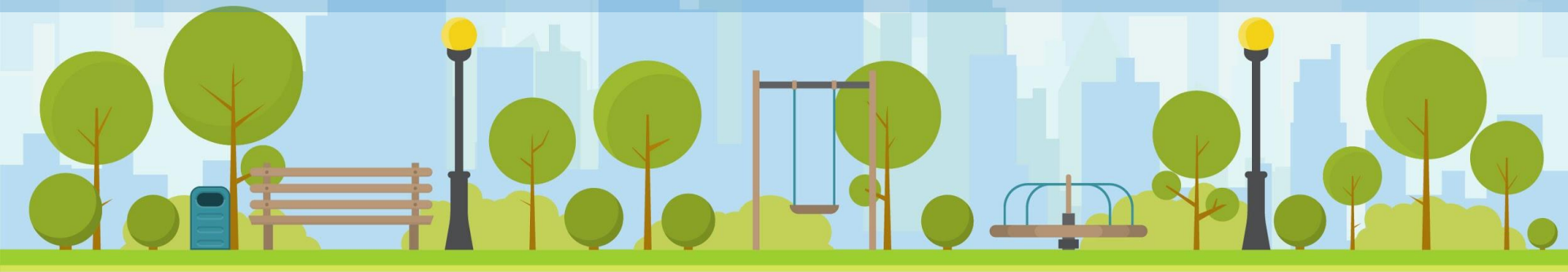

DEVOPS
PLAYGROUND
LONDON

DevOps Playground

Inspiring tech-enthusiasts to explore
new technology during hands-on monthly events.

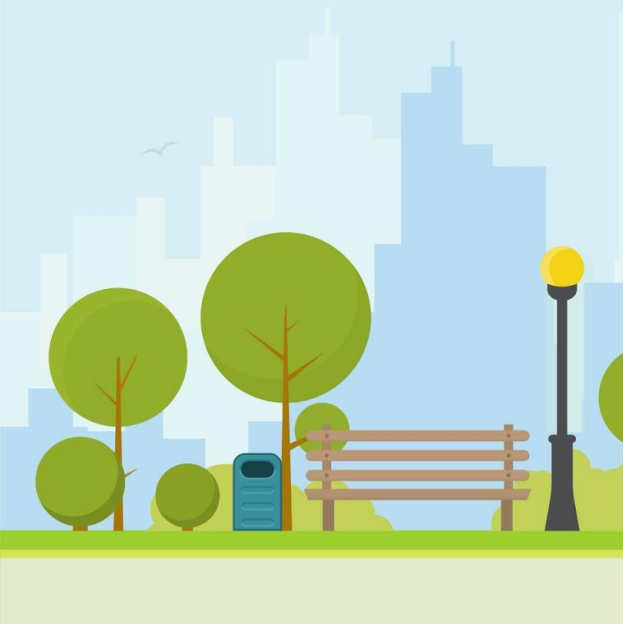


Hands on with Mutual TLS Authentication for Microservices

Roger Carhuatocto

@Chilcano
https://linkedin.com/in/chilcano
https://holisticsecurity.io

Synopsis



While Transport Layer Security (TLS) protects communications between two parties for confidentiality and integrity, the Public Key Certificates (x.509) and the Certificate Authority (CA) will allow users to set a cryptographic identity and build a trust relationship between these parties.

With all these concepts, we will be able to secure (encrypt) data in transit and authenticate the parties in scenarios such as Client-Server or Service-Service.

During this Playground you will learn:

- What One-Way, Two-Way or Mutual TLS are.
- How to use Public Key Certificates.
- How to use Mutual TLS together for securing communications among Microservices.
- How to automate everything.

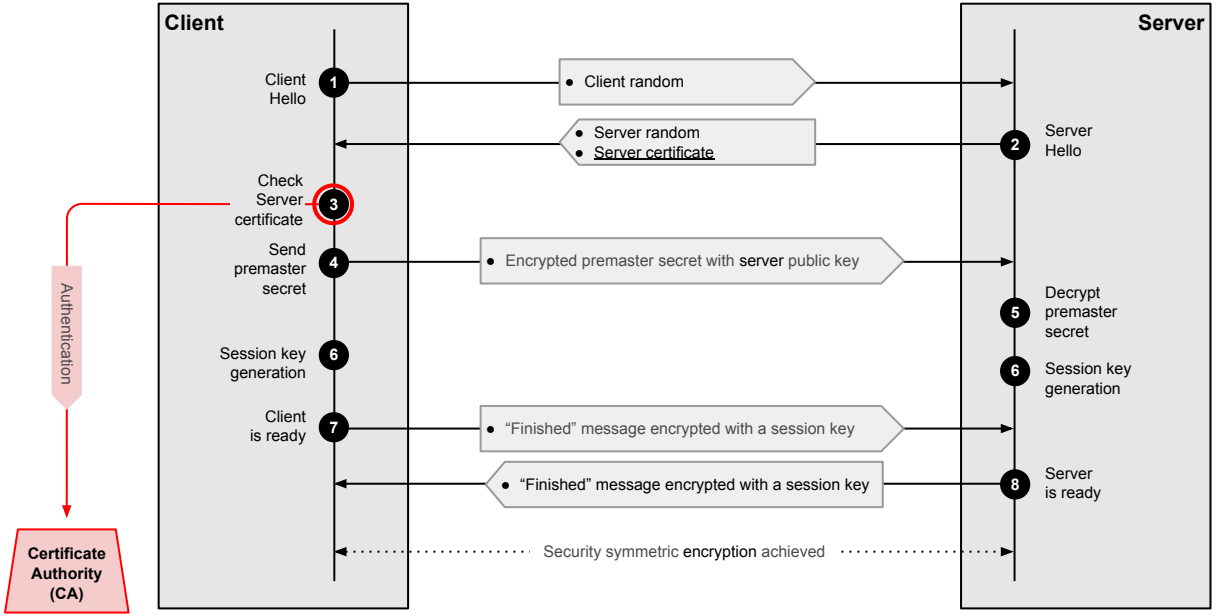
What is TLS?

"Transport Layer Security (TLS), and its now-deprecated predecessor, Secure Sockets Layer (SSL), are cryptographic protocols designed to provide communications security over a computer network. Several versions of the protocols are widely used in applications such as email, instant messaging, and voice over IP, but its use as the Security layer in HTTPS remains the most publicly visible". (https://en.wikipedia.org/wiki/Transport_Layer_Security)

TLS is able to work over these layers

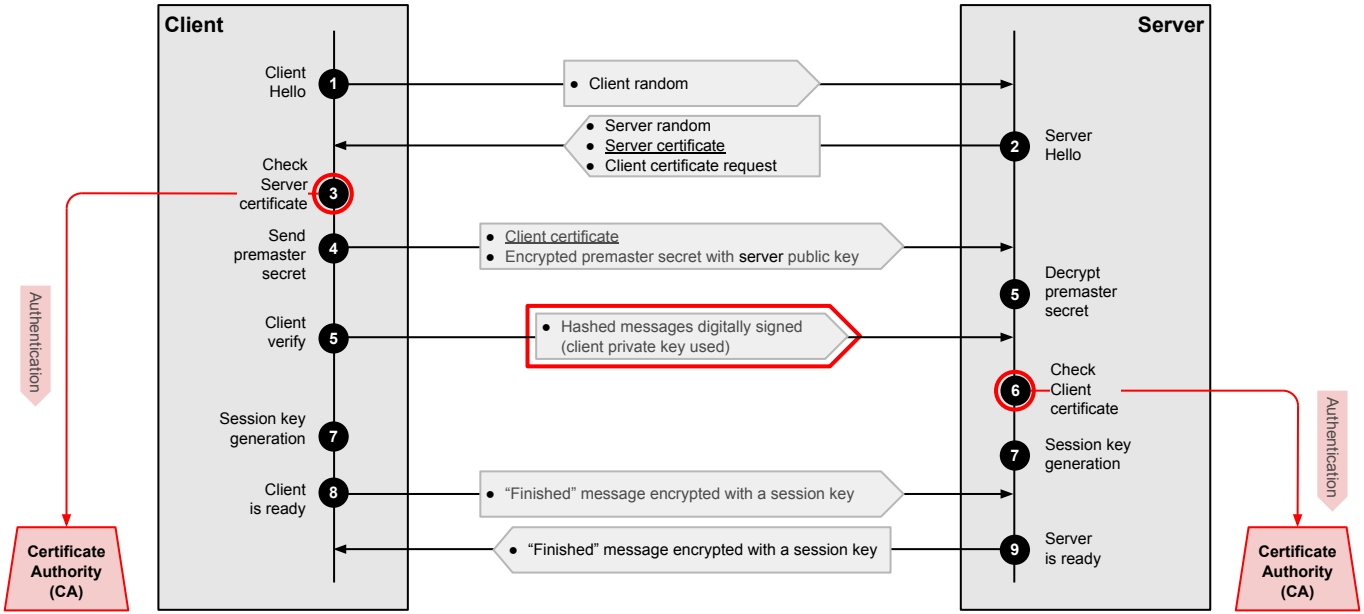
Layer	Function	Example of protocols and/or equipment
Application - 7	Services affecting end user applications	SMTP
Presentation - 6	Presentation Layer	JPEG - MIDI - MPEG - PICT - TIFF - GIF - HTTPS - SSL - TLS
Session - 5	Session Layer	NetBIOS - NFS - PAP - SCP - SQL - ZIP
Transport - 4	Transport Layer	TCP - UDP
Network - 3	Network Layer	Routers - Layer 3 Switches - IPsec - IPv4 - IPv6 - IPX - RIP
Data Link - 2	Data Link Layer	Switches - ARP - ATM - CDP - FDDI - Frame Relay - HDLC - MPLS - PPP - STP - Token Ring
Physical - 1	Physical Layer	Hubs - Bluetooth - Ethernet - DSL - ISDN - 802.11 - WiFi

One-way TLS (Server Authn)



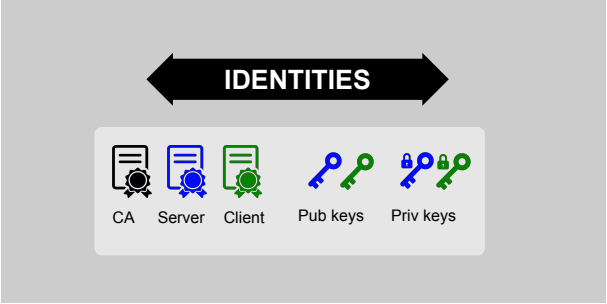
1. The client initiates the handshake by sending a "hello" message to the server. The message will include which TLS version the client supports, the cipher suites supported, and a string of random bytes known as the "client random."
2. In reply to the client hello message, the server sends a message containing the server's TLS certificate, the server's chosen cipher suite, and the "server random," another random string of bytes that's generated by the server.
3. The client verifies the server's TLS certificate with the certificate authority that issued it. This confirms that the server is who it says it is, and that the client is interacting with the actual owner of the domain.
4. The client sends one more random string of bytes, the "premaster secret." The premaster secret is encrypted with the public key and can only be decrypted with the private key by the server. (The client gets the public key from the server's TLS certificate.)
5. The server decrypts the premaster secret using its asymmetric private key.
6. Both client and server generate session keys from the client random, the server random, and the premaster secret. They should arrive at the same results.
7. The client sends a "finished" message that is encrypted with a session key.
8. The server sends a "finished" message encrypted with a session key.

Two-way TLS (Client + Server Authn)



What is the Security approach?

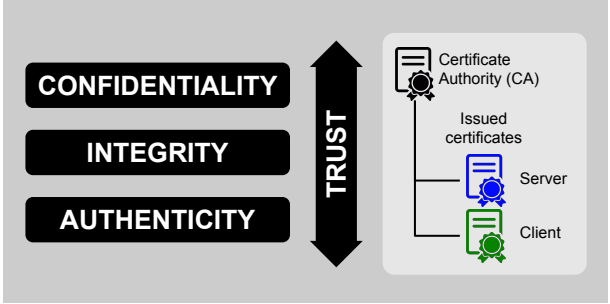
1 Identity-based Security



- Based on asymmetric-key cryptography.
- Public Key Certificate (X.509) and Private Key.

"Identity-based security is a type of security that focuses on access to digital information or services based on the authenticated identity of an individual".
https://en.wikipedia.org/wiki/Identity-based_security

2 Trust-based Security



- Public Key Infrastructure (PKI) provides "trust services" (CA, RA, OCSP, ..) and manages the X.509 Certificates lifecycle.

"In cryptography, a trusted third party (TTP) is an entity which facilitates interactions between two parties who both trust the third party".
https://en.wikipedia.org/wiki/Trusted_third_party

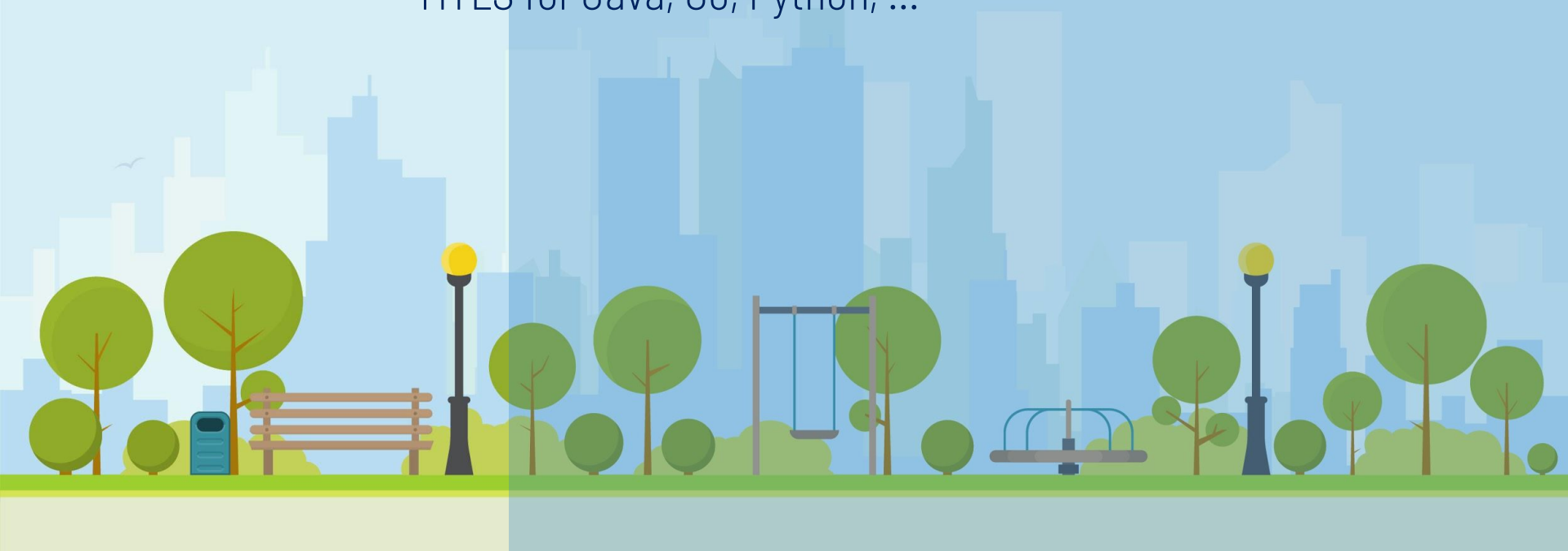
MINIMUM VIABLE SECURITY

MVP: minimum amount of effort invested in order to prove the viability of an idea.

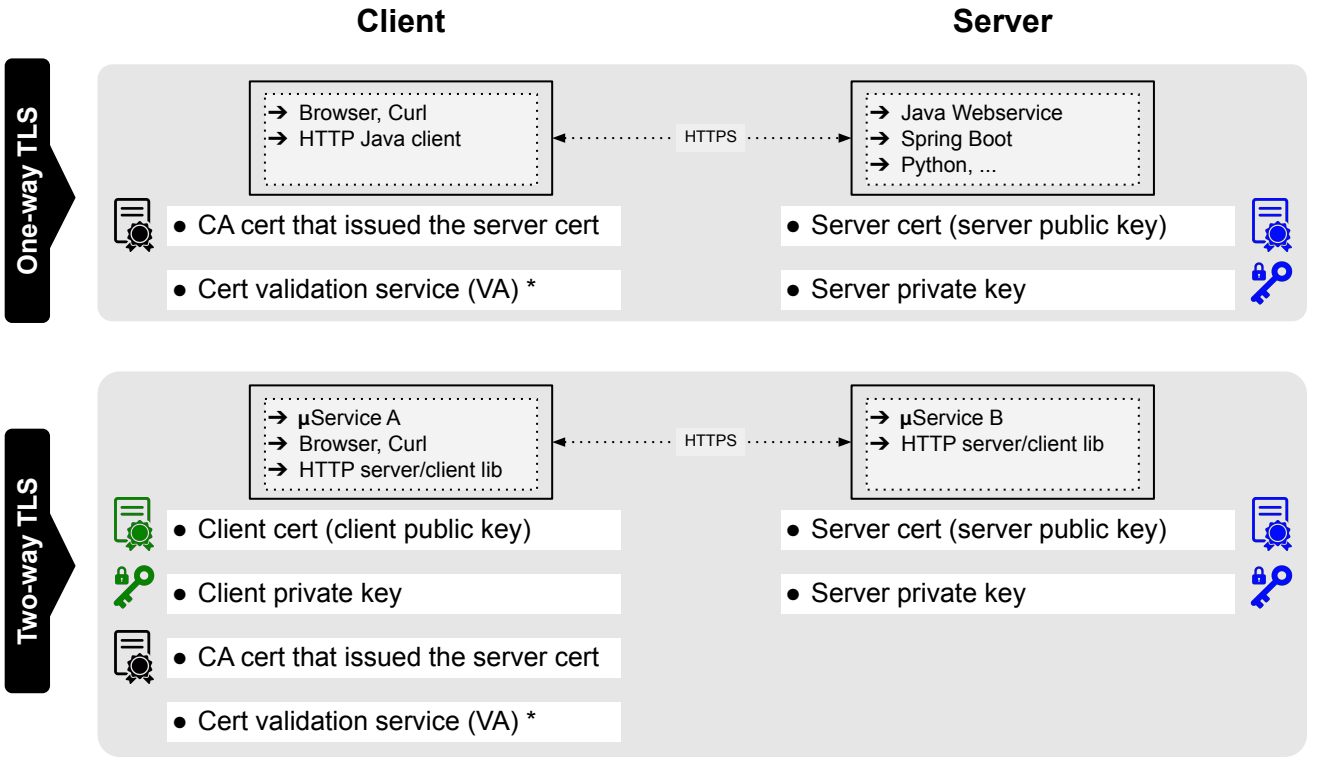
MVS: minimum amount of security controls implemented in order to make ~80% secure.

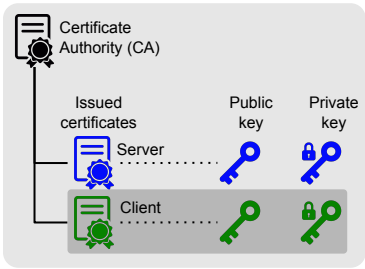
Use case 1:

MTLS for Java, Go, Python, ...



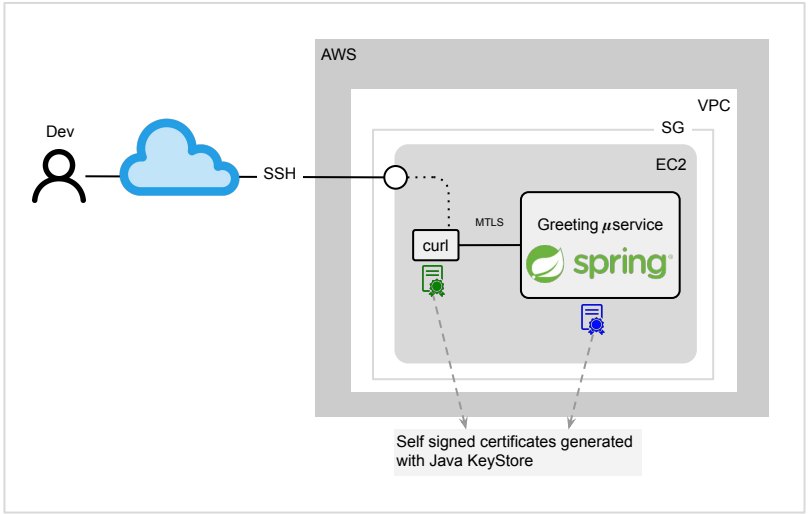
Use case 1: Java, Go, Python, ...



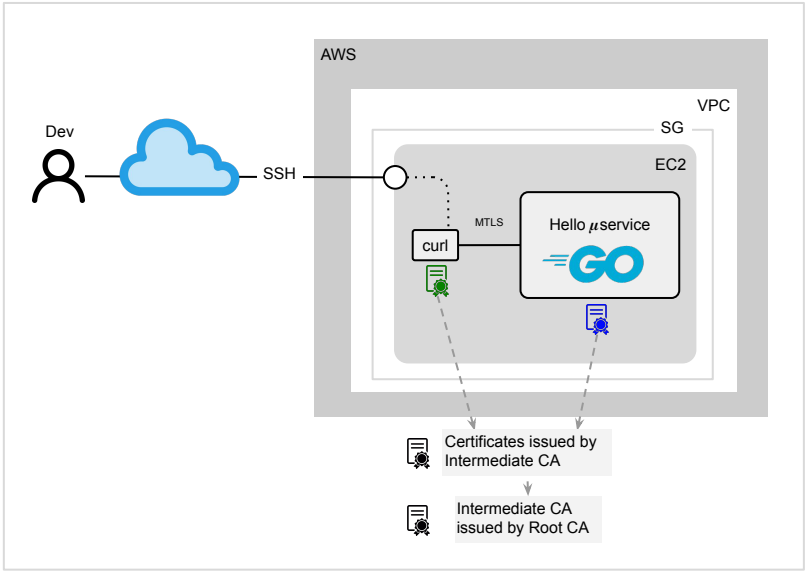


Use case 1: Hands-on time!

Lab #1: Self-signed certificate



Lab #2: Root CA and Intermediate CA

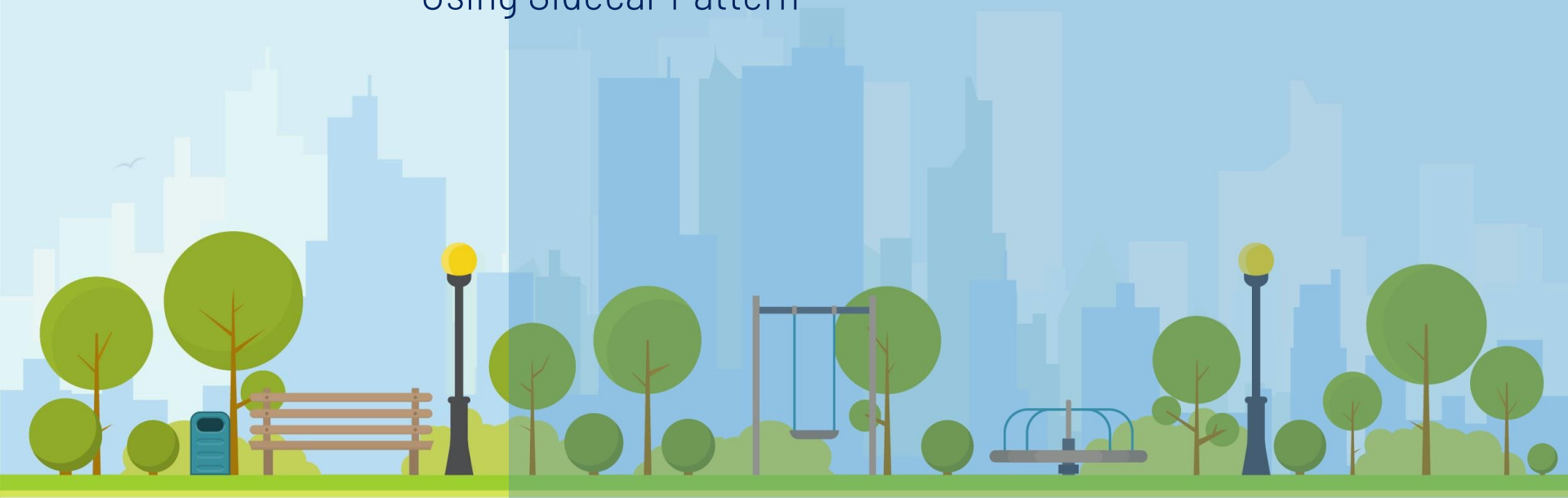


Use case 1: Conclusions

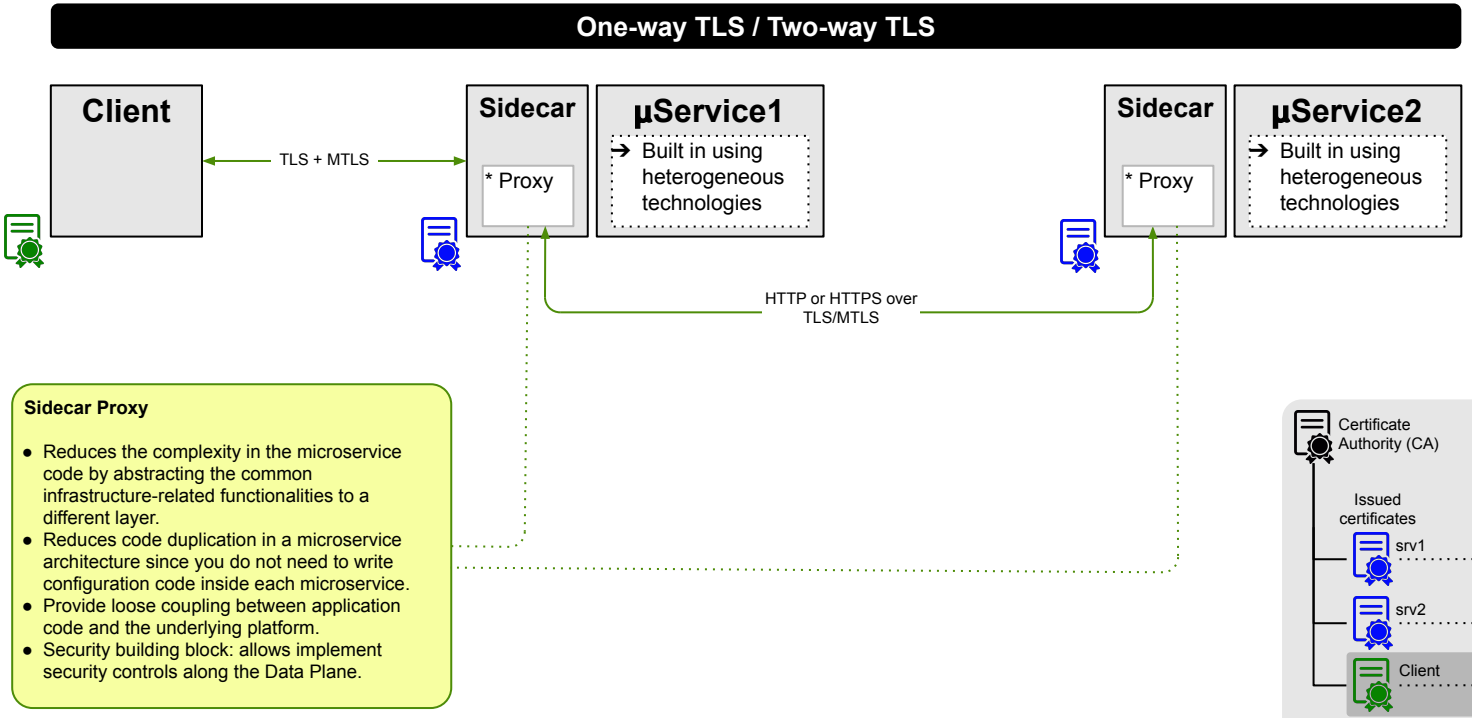
- Self-signed certificates are for testing, but definitely it has more problems than benefits.
  - Use a proper PKI with Root CA and Intermediate CA to issue certificates.
- The key-pair and certificate provisioning and its configuration depend on program language used.
  - Java recommends KeyTool and others lays on OpenSSL.
  - TLS and Cryptography specifics must be coded in the language selected.
- The certs, key-pair lifecycle (rotation, revocation, validation, etc.) management is manual.
  - We can do above operations for a few services, but not for over x10.
- The TLS configuration and keys distribution are manual tasks and can not scale.
  - Initialization and distribution of configuration and keys must be part of CI/CD Pipelines to be able to scale.

Use case 2:

Using Sidecar Pattern

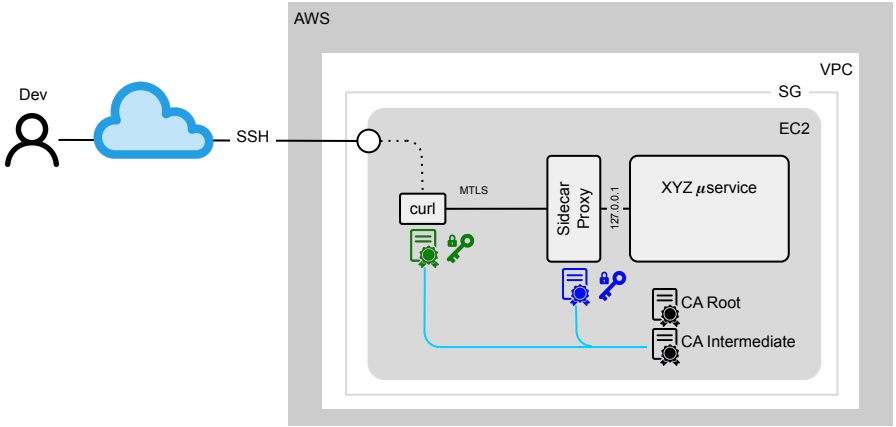


Use case 2: Sidecar Pattern



Use case 2: Hands-on time!

Lab #3: Sidecar Proxy



Sidecar Pattern:
- Proxy, Ingress/Egress
- Acme protocol
- Secret Discovery Service (SDS)

Caddy
https://caddyserver.com/docs/quick-starts/reverse-proxy

CA & PKI:
- Step Certificates
- Acme protocol
- Step SDS

smallstep™
https://github.com/smallstep/step-sds
https://github.com/smallstep/certificates

Use case 2: Conclusions

- Adopt a proper PKI tooling that has a proven mature API.
  - We are using SmallStep (Acme (RFC 8555) protocol support) Certificate as PKI to get an internal CA.
- Adopt Sidecar Proxy.
  - Caddy Server is a HTTP Lightweight Proxy, written in Go without dependencies, powerful API, support HTTPS by default and obtain automatically a Let's Encrypt TLS Certificates and ZeroSSL, it supports Acme protocol.
  - Lack of monitoring and telemetry features (?).
  - Chicken-Egg problem: No clear Secret Bootstrapping Process (?).
- Secure Data Plane, your Service Mesh MVP.
  - It does not have all the support of the marketing machinery that Istio, and therefore Envoy Proxy, does, but really you can build a secure Data Plane for your microservices from minute zero.
  - I've written an article about this:
    - https://holisticsecurity.io/2020/05/13/sidecar-proxy-the-security-building-block

Thank you

Any questions? Stay for a chat!