DTLS Implementation for CoAP Server

By Manik Singla 222092193, 22/09/2024

# Steps taken and validated during PoC

1. **Set Up EC2 Instance:** I launched an EC2 instance by selecting the Amazon Machine Image (AMI) for Ubuntu 22.04 to ensure compatibility with the software I intended to install. For the instance type, I chose the t2.micro, which offers sufficient resources for my needs while remaining cost-effective. I then configured the security groups to allow port 22 for SSH, UDP traffic on port 5683 for CoAP and 5684 for DTLS, as well as HTTP and HTTPS traffic as necessary for web services. Used EC2 Instance connect to connect to the machine

A screenshot of a computer

Description automatically generated

1. **Setting Up Domain and Elastic IP:** To ensure reliable access to the CoAP server, I set up a domain and associated it with an Elastic IP. First, I allocated an Elastic IP address in the AWS Management Console, which provides a static IP that remains the same even if the EC2 instance is stopped or restarted. I then associated this Elastic IP with my EC2 instance to enable direct access over the internet. Next, I registered a domain through a domain registrar and configured the DNS settings to point to the Elastic IP.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

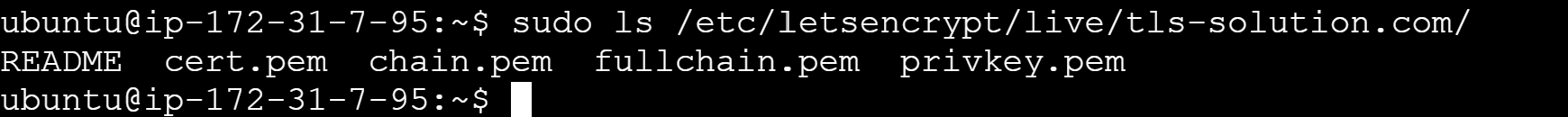
1. **Installing Certbot and Obtaining ECDSA Certificates:** I began by updating my EC2 instance to ensure all packages were current and installing certbot. This was accomplished with the following commands:

**sudo apt-get update**

**sudo apt-get install certbot**

After successfully installing Certbot, I proceeded to obtain an ECDSA certificate for my domain, tls-solution.com. To obtain the certificate, I executed the following command:

**sudo certbot certonly --standalone --preferred-challenges http --key-type ecdsa -d tls-solution.com**



This process ensured that my CoAP server would be able to establish secure

communications using DTLS.

1. **Convert Certificates to PKCS12 Format:** I converted the obtained .pem files into a .p12 format using following command, which is necessary for enabling DTLS in the CoAP server. This conversion was accomplished using the openssl command. The command utilized the fullchain.pem certificate and the associated private key (privkey.pem) to create a single PKCS12 file named coap-server.p12. This format is essential for secure communication in the CoAP server setup.

**openssl pkcs12 -export -in /etc/letsencrypt/live/tls-solution.com/fullchain.pem -inkey /etc/letsencrypt/live/tls-solution.com/privkey.pem -out coap-server.p12 -name "coap-server"**

1. **Install Prerequisites and Libcoap from Source:** To install libcoap, I first ensured that the necessary development tools and libraries were available on the EC2 instance. This included installing pkg-config, libtool, autoconf, automake, build-essential, as well as asciidoc and doxygen, which are useful for documentation. I executed the following command:

**sudo apt install pkg-config libtool autoconf automake build-essential asciidoc doxygen**

1. After installing the prerequisites, I cloned the official libcoap repository from GitHub. This was done using the git command, which fetched the latest version of the source code. I then navigated to the cloned directory to initiate the build process:

**git clone** [**https://github.com/obgm/libcoap.git**](https://github.com/obgm/libcoap.git)

**cd libcoap**

**./autogen.sh**

The ./autogen.sh script sets up the configuration files necessary for building the library from source. This step is crucial for ensuring that libcoap is correctly compiled and configured for use in the CoAP server setup.

1. **Configure and Install Libcoap with DTLS Support:** After running the ./autogen.sh script, I configured the libcoap build to enable shared libraries and DTLS support. This was accomplished using the following command:

**./configure --enable-shared --enable-dtls**

Once the configuration was complete, I proceeded to compile the library by executing:

**make**

Finally, I installed the compiled library onto the system with:

**sudo make install**

This step ensured that libcoap was successfully installed and ready for use in the CoAP server setup, including DTLS functionality.

1. **Configure CoAP Server for DTLS Support Using PKCS 12 File:** To configure the CoAP server (using libcoap) to support DTLS, I started by converting the coap-server.p12 file into separate PEM files for the certificate, private key, and CA certificate. This process was done using OpenSSL.

I ran the following commands to extract the necessary components from the .p12 file:

* **Extract the Private Key:**

**Sudo openssl pkcs12 -in coap-server.p12 -nocerts -out server-key.pem -nodes**

* **Extract the Certificate:**

**Sudo openssl pkcs12 -in coap-server.p12 -clcerts -nokeys -out server-cert.pem**

* **Extract the CA Certificate:**

**Sudo openssl pkcs12 -in coap-server.p12 -clcerts -nokeys -out server-cert.pem**

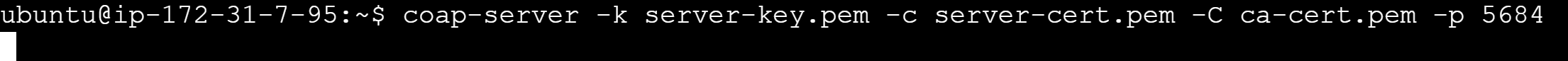
These commands provided the necessary PEM files to configure the CoAP server for secure communication using DTLS.



1. **Configure the CoAP Server to Use PEM Files for DTLS:**

With the PEM files ready, I configured the CoAP server (using libcoap) to utilize these files for DTLS. To launch the CoAP server with DTLS support on port 5684, I executed the following command:

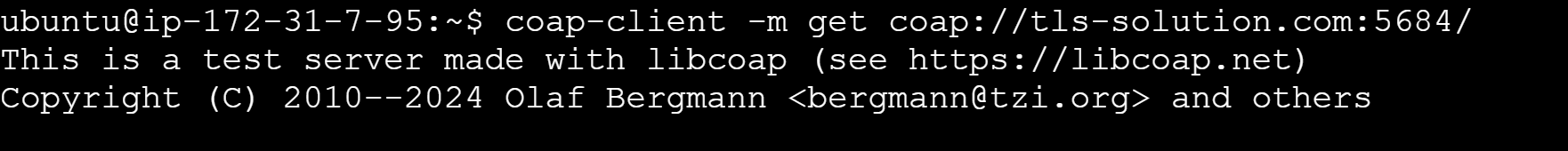
**coap-server -k server-key.pem -c server-cert.pem -C ca-cert.pem -p 5684**



1. **Verification of DTLS Setup:** After starting the CoAP server, I verified its operation with DTLS using a compatible CoAP client.

Command used to Test the Server:

**coap-client -m get coap://tls-solution.com:5684/**



# Conclusion: Summary of PoC Outcomes

In this PoC, I successfully configured the CoAP server to support DTLS using certificates generated via Certbot. The server was deployed on an EC2 instance, secured using ECDSA certificates, and configured to handle secure CoAP communication over DTLS (port 5684). After setting up the certificates and configuring the server, I verified the functionality using the coap-client to ensure the communication was encrypted and secure.

The steps documented, from EC2 setup to configuring libcoap, validate that DTLS is now fully operational, providing enhanced security for CoAP communication. This PoC demonstrates how CoAP servers can be secured efficiently using modern DTLS standards, supporting the secure transmission of data in IoT networks.