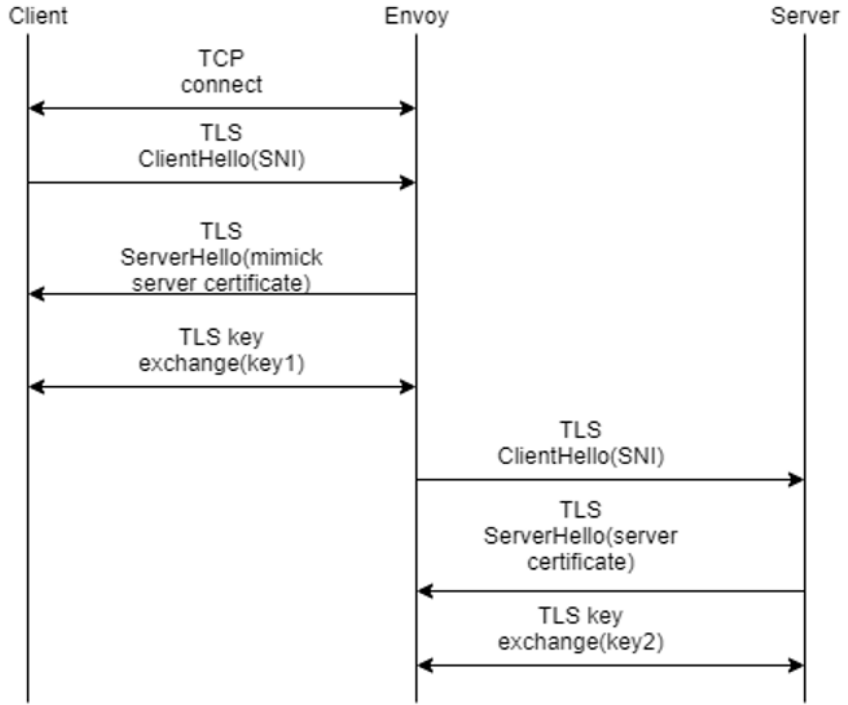
Prerequisites

- TLS bumping (Envoy* as a forward proxy to decrypt TLS traffic)
- Security functions (SWG, CASB, DLP, and ModSecurity) as WASM plugins
- CA key protection

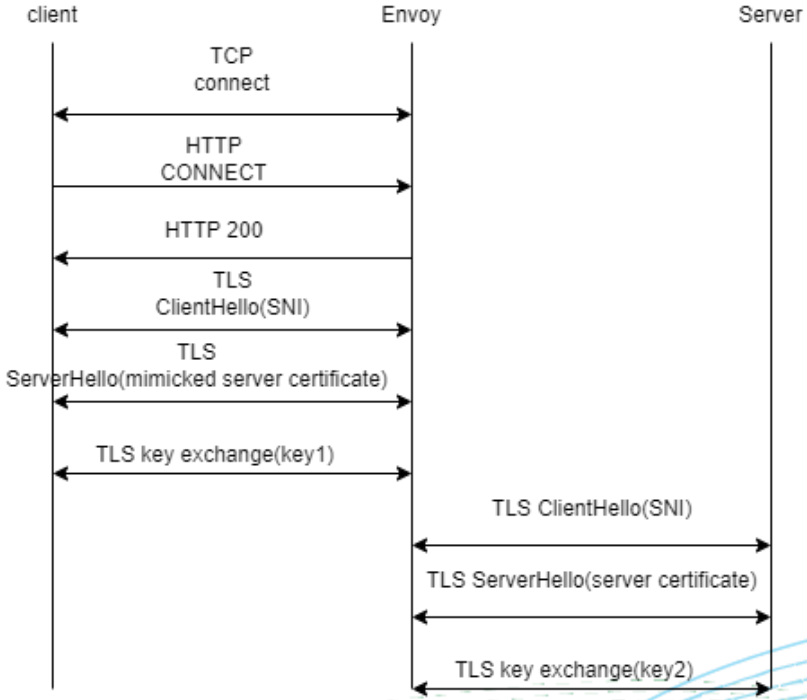
TLS bumping

- Scenario 1(sidecar)
  - The client is configured with no proxy, traffic is hijacked to Envoy*
- Scenario 2(proxy)
  - The client is configured with Envoy as the proxy

TLS bumping without CONNECT



TLS bumping with CONNECT

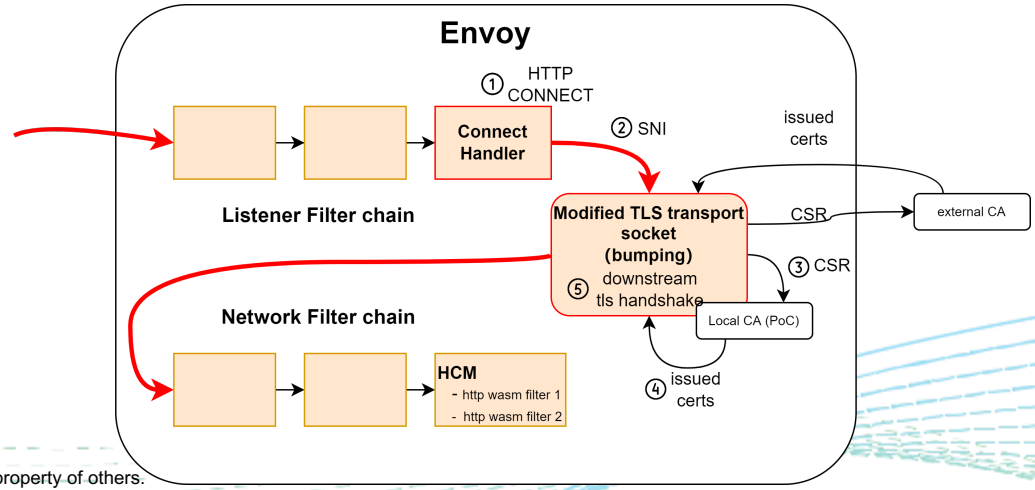


Gaps in Envoy

- CONNECT support is for tunneling
- DO NOT support mimicking cert
  - DO NOT support issuing cert at runtime
  - Only issues certs for internal service

Envoy* enhancements

- Current design
  - Listener filter to terminate HTTP1 CONNECT
  - Downstream-first secure connection
  - Mimic cert based on SNI
  - One-way mimicking



Configuration

- Set conn_handler listener filter

```
listener_filters:
- name: envoy.filters.listener.tls_inspector
- name: envoy.filters.listener.connect_handler
```

- Set CA cert to downstream TLS transport socket

```
transport_socket:
  name: envoy.transport_sockets.tls
  typed_config:
    "@type": type.googleapis.com/envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
    common_tls_context:
      tls_root_ca_certificate:
        cert: {"filename": "root-ca.pem"}
        private_key: {"filename": "root-ca.key"}
```


Demo

- TLS bumping without HTTP CONNECT
  - Envoy* as transparent proxy

test@node1:~/envoy$ curl –v https://www.baidu.com/

```
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: CN=www.baidu.com
*  start date: Mar 15 07:42:28 2022 GMT
*  expire date: Mar 14 07:42:28 2024 GMT
*  subjectAltName: host "www.baidu.com" matched cert's "www.baidu.com"
*  issuer: OU=MyRootCA R2; O=MyRootCA; CN=MyRootCA
*  SSL certificate verify ok.
```

- TLS bumping with HTTP CONNECT
  - specify Envoy as front proxy

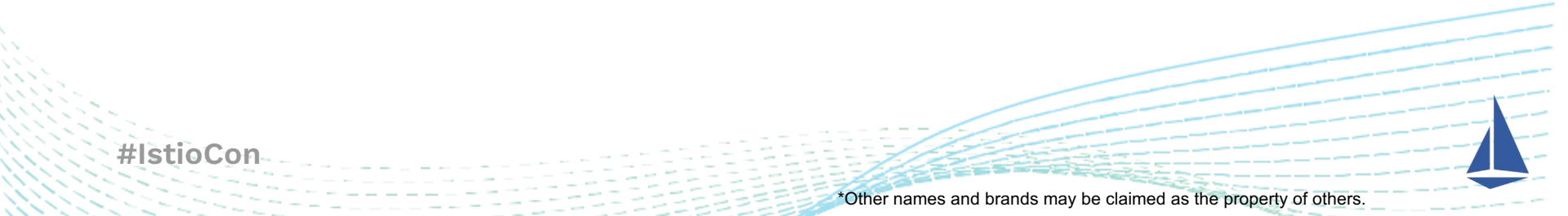
ubuntu@node1:~/envoy$ curl -v -x 127.0.0.1:1234 https://www.baidu.com/

```
< HTTP/1.1 200 Connection Established
<
* Proxy replied 200 to CONNECT request
* CONNECT phase completed!
```

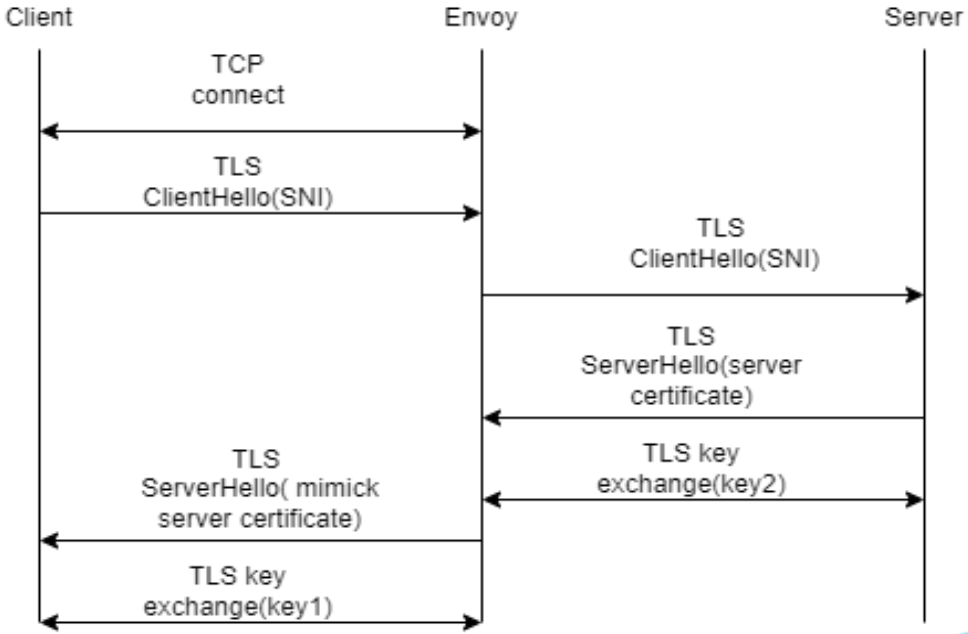
```
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: CN=www.baidu.com
*  start date: Mar 15 07:42:28 2022 GMT
*  expire date: Mar 14 07:42:28 2024 GMT
*  subjectAltName: host "www.baidu.com" matched cert's "www.baidu.com"
*  issuer: OU=MyRootCA R2; O=MyRootCA; CN=MyRootCA
*  SSL certificate verify ok.
```

Future plan

- Support terminating http2/3 CONNECT
- Upstream-first secure connection
- Mimic cert based on real server cert
- Mutual mimicking
- Istio* integration



Expected flow



Thank you!

lei.a.zhang@intel.com
luyao.Zhong@intel.com

#IstioCon

Notices and Disclaimers

Intel technologies may require enabled hardware, software or service activation.
No product or component can be absolutely secure.
Your costs and results may vary.