CEI : Analyse des protocoles standards d'échange pour la détection d'intrusion

ISSANCHOU DAMIEN – RICTER DIANE

PRÉSENTATION DU 07-02-2019

CentraleSupélec

La force de l'innovation

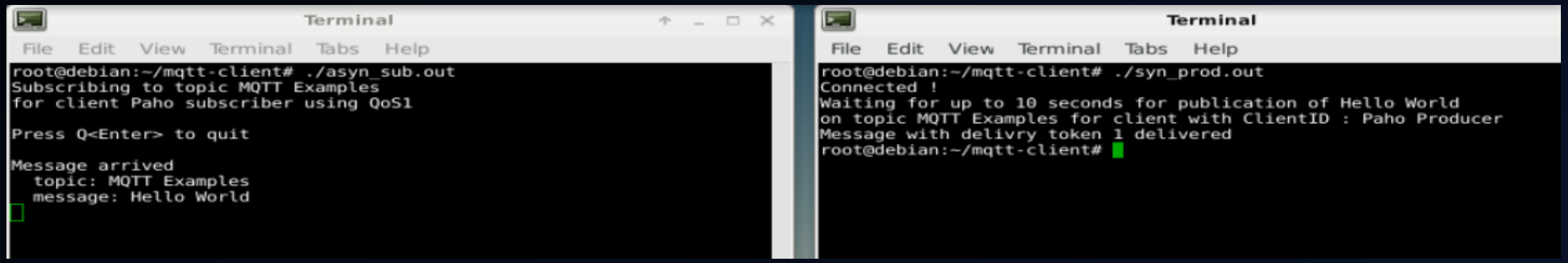
1. Exemple de MQTT et AMQP via TLS

```
[
        {rabbit, [{ ssl_options, [{ cacertfile, "/etc/rabbitmq/rmqtrustedCA.pem"},
                                  { certfile, "/etc/rabbitmq/certs/rmq.pem"},
                                  { keyfile, "/etc/rabbitmq/certs/rmq.key"},
                                  { verify, verify_peer},
                                  { fail_if_no_peer_cert, true}
                                 ]
                   }
                  ]
        },
        {rabbitmq_mqtt, [{allow_anonymous, false},
                         {vhost, "/"},
                         {exchange, "amq.topic"},
                         {subscription_ttl, 1800000},
                         {prefect, 10},
                         {ssl_listeners, [8883]},
                         {tcp_listeners, [1883]}
                        ]
     }

].
```

```c
int main(int argc, char *argv[])
{
    MQTTClient client;
    MQTTClient_deliveryToken token;
    MQTTClient_message pubmsg = MQTTClient_message_initializer;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_SSLOptions ssl_opts = MQTTClient_SSLOptions_initializer;

    /* return code */
    int rc;

    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);

    /* Setting connection options */
    conn_opts.keepAliveInterval = 20;
    conn_opts.cleansession = 1;

    conn_opts.username = USERNAME;
    conn_opts.password = PASSWORD;

    /* Settings SSL Options */
    ssl_opts.trustStore = TRUSTED_CERT_PATH ;
    ssl_opts.keyStore = CLIENT_CERT_PATH ;
    ssl_opts.privateKey = CLIENT_KEY_PATH ;
    ssl_opts.privateKeyPassword = "";
    conn_opts.ssl = &ssl_opts;

    if ( (rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS){
        printf("Failed to connect, return code %d\n", rc);
        exit(EXIT_FAILURE);
    }else
        puts("Connected !");

    /* Preparing Message */
    pubmsg.payload = PAYLOAD ;
    pubmsg.payloadlen = strlen(PAYLOAD);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;

    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    printf("Waiting for up to %d seconds for publication of %s\n"
           "on topic %s for client with ClientID : %s\n",
           (int) (TIMEOUT/1000), PAYLOAD, TOPIC, CLIENTID);
    rc = MQTTClient_waitForCompletion(client, token, TIMEOUT);
    printf("Message with delivry token %d delivered\n", token);

    /* Disconnect and Clean-up */
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
    return rc;
}
```

```c
int main(int argc, char *argv[])
{
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_SSLOptions ssl_opts = MQTTClient_SSLOptions_initializer;

    int rc, ch;

    MQTTClient_create(&client, ADDRESS, CLIENTID, MQTTCLIENT_PERSISTENCE_NONE, NULL);

    /* Setting connection options */
    conn_opts.keepAliveInterval = 20;
    conn_opts.cleansession = 1;
    conn_opts.username = USERNAME;
    conn_opts.password = PASSWORD;

    /* Setting SSL Options */
    ssl_opts.trustStore = TRUSTED_CERT_PATH ;
    ssl_opts.keyStore = CLIENT_CERT_PATH ;
    ssl_opts.privateKey = CLIENT_KEY_PATH ;
    ssl_opts.privateKeyPassword = "";
    conn_opts.ssl = &ssl_opts;

    MQTTClient_setCallbacks(client, NULL, connlost, msgarrvd, delivered);

    if ( (rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS){
        printf("Failed to connect, return code %d\n", rc);
        exit(EXIT_FAILURE);
    }

    printf("Subscribing to topic %s\nfor client %s using QoS%d\n\n"
           "Press Q<Enter> to quit\n\n", TOPIC, CLIENTID, QOS);
    rc = MQTTClient_subscribe(client, TOPIC, QOS);
    if ( rc != MQTTCLIENT_SUCCESS ) {
        printf("Failed to subscribe, return code %d\n", rc);
        exit(EXIT_FAILURE);
    }
    /* Loop until quitting */
    do{
        ch = getchar();
    }while (ch !='Q' && ch != 'q');

    /* Disconnect and clean-up */
    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
    return rc;
}
```

1. Exemple de MQTT via TLS

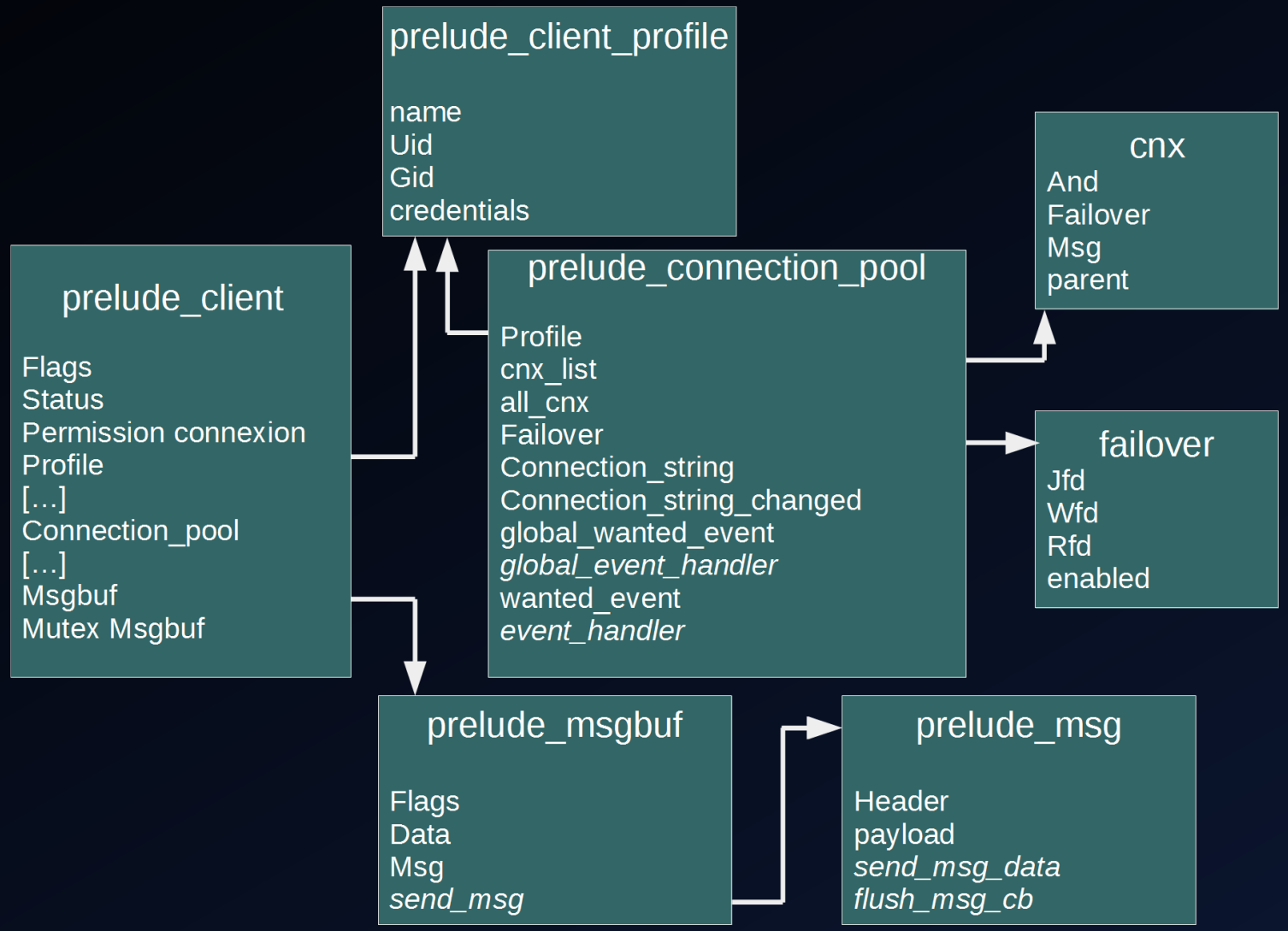


1. Exemple de AMQP via TLS

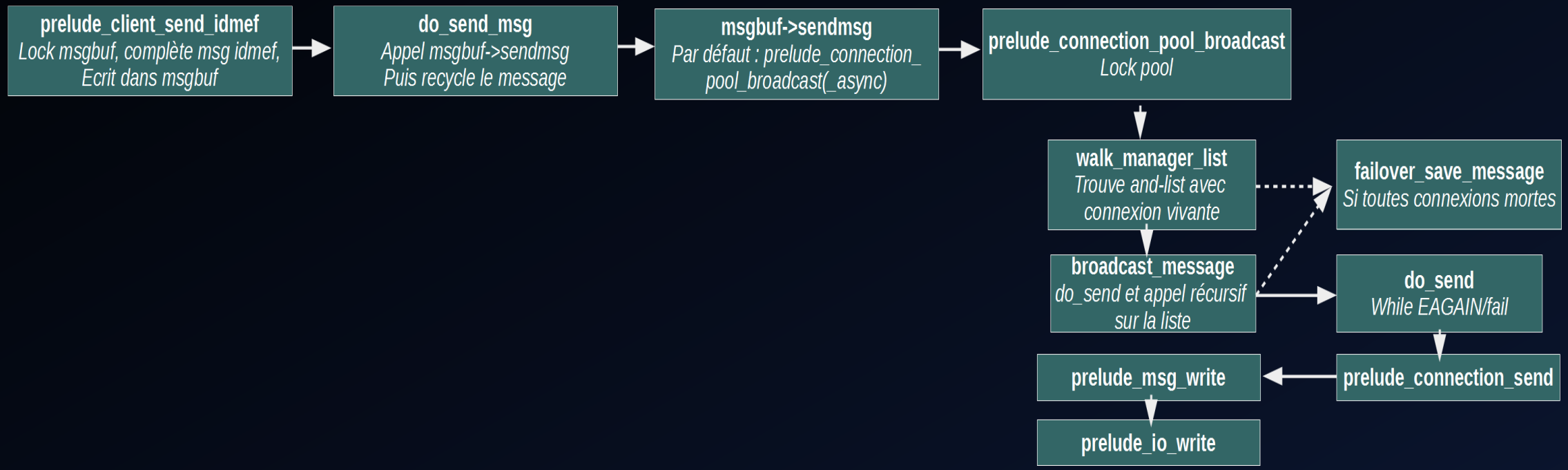
```python
import pika
import sys
import logging
import ssl

logging.basicConfig(level=logging.INFO)

cp=pika.ConnectionParameters(
ssl=True,
ssl_options = dict(
ssl_version=ssl.PROTOCOL_TLSv1,
ca_certs="/home/diane/tls-gen/basic/result/ca_certificate.pem",
keyfile="/home/diane/tls-gen/basic/result/client_key.pem",
certfile="/home/diane/tls-gen/basic/result/client_certificate.pem",
cert_reqs=ssl.CERT_REQUIRED))


connection = pika.BlockingConnection(cp)
channel = connection.channel()

channel.queue_declare(queue='hello')
message ="Test"

channel.publish(exchange='', routing_key='hello', body=message)
print(" [x] Sent %r" % message)
connection.close()
```

```python
import pika
import time
import logging
import ssl

logging.basicConfig(level=logging.INFO)

cp= pika.ConnectionParameters(ssl=True, ssl_options = dict(
ssl_version=ssl.PROTOCOL_TLSv1, ca_certs="/home/diane/tls-gen/basic/result/ca_cert$
keyfile="/home/diane/tls-gen/basic/result/client_key.pem",
certfile="/home/diane/tls-gen/basic/result/client_certificate.pem",
cert_reqs=ssl.CERT_REQUIRED))



connection = pika.BlockingConnection(cp)
channel=connection.channel()

channel.queue_declare(queue='hello')

def callback(ch, method, properties, body):
        print("Received %r" % body)
        time.sleep(body.count(b'.'))
        print("[x] Done")

channel.basic_consume(callback, queue='hello')

print('Waiting for messages')
channel.start_consuming()
```

2. LibPrelude



2. LibPrelude



3. Difficultés et Questions

- Certificat : garder le point à point ou faire une PKI ?

- Difficultés :
  - Téléchargement et Installation LibPrelude
    - A venir : un récap détaillé