Logo

Description automatically generated

# TFG S7Comm

Antonio Pérez Sánchez

Miguel Oleo Blanco



Table of Contents

[TFG S7Comm 1](#_Toc67671969)

[Protocolo 3](#_Toc67671970)

[Bibliografía 4](#_Toc67671971)

## Protocolo

S7Comm es un protocolo propietario de Siemens para las comunicaciones entre sus equipos (PLCs, HMIs …). Este protocolo fue introducido con la familia de PLCs S-300 y S-400 y desde entonces sigue en desarrollo constante. Al ser un protocolo propietario, la cantidad de información que tenemos disponible es muy limitada. También hay que indicar que como estos equipos trabajan en sectores donde los ataques pueden llegar a ser catastróficos, el secretismo de este es muy importante. Un ejemplo del impacto que puede tener un ataque sobre estos sistemas fue el ataque que se lanzó contra una planta de enriquecimiento de uranio en Irán. Se consiguió inyectar un programa malicioso sobre unos PLCs, consiguiendo modificar unos parámetros de la centrifugadora a la vez que se mandaban datos correctos a las estaciones de ingeniería [1].

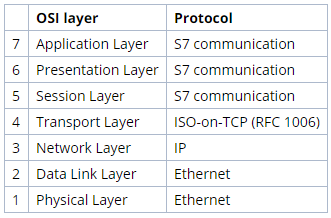
 Este protocolo manda los datos en el payload de COTP (Connection Oriented Transport Protocol). Las capas del protocolo son los siguientes:

Tabla 1: Capas del protocol S7comm

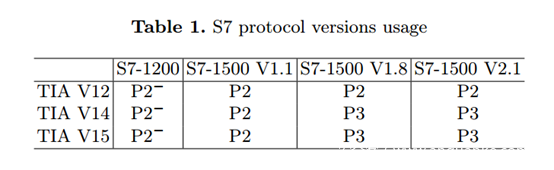
 En concreto en este trabajo nos centraremos en una variante de este protocolo. Esta variante es el S7comm-plus y salió para usarse en equipos modernos. Este protocolo empezó con el S7-1200 PLC. Principalmente ese protocolo es una mejora en cuanto a seguridad del S7comm. El s7comm+ integra completamente un sistema de autenticación y encriptación de los datos de la comunicación. Entre otras cosas, esto hace que ataques como un replay de un paquete sencillo sean mucho más complejos de efectuar. Para poder realizar los ataques, hay que entender en profundidad cómo funciona la seguridad de este protocolo. La gran mayoría de información se encuentra para los PLCs de la familia S7-1200, pero nosotros vamos a investigar sobre el S7-1500. También es importante saber que dependiendo de la versión del TIA-Portal que empleemos, se usará una versión del protocolo distinto. A continuación, se muestra una tabla con estas versiones, donde se puede ver que para la versión V16 (la presente en el laboratorio) no tenemos información:

Tabla 2: Versiones S7comm plus dependiendo de PLC y versión de TIA Portal

Antes de ver como se efectúa el proceso de handshake, es necesario saber que wireshark no nos muestra este tráfico por defecto, ya que tenemos que añadir el disector. Este lo podemos conseguir en la referencia [3] y según el sistema operativo se añade de formas distintas. En sistemas Windows basta con descargar el .dll y ponerlo en la carpeta de plugins de Wireshark. Por otro lado, para sistemas Linux, tenemos que descargarnos el código fuente del disector y el de Wireshark, juntar los dos códigos (el disector hay que meterlo con el resto) y compilar la aplicación.

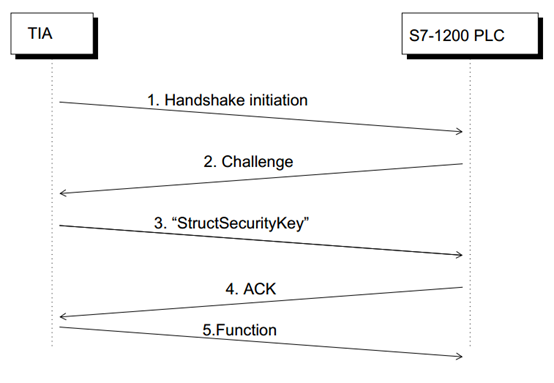
 Este proceso de handshake es muy parecido al de otros protcolos, en los que se negocia las claves que se van a emplear en la comunicación para asegurar la integridad de los datos y la seguridad del sistema. A continuación, se muestra como es este proceso de handshake en un diagrama secuencial (para un S7-1200):

Figura 1: Proceso de handshake [4]

En la siguiente imagen se muestra el tráfico generado en este proceso y se detallará el contenido de estos mensajes:

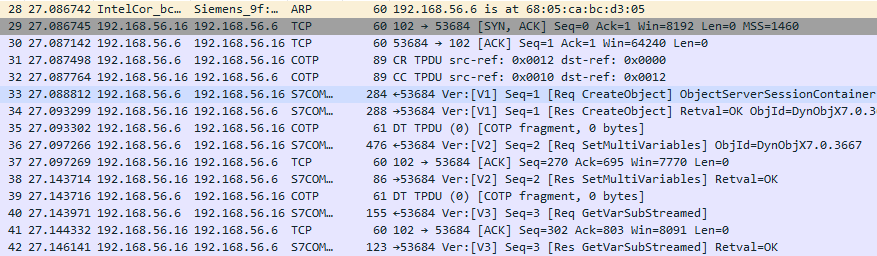


Figura 2: Tráfico del proceso de handshake

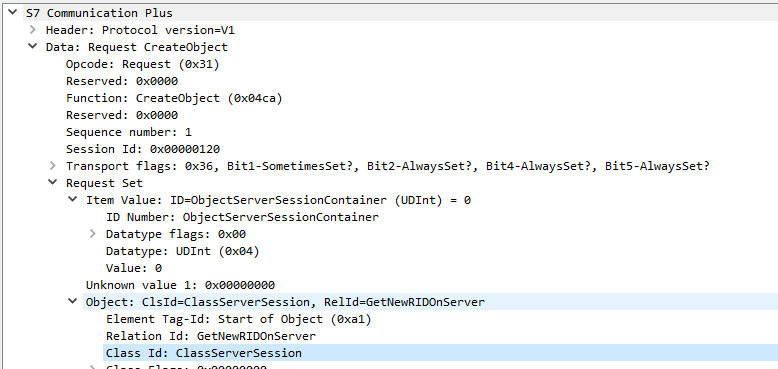
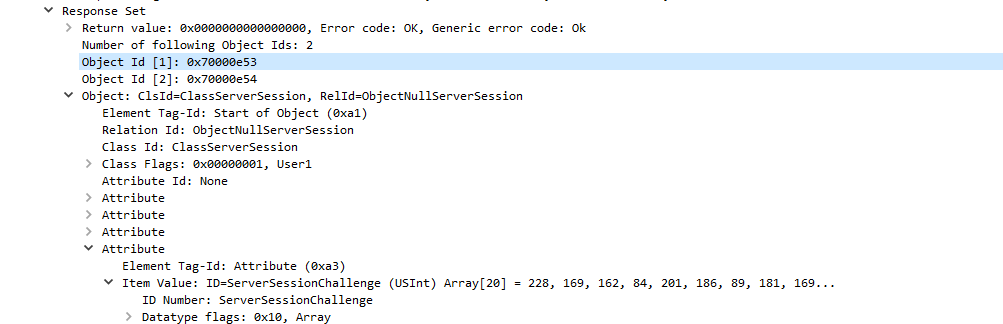
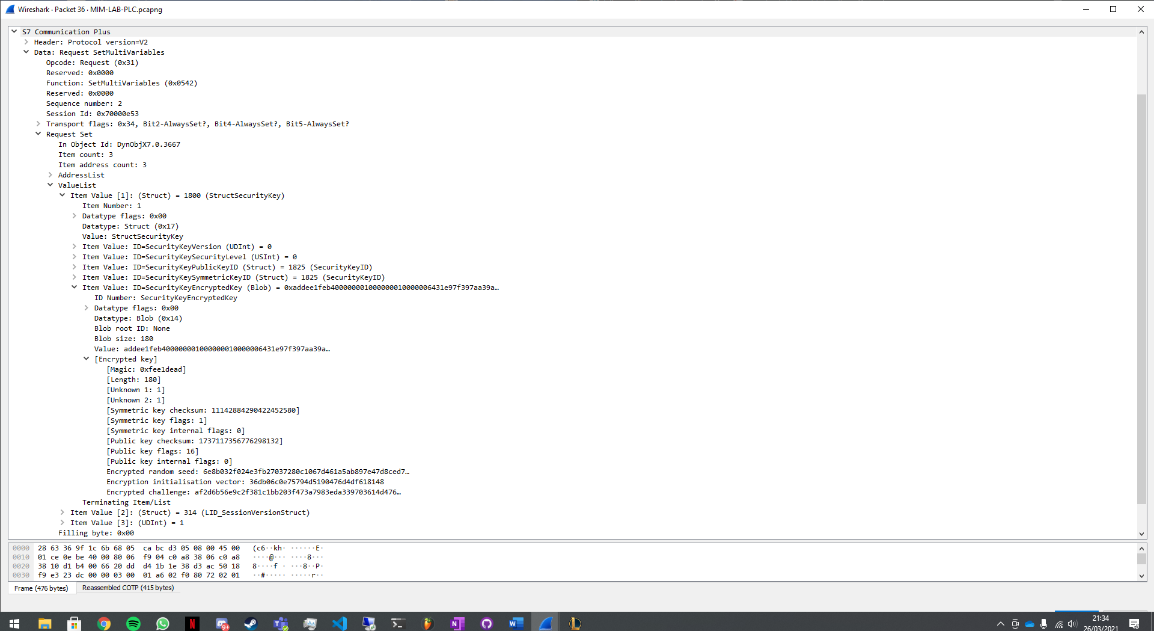
La fase de iniciación del handshake son los paquetes 31 y 32. Estos paquetes contienen los campos Connect Request (CR) y Connect Confirm (CC) respectivamente. La fase del Challenge son los paquetes 33 y 34. En este punto, el plc genera 20 bytes aleatorios que se usarán para identificar la sesión y otros parámetros. El paquete 33 es el que manda TIA al PLC y en este se solicita crear el objeto ClassServerSession:

Figura 3: TIA solicita al PLC crear el objeto ClassServerSession

A este mensaje le responde con la información sobre el firmware que el PLC esta corriendo y 20 bytes generados aleatoriamente citados anteriormente (ServerSessionChallenge). También se manda el sessionID, que se emplea para saber que mensajes son de cada sesión:



La siguiente fase es la llamada StructSecurityKey y se corresponden a los paquetes 36 y 38. TIA al recibir los 20 bytes del PLC, emplea los 16 bytes centrales para cálculos de la clave publica de la sesión. Para ello utiliza complejos algoritmos criptográficos, desde más sencillos como XOR, pasando por SHA-156, HMAC-SHA-256, CBC-MAC, AES-ECB y ECC. Con esto genera datos de autenticación que enviará al PLC. Esta información también la podemos ver dentro del paquete:

El PLC al recibir esto, lo desencriptará con su clave privada. Si este proceso es correcto, se lo avisará al TIA. En nuestro ejemplo, esta respuesta es la del paquete 38.A partir de este punto, la comunicación está establecida y la comunicación empieza a transportar el tráfico generado por el programa específico.

## Bibliografía

[1] Ataque a Iran

<https://en.wikipedia.org/wiki/Stuxnet>

[2] Documentación S7comm Wireshark

<https://wiki.wireshark.org/S7comm>

[3] Disector S7comm+ Wireshark

<https://sourceforge.net/projects/s7commwireshark/>

[4] Documentación sobre S7comm-plus

<https://www.programmersought.com/article/64975702507/>