

# POWERED LOGICAL CONTROLLER

DC11331

28/11/2019



# Who am I

---

- **Twitter** : @SeckKoala
- **Company** : Risk&CO
- **Interest** : fuzzing, pentest, industrial, etc.

**RISK & CO**  
G R O U P

# POWNED LOGICAL CONTROLLER

## OBJECTIF DE LA PRÉSENTATION

1. Comprendre le contexte d'attaquer pour un PLC
2. Comprendre l'analyse de firmware
3. Comprendre les protocoles de communication
4. Apprendre à fuzzer les protocoles industriels.

# Programmable logic controller (PLC)

**Définition :** Un PLC est un équipement industriel permettant la supervision ou le contrôle d'une valeur/d'un actionneur. Il peut être programmé en fonction de ces entrées et de ces sorties pour fonctionner de manière autonome.

## Ces différents composants :

- Entrée/sortie analogique/numérique ;
- Point terminal du Réseau industriel ;
- Contrôle un point spécifique ;
- Supervision et reprogrammation par IP ;
- Plusieurs cartes possibles (et indépendante).

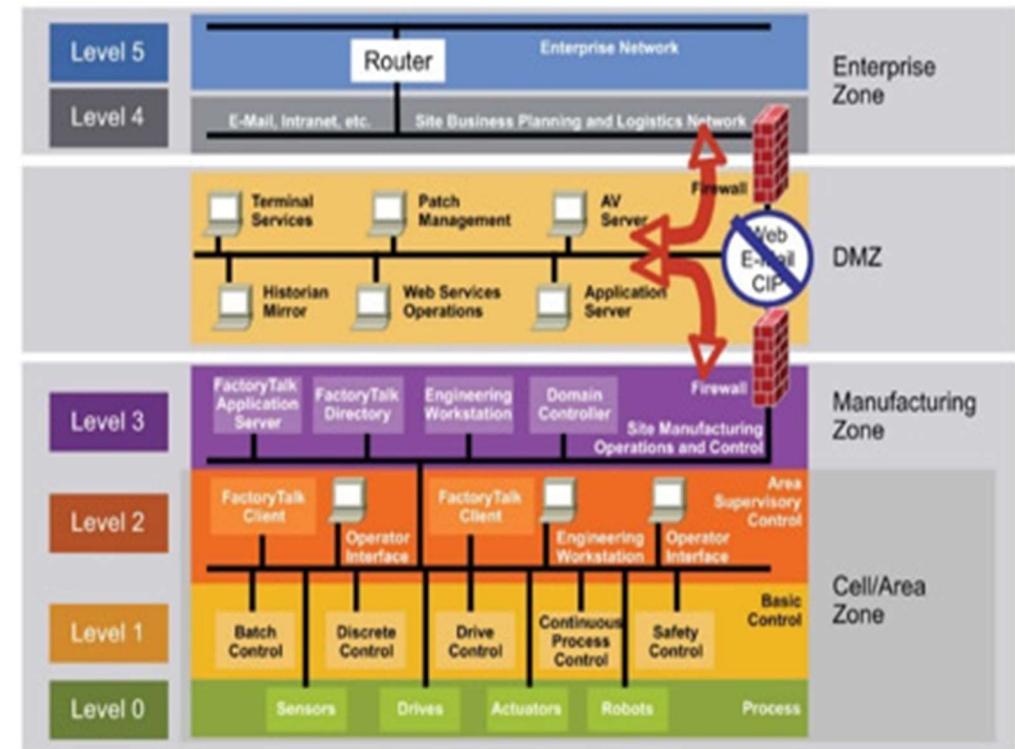


# Industrial network - Infrastructure

## (ISA 99/IEC 62443)

IEC 62443, appelé « modèle de purdue », propose un standard industriel abordant une architecture maintenable et sécurisée. Il se résume en plusieurs étages :

- **Level 0** : Équipement de terrain (capteurs/actuateurs) ;
- **Level 1** : PLC/RTU ;
- **Level 2** : Operator/engineering station ;
- **Level 3** : Control/supervision system.



# Industrial network - Infrastructure

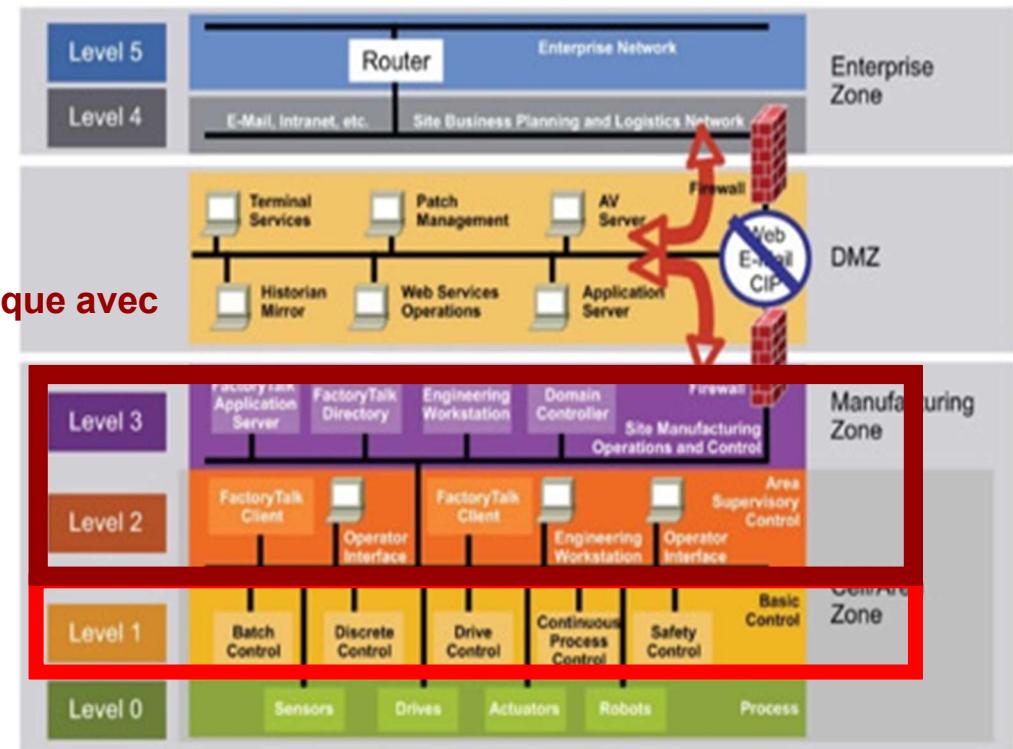
(ISA 99/IEC 62443)

IEC 62443, appelé « modèle de purdue », propose un standard industriel abordant une architecture maintenable et sécurisée. Il se résume en plusieurs étages :

- **Level 0** : Équipement de terrain (capteurs/actuateurs) ;
- **Level 1** : PLC/RTU ;
- **Level 2** : Operator/engineering station ;
- **Level 3** : Control/supervision système.

PLC sont dans le niveau 1, relié au niveau 2-3 en IP le plus souvent. (le cas qui nous intéresse)

Communique avec



# What can affect a PLC

Les PLC permettent le bon fonctionnement d'un process métier, chaque PLC contrôle une tâche unitaire précise.

Cependant :

- Les PLC sont régulièrement sur le même réseau, une cyberattaque peut impacter toute une usine ;
- Il est courant de rencontrer les mêmes PLC sur une même usine ;
- Souvent des configurations similaires (niveau de mise à jour, configuration métier, etc.)

# Vulnérabilité

## Impact

Vulnérabilité	Impact
<b>Change programmed logic</b>	Modifie la logique métier, peut engendrer aussi une interruption de la disponibilité qu'un impact humain (dépend de la tâche attribué).  Si le process n'est pas arrêté dans un état stable : des risques humains/environnementaux sont à craindre
<b>Change/remove supervision</b>	Perte de la visibilité des actions effectuées sur le réseau. Il n'est plus possible de visualiser ou de faire confiance à la supervision.  Dans le cas d'usine avec des process critique. L'usine est stoppée.
<b>Default password/backdoor</b>	Compromission de l'intégrité du PLC, la logique peut être modifiée par la suite.
<b>Remote code execution</b>	Compromission de l'intégrité du PLC, la logique peut être modifiée par la suite.

# Firmware

## Introduction

Définition : Le firmware est un fichier contenant toutes les informations nécessaires au bon fonctionnement d'un PLC.

Ils sont disponibles sous 3 formats :

- **Système de fichier/binaire standalone** : un système de fichier exécutable standalone ;
- **Archive** : une archive contenant toutes les informations nécessaires ;
- **Chiffré** : un fichier binaire ou archive chiffrée.

# Firmware

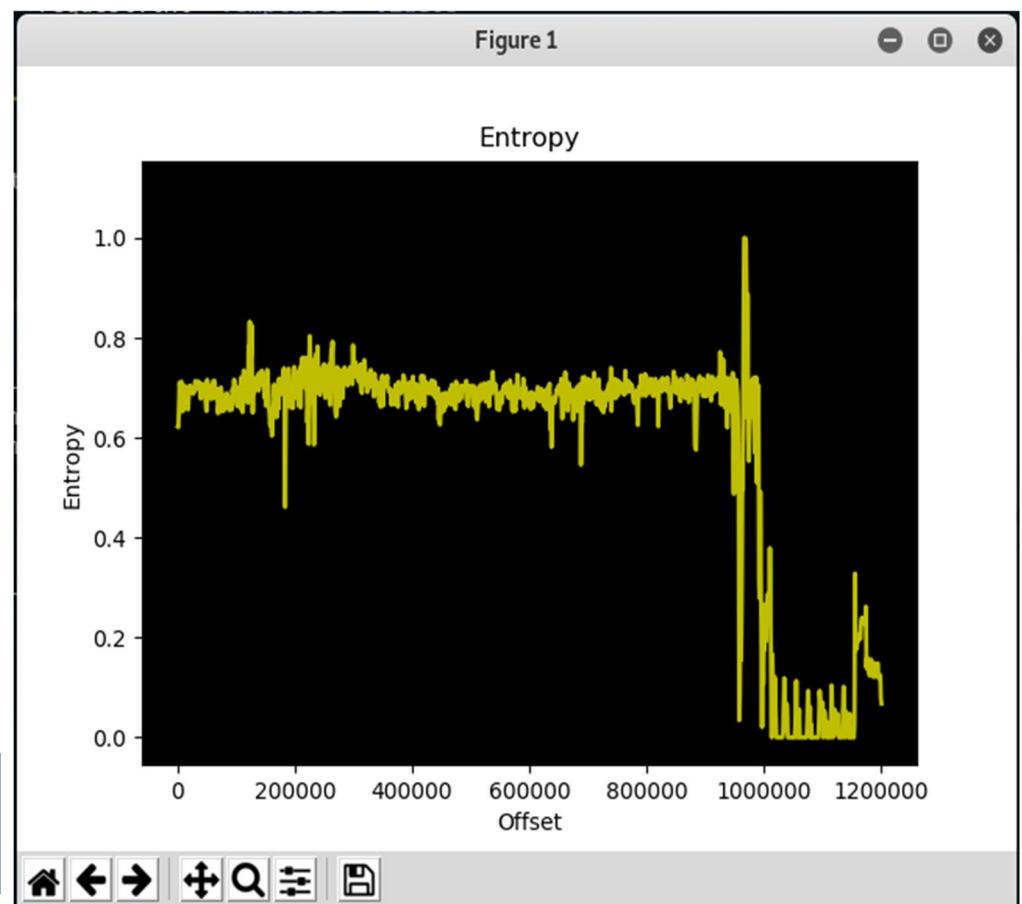
# Système de fichier / binaire standalone

Majoritairement le système de fichier d'un système d'exploitation (VxWorks, QNX).

Fichier binaire, contenant le code exécuté sur l'automate.

- Architecture souvent rencontrée :
    - ARM
    - PowerPC

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	ARMeb (size=0xe7800, entropy=0.763488)
948224	0xE7800	None (size=0x3a800, entropy=0.269286)
1187840	0x122000	OCaml (size=0x3000, entropy=0.137257)

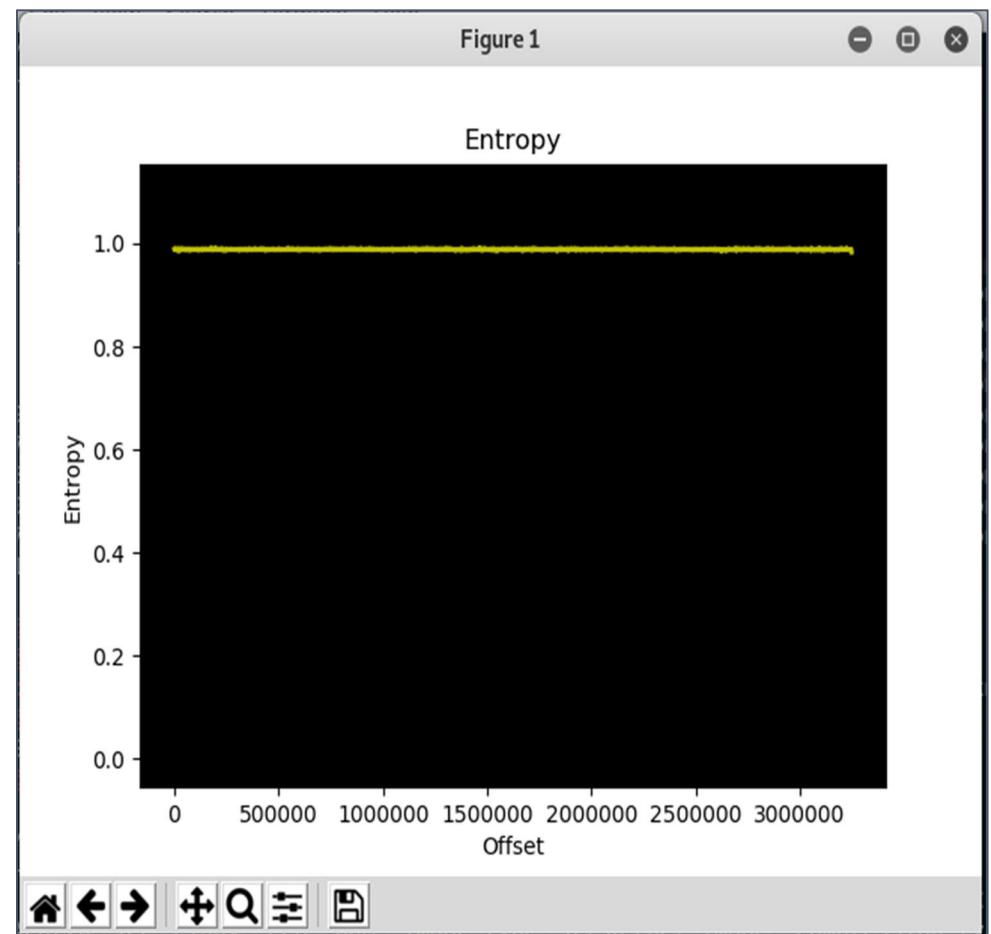


# Firmware

## Chiffré

- Entropie : 1 (ou très proche)
- L'ensemble du fichier est chiffré, courant chez les grands constructeurs ou les automates sensibles.

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	6502 (size=0x319000, entropy=0.999994)

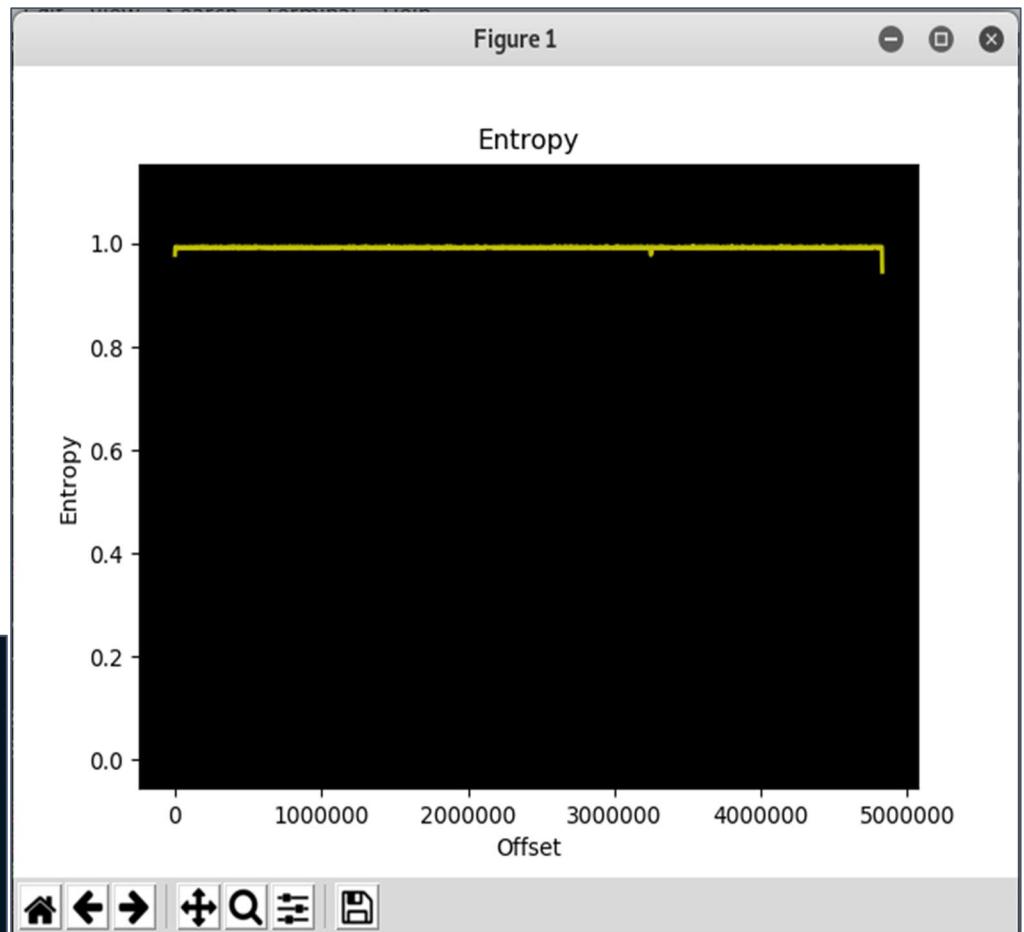


# Firmware

## Archive

- Entropie : 0.8 - 1.0
- Plus courants, les firmwares peuvent être des archives. Les archives vont contenir :
  - Les fichiers de configurations ;
    - Les services/pages web/etc. ;
    - Le code compilé (ex: .img .bin) ;
    - Système d'exploitation (ex.: VxWorks, QNX).

```
v2.0 to extract, compressed size: 60, uncompressed size: 98, name: CleanUp.ftp
v2.0 to extract, compressed size: 159, uncompressed size: 271, name: fw.ini
v2.0 to extract, compressed size: 69, uncompressed size: 127, name: M580SMP.ftp
v2.0 to extract, compressed size: 3250084, uncompressed size: 3249584, name: M580SMP.img
v1.0 to extract, compressed size: 576, uncompressed size: 576, name: M580SMP SIG.img
v2.0 to extract, compressed size: 982, uncompressed size: 2272, name: Osl.xml
v2.0 to extract, compressed size: 69, uncompressed size: 127, name: webpage.ftp
v2.0 to extract, compressed size: 1578501, uncompressed size: 1578256, name: webpage.img
v1.0 to extract, compressed size: 576, uncompressed size: 576, name: webpage_sig.img
length: 22
```



# Firmware

## Custom header

	Magic word	Version	checksum	Langue	
00000000	01	05 14 0a	00 0d 35 bc	37 09 10 62 18 09 0e 09	.....5.7..b....
00000010	1c 38	00 00 00 00 00 00	00 00 00 00 00 00	00 01	.8.....
00000020	40 02 00 3c	00 06 00 00	00 26 16 61	00 26 6b 98	@..<.....&.a.&k.
00000030	00 26 6b 98	00 52 13 e0	00 06 00 00	1f 8b 08 08	.Gk.R.....
00000040	b8 72 22 54	00 0b 62 61	73 65 2e 74 6d 70 2e 62		r"T..base.tmp.bl
00000050	69 6e 2e 70	30 32 00 cc	7c 79 5c 53 47 d7 ff b9		in.p02.. y\SG...
00000060	10 30 60 d0	68 51 83 22	04 45 05 0d 42 15 eb 4d		.0` .hQ." .E..B..M
00000070	c3 12 04 35	28 20 22 28	56 94 c4 24 48 ca 12 84		...5( "(V..\$H...
00000080	e0 52 41 c2	e2 42 05 4b	8a 6d dd 50 ac 4b ab 55		.RA..B.K.m.P.K.U
00000090	eb da d2 aa	55 5b b5 76	b1 d2 d6 b6 da ba 60 5d		....U[.v.....`]
000000a0	aa ad fa d0	ba b6 2a f9	9d b9 f7 12 a2 a5 7d df		.....*.....}.

GZIP magic word

GZIP data

- Les firmwares constructeurs comportent régulièrement des headers spécifiques, ils sont utilisés pour ajouter une description ou une validation de l'intégrité du firmware.

Ces headers comportent régulièrement :

- Les metadata (version langue) ;
- La taille du firmware ;
- Vérification/signature/checksum ;
- ...

# Firmware

## Vulnérabilité

Comportement recherché pour les binaires :

- Gestion des utilisateurs/mot de passe/mot de passe par défaut ;
- Entrée réseau (Port/protocole) ;
- Protocole propriétaire ;
- Interface web de management ;
- ...

# Firmware

## Les points recherchés

### Les vulnérabilités recherchées dans un firmware sont :

- Fonctions d'administration/supervision/mise à jour ;
- Bug d'exécution ou de logique ;
- Mauvaise gestion des utilisateurs ;
- Etc.

### Les vulnérabilités recherchées dans les headers sont :

- Compromission de l'intégrité du firmware ;
- Extraction d'information sensible ;
- Etc.

# Firmware

## Exemple de vulnérabilité

### EXPERTS FOUND A BACKDOOR IN SIEMENS PLCS. CRITICAL INFRASTRUCTURE AND SCADA NETWORKS AFFECTED

---

A team of [web application security](#) specialists from Ruhr University in Bochum, Germany, has discovered a critical vulnerability in some new programmable logic controller (PLC) models manufactured by [Siemens](#). According to the experts, the flaw is related to the presence of a hidden access feature and could be exploited both to perform cyberattacks and security tool.

The security issue is related to the hardware access function of the Siemens S7-1200 PLC (this feature processes software updates and verifies the integrity of the PLC firmware when starting the device). Apparently, this access shows behavior similar to that of a [backdoor](#).

# Firmware

## Exemple de vulnérabilité

### CVE-2019-6567 Detail

MODIFIED

---

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

### Current Description

A vulnerability has been identified in SCALANCE X-200 (All Versions < V5.2.4), SCALANCE X-200IRT (All versions), SCALANCE X-300 (All versions), SCALANCE X-414-3E (All versions). The affected devices store passwords in a recoverable format. An attacker may extract and recover device passwords from the device configuration. Successful exploitation requires access to a device configuration backup and impacts confidentiality of the stored passwords. At the time of advisory publication no public exploitation of this security vulnerability was known.

# Protocole industriel

## Introduction

Description : Un protocole créé, le plus souvent par un constructeur, pour effectuer un échange de données ou d'ordre entre deux équipements industriels.

Les spécifications de ces protocoles ne sont souvent pas libres de droits et peuvent être payantes si elles ne sont pas fermées (propriétaire).

Le choix de la communication utilisé est basé sur :

- Le type de procédé/d'activité de l'usine ;
- Le support de communication (client/serveur) ;
- Les caractéristiques de l'équipement ;
- Etc.

# Protocole industriel

## Libre de droit - ouvert

La création d'une spécification et d'une implémentation d'un protocole industriel peut être ouverte au public. Les spécifications peuvent également être payantes et non libres d'accès (ex.: *IEC 104*).

Exemple de protocole industriel ouvert :

- Modbus
- OPC UA
- Bacnet

Plusieurs implémentations peuvent être trouvées pour chacun de ces protocoles.

# Protocole industriel

## Self Made

Il est courant de trouver un protocole non standard créé pour les besoins d'un automate ou d'un processus métier. Cette implémentation peut être :

- Création d'un client et d'un serveur ;
- Réimplémenter d'un protocole dans un opcode fonction ou dans les échanges de données d'un protocole.

Il est notamment possible de recréer un protocole à l'aide des fonctionnalités offertes par un protocole. Par exemple, Modbus laisse à disposition des opcodes fonction libre ainsi qu'une range d'adresse en lecture/écriture (une zone mémoire).

# Protocole industriel

## Protocole propriétaire et encapsulation (exemple)

```
▼ TPKT, Version: 3, Length: 2494
    Version: 3
    Reserved: 0
    Length: 2494
▼ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
    Length: 2
    PDU Type: DT Data (0x0f)
    [Destination reference: 0x80000]
    .000 0000 = TPDU number: 0x00
    1.... .... = Last data unit: Yes
▼ MMS
    ▼ confirmed-ResponsePDU
        invokeID: 7917
    ▼ confirmedServiceResponse: read (4)
        ▼ read
            ▼ listOfAccessResult: 1 item
                > AccessResult: success (1)
```

# Protocole industriel

## Structure des packets (base)

Message Type	Message Size	RequestID	...	...	Data
--------------	--------------	-----------	-----	-----	------

- **Message Type** : Type of action asked, can be a string or a int ;
- **Message Size** : Size of the message send ;
- **Request ID** : The ID of the exchange, used to handle multiple client ;
- **Data** : Data needed to handle the request.

# Protocole industriel

## Structure des packets (base)

Message Type	Message Size	RequestID	...	...	Data
--------------	--------------	-----------	-----	-----	------

- Message Type : Type of action asked, can be a string or a int ;
  - Message Size : Size of the message send ;
  - Request ID : The ID of the exchange, used to handle multiple client ;
  - Data : Data needed to handle the request.
- 
- Must be fuzzed
  - Manually check
  - Low probability of bug

# Protocole industriel

## Structure des packets (base)

Message Type	Message Size	RequestID	Version	Handshake	...	Data
--------------	--------------	-----------	---------	-----------	-----	------

- Handshake - Checksum : Handshake verification or authentication result.
- Version/Metadata : Build date, date, version of the protocole (ex.: OPC), etc.
- Encapsulation : Protocole in a specific opcode or in data of a protocole.

# Protocole industriel

## Identification des fonctions

```
ew-A Pseudocode-A Hex View-1
default:
    goto LABEL_43;
case 0xC:
    memset(&s, 0, 0x80u);
    memcpy(&s, &buf[x + 2], buf[x + 1]);
    x += buf[x + 1] + 2;
    if (buf[x] == 170)
        x = 0;
    device_mode = 2;
    ackbuf[0] = -86;
    byte_1231D = 11;
    byte_1231E = 1;
    byte_1231F = 2;
    byte_12320 = 85;
    if (sendto(0, ackbuf, 5u, 0, (const sti
        goto LABEL_43;
    puts("amjrd:sendto Failed");
    return 1;
case 0xD:
    v17 = (((((buf[x + 2] << 8) + buf[x + 3]
    x += buf[x + 1] + 2;
    if (buf[x] == 170)
        x = 0;
    goto LABEL_43;
case 0xE:
    memset(&name, 0, 0x14u);
    memcpy(&name, &buf[x + 2], buf[x + 1]);
    x += buf[x + 1] + 2;
```

Il est possible de retrouver cette structure dans les firmwares.

Les éléments caractéristiques peuvent être :

- Les opcodes fonctions ;
- Les chaînes de caractères (ex. : message d'erreur) ;

Il est cependant complexe d'exécuter un firmware en dehors de l'automate.

# Protocole industriel

## Identification des fonctions

default:  
    goto LABEL\_43;  
else if (buf[x] == 0x80)  
{  
    memset(&s, 0, 0x80u);  
    memcpy(&s, &buf[x + 2], buf[x + 1]);  
    if (buf[x] == 170)  
        x = 0;  
    device\_mode = 2;  
    ackbuf[0] = -86;  
    byte\_1231D = 11;  
    byte\_1231E = 1;  
    byte\_1231F = 2;  
    byte\_12320 = 85;  
    if (sendto(0, ackbuf, 5u, 0, (const struct sockaddr\*)&addr))  
        goto LABEL\_43;  
    puts("amgrd:sendto failed");  
    return 1;  
}  
else if (buf[x] == 0x81)  
{  
    u17 = (((((buf[x + 2] << 8) + buf[x + 3]) \* 0x100) + buf[x + 1]) \* 0x100) + buf[x];  
    x += buf[x + 1] + 2;  
    if (buf[x] == 0x80)  
        x = 0;  
    goto LABEL\_43;  
}  
memset(&name, 0, 0x14u);  
memcpy(&name, &buf[x + 2], buf[x + 1]);  
if (buf[x] == 0x80)

## Function implementation

Il est possible de retrouver cette structure dans les firmwares.

Les éléments caractéristiques peuvent être :

- Les opcodes fonctions ;
  - Les chaînes de caractères (ex. : message d'erreur).

La gestion de la réception du message ou des données est une bonne cible pour les fuzzers, générant des erreurs.

# Cas d'étude



## Cas d'étude – OPC UA

---

The background of the slide features a blurred image of industrial equipment, likely a control panel or a network of pipes, with various colored lights (red, blue, green) visible. Overlaid on the bottom left is the OPC UA logo, which consists of the letters "OPC UA" in a bold, black, sans-serif font. The letter "O" is preceded by a decorative graphic of blue diamonds and squares.

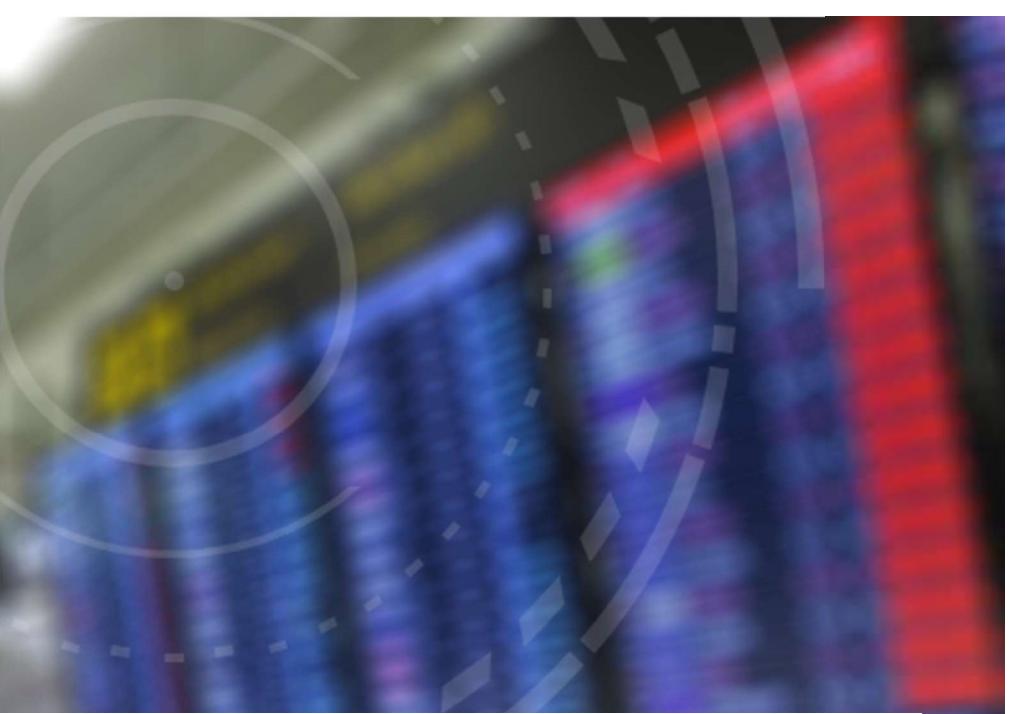
OPC UA est un protocole développé pour remplacer OPC DA (basé sur DCOM). Il est utilisé pour la communication entre un serveur SCADA et les différents équipements.

Versions disponibles :

- V1.00 : 2006 ;
- V1.01 : 2009 ;
- V1.02 : 2013 ;
- V1.03 : 2015 ;
- V1.04 : 2018 ;
- V1.05 : 2020.

## Cas d'étude – OPC UA

---

A blurred background image showing industrial equipment, specifically a stack of colorful cylindrical components, likely heat exchangers or pipes, with circular motion blur patterns.

OPC UA est un protocole développé pour remplacer OPC DA (basé sur DCOM). Il est utilisé pour la communication entre un serveur SCADA et les différents équipements.

19 implémentations différentes en open source, dans plusieurs langages et plateformes.

Fonctionnalité supportée :

- - Authentification (user/passwd) ;
- - Signature des messages ;
- - Chiffrement ;
- - RBAC ;

# Cas d'étude OPC-UA

## Gestion des messages

Trois types de messages en OPC UA :

- **OPN** : Ouverture d'un flux ;
- **MSG** : Message de lecture/écriture/etc. ;
- **CLO** : Fermeture d'un flux.

```
if(OpcUa_MemCmp(abyBuffer,"MSG",3) == 0)
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_DEBUG, "SecureStream - CheckInputHeaderType - Common Service\n");
    *a_pInputType = OpcUa_SecureMessageType_SM;
}
else if(OpcUa_MemCmp(abyBuffer,"OPN",3) == 0)
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_DEBUG, "SecureStream - CheckInputHeaderType - OpenSecureChannel Service\n");
    *a_pInputType = OpcUa_SecureMessageType_SO;
}
else if(OpcUa_MemCmp(abyBuffer,"CLO",3) == 0)
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_DEBUG, "SecureStream - CheckInputHeaderType - CloseSecureChannel Service\n");
    *a_pInputType = OpcUa_SecureMessageType_SC;
}
else
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_DEBUG, "SecureStream - CheckInputHeaderType - Unknown Service\n");
    uStatus = OpcUa_Bad;
}

/* OpenSecureChannel */
case OpcUa_SecureMessageType_SO:
{
    uStatus = OpcUa_SecureListener_ProcessOpenSecureChannelRequest(
        a_pSecureListenerInterface,
        a_hTransportConnection,
        a_ppTransportIstrm,
        a_bRequestComplete);
    break;
}

/* CloseSecureChannel */
case OpcUa_SecureMessageType_SC:
{
    uStatus = OpcUa_SecureListener_ProcessCloseSecureChannelRequest(
        a_pSecureListenerInterface,
        a_hTransportConnection,
        a_ppTransportIstrm,
        a_bRequestComplete);
    break;
}

/* SecureMessage - standard protocol message */
case OpcUa_SecureMessageType_SM:
{
    uStatus = OpcUa_SecureListener_ProcessSessionCallRequest(
        a_pSecureListenerInterface,
        a_hTransportConnection,
        a_ppTransportIstrm,
        a_bRequestComplete);
    break;
}

/* Undefined - Error Status */
default:
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_WARNING, "ProcessRequest: Invalid message header detected!\n");
    OpcUa_GotoErrorWithStatus(OpcUa_Bad);
}
}
```

# Cas d'étude OPC-UA

## Gestion des messages

Trois types de messages en OPC UA :

- **OPN** : Ouverture d'un flux ;
- **MSG** : Message de lecture/écriture/etc. ;
- **CLO** : Fermeture d'un flux.

Choix du traitement : Ne pas fuzzer

Traitement : Fuzzer (memory manipulation)



```
if(OpcUa_MemCmp(abyBuffer,"MSG",3) == 0)
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_DEBUG, "SecureStream - CheckInputHeaderType - Common Service\n");
    *a_pInputType = OpcUa_SecureMessageType_SM;
}
else if(OpcUa_MemCmp(abyBuffer,"OPN",3) == 0)
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_DEBUG, "SecureStream - CheckInputHeaderType - OpenSecureChannel Service\n");
    *a_pInputType = OpcUa_SecureMessageType_SO;
}
else if(OpcUa_MemCmp(abyBuffer,"CLO",3) == 0)
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_DEBUG, "SecureStream - CheckInputHeaderType - CloseSecureChannel Service\n");
    *a_pInputType = OpcUa_SecureMessageType_SC;
}
else
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_DEBUG, "SecureStream - CheckInputHeaderType - Unknown Service\n");
    uStatus = OpcUa_Bad;
}

/* OpenSecureChannel */
case OpcUa_SecureMessageType_SO:
{
    uStatus = OpcUa_SecureListener_ProcessOpenSecureChannelRequest(
        a_pSecureListenerInterface,
        a_hTransportConnection,
        a_ppTransportIstrm,
        a_bRequestComplete);
    break;
}
/* CloseSecureChannel */
case OpcUa_SecureMessageType_SC:
{
    uStatus = OpcUa_SecureListener_ProcessCloseSecureChannelRequest(
        a_pSecureListenerInterface,
        a_hTransportConnection,
        a_ppTransportIstrm,
        a_bRequestComplete);
    break;
}
/* SecureMessage - standard protocol message */
case OpcUa_SecureMessageType_SM:
{
    uStatus = OpcUa_SecureListener_ProcessSessionCallRequest(
        a_pSecureListenerInterface,
        a_hTransportConnection,
        a_ppTransportIstrm,
        a_bRequestComplete);
    break;
}

/* Undefined - Error Status */
default:
{
    OpcUa_Trace(OPCUA_TRACE_LEVEL_WARNING, "ProcessRequest: Invalid message header detected!\n");
    OpcUa_GotoErrorWithStatus(OpcUa_Bad);
}
```

# Cas d'étude OPC-UA

## Gestion de la mémoire - Ecriture

La réimplémentassions de fonction basique d'écriture/lecture est courante.

Il permet de s'adapter sur une grande variété de plateformes.

```
=====
 * OpcUa_MemoryStream_Write
=====
static
OpcUa_StatusCode OpcUa_MemoryStream_Write(
    OpcUa_OutputStream* ostrm,
    OpcUa_Byte*        buffer,
    OpcUa_UInt32        count)
{
    OpcUa_MemoryStream* handle = OpcUa_Null;
    OpcUa_DeclareErrorTraceModule(OpcUa_Module_MemoryStream);
    OpcUa_ReturnErrorIfArgumentNull(ostrm);
    OpcUa_ReturnErrorIfArgumentNull(buffer);
    OpcUa_ReturnErrorIfInvalidStream(ostrm, Write);

    handle = (OpcUa_MemoryStream*)ostrm->Handle;

    if (handle->Closed)
    {
        return OpcUa_BadInvalidState;
    }

    return OpcUa_Buffer_Write(handle->pBuffer, buffer, count);
}
```

No security check ☺

# Cas d'étude OPC-UA

## Gestion de la mémoire - Ecriture

La réimplémentation de fonction basique d'écriture/lecture est courante.

Il permet de s'adapter sur une grande variété de plateformes.

Les réimplémentations des fonctions d'écriture sont dangereuses sans vérification de la taille ou de la localisation de l'écriture.

```
OpcUa_StatusCode OpcUa_Buffer_Write(  
    OpcUa_Handle a_Handle,  
    OpcUa_Byte* a_pData,  
    OpcUa_UInt32 a_uCount)  
{  
    OpcUa_UInt32 bytesAvailable = 0;  
    OpcUa_Buffer* buffer = OpcUa_Null;  
    OpcUa_StatusCode uStatus = OpcUa_Good;  
  
    OpcUa_DeclareErrorTraceModule(OpcUa_Module_Buffer);  
  
    OpcUa_ReturnErrorIfArgumentNull(a_Handle);  
    OpcUa_ReturnErrorIfArgumentNull(a_pData);  
  
    OpcUa_ReturnErrorIfInvalidBuffer();  
    buffer = (OpcUa_Buffer*)a_Handle; Security check 😊  
  
    bytesAvailable = buffer->Size - buffer->Position;  
    if(bytesAvailable < a_uCount)  
    {  
        /* Try to prevent to get into this codepath by setting the b  
        OpcUa_Byte* newData = OpcUa_Null;  
        OpcUa_UInt32 newSize = buffer->Size;  
  
        newSize += a_uCount - bytesAvailable;
```

# Cas d'étude OPC-UA

## Gestion de la mémoire - Lecture

La réimplémentation des fonctions de lecture est également sensible.

La lecture d'une zone arbitraire peut engendrer, par exemple, une divulgation d'information.

```
=====
* OpcUa_MemoryStream_Read
=====
static
OpcUa_StatusCode OpcUa_MemoryStream_Read(
    OpcUa_InputStream*          istrm,
    OpcUa_Byte*                 buffer,
    OpcUa_UInt32*                count)
{
    OpcUa_MemoryStream* handle = OpcUa_Null;

    OpcUa_DeclareErrorTraceModule(OpcUa_Module_MemoryStream);

    OpcUa_ReturnErrorIfArgumentNull(istrm);
    OpcUa_ReturnErrorIfArgumentNull(buffer);
    OpcUa_ReturnErrorIfArgumentNull(count);
    OpcUa_ReturnErrorIfInvalidStream(istrm, Read);

    handle = (OpcUa_MemoryStream*)istrm->Handle;

    if (handle->Closed)
    {
        return OpcUa_BadInvalidState;
    }

    return OpcUa_Buffer_Read(handle->pBuffer, buffer, count);
}
```

No security check ☺

# Cas d'étude OPC-UA

## Gestion de la mémoire - Lecture

La réimplémentation des fonctions de lecture est également sensible.

La lecture d'une zone arbitraire peut engendrer, par exemple, une divulgation d'information.

Les vérifications nécessaires sont :

- Vérification de la taille de lecture ;
- Vérification de l'adresse de lecture ;
- Vérification de l'adresse de retour.

```
OPCUA_EXPORT OpcUa_StatusCode OpcUa_Buffer_Read(
    OpcUa_Handle a_Handle,
    OpcUa_Byte* data,
    OpcUa_UInt32* count)
{
    OpcUa_UInt32 bytesToRead = 0;
    OpcUa_Buffer* buffer = OpcUa_Null;
    OpcUa_StatusCode uStatus = OpcUa_Good;

    OpcUa_DeclareErrorTraceModule(OpcUa_Module_Buffer);

    OpcUa_ReturnErrorIfArgumentNull(a_Handle);
    OpcUa_ReturnErrorIfArgumentNull(data);
    OpcUa_ReturnErrorIfArgumentNull(count);

    OpcUa_ReturnErrorIfInvalidBuffer();

    buffer = (OpcUa_Buffer*)a_Handle;

    if (*count == 0)
    {
        return OpcUa_Good;
    }

    bytesToRead = buffer->EndOfData - buffer->Position;

    if (bytesToRead == 0)
    {
        *count = 0;
        return OpcUa_BadEndOfStream;
    }

    if (bytesToRead > *count)
    {
        bytesToRead = *count;
    }

    OpcUa_MemCpy(data, *count, buffer->Data+buffer->Position, bytesToRead);
    buffer->Position += bytesToRead;

    *count = bytesToRead;

    return uStatus;
}
```

Security check

Security check

Security check

# Cas d'étude OPC-UA

## Gestion de la mémoire - Other

La reiméplmentation des fonctions de manipulation de type doit être sujette à la même attention (ex. : `stcncpy`).

Elles sont plus compliquées à gérer, car plus varié et manipulant plusieurs types de données :

- Entier ;
- Pointeurs ;
- Des structures ;
- etc.

```
=====
 * Copy string.
=====
OpcUa_StatusCode OpcUa_String_StrnCpy( /* bi */ OpcUa_String*      a_pDestString,
                                         /* in */ const OpcUa_String* a_pSrcString,
                                         /* in */ OpcUa_UInt32        a_uLength)
{
    OpcUa_StringA          strRawSrc = OpcUa_Null;
    OpcUa_UInt32            uiSrcLen = 0;
    OpcUa_StringInternal*  pStringInt = (OpcUa_StringInternal*)a_pSrcString;
    OpcUa_StatusCode        uStatus   = OpcUa_Good;

    OpcUa_ReturnErrorIfArgumentNull(a_pDestString);

    /* check src string */
    OpcUa_String_Clear(a_pDestString);

    if(OpcUa_String_IsNull(a_pSrcString))
    {
        return OpcUa_Good;
    }

    strRawSrc = _OpcUa_String_GetRawString(a_pSrcString);

    if(a_uLength == OPCUA_STRING_LENDONTARE)
    {
        /* size in bytes */
        uiSrcLen = OpcUa_String_StrSize(a_pSrcString);
    }
    else
    {
        /* min of given maximum number of bytes and the real length */
        uiSrcLen = (a_uLength < pStringInt->uLength)?a_uLength:pStringInt->uLength;
    }

    uStatus = OpcUa_String_AttachToString( strRawSrc,
                                            uiSrcLen,
                                            0,
                                            OpcUa_True,
                                            OpcUa_True, /* since we copy, this is
                                            a_pDestString);

    return uStatus;
}
```

Euh... 😊

# Cas d'étude OPC-UA

## Analyse réseau

▼ OpcUa Binary Protocol

- Message Type: MSG
- Chunk Type: F
- Message Size: 98
- SecureChannelId: 17
- Security Token Id: 1
- Security Sequence Number: 4
- Security RequestId: 4

▼ OpcUa Service : Encodeable Object

- ▼ TypeId : ExpandedNodeId
  - NodeId EncodingMask: Four byte encoded Numeric (0x01)
  - NodeId Namespace Index: 0
  - NodeId Identifier Numeric: BrowseRequest (527)
- ▼ BrowseRequest
  - > RequestHeader: RequestHeader
  - > View: ViewDescription
  - RequestedMaxReferencesPerNode: 0
  - > NodesToBrowse: Array of BrowseDescription

Il est compliqué de fuzzer un PLC lié à l'absence de contrôle sur le système (absence de Shell sur le système, performance de l'équipement, etc.)

Il est plus simple de fuzzer via le réseau. Les comportements permettant de détecter les erreurs d'exécution/logiques sont :

- temps d'exécution ;
- Redémarrage de l'automate ;
- Code réponse ;
- etc.



Back to basic



```
38.     {
39.         config.CertificateValidator.CertificateValidation += (s, e) => { e.Accept = (e.Error.StatusCode ==
StatusCodes.BadCertificateUntrusted); };
40.     }
41.
42.     ApplicationInstance application = new ApplicationInstance
43.     {
44.         ApplicationName = "My Console Client",
45.         ApplicationType = ApplicationType.Client,
46.         ApplicationConfiguration = config
47.     };
48.     application.CheckApplicationInstanceCertificate(false, 2048).GetAwaiter().GetResult();
49.
50.     var selectedEndpoint = CoreClientUtils.SelectEndpoint("opc.tcp://minint-n3864sr:62548/Quickstarts/DataAccessServer", useSecurity:
true, operationTimeout: 15000);
51.
52.     //Create session with your server
53.     using (Session session = Session.Create(config, new ConfiguredEndpoint(null, selectedEndpoint,
EndpointConfiguration.Create(config)), false, "", 60000, null, null).GetAwaiter().GetResult())
54.     {
55.         //Browse the server namespace
56.         ReferenceDescriptionCollection refs;
57.         Byte[] cp;
58.         session.Browse(null, null, ObjectIds.ObjectsFolder, 0u, BrowseDirection.Forward, ReferenceTypeIds.HierarchicalReferences,
true, (uint)NodeClass.Object | (uint)NodeClass.Method, out cp, out refs);
59.         Console.WriteLine("DisplayName: BrowseName, NodeClass");
60.         foreach (var rd in refs)
61.         {
62.             if (rd.BrowseName != null && rd.NodeClass != null)
63.             {
64.                 Console.WriteLine("BrowseName: {0}, NodeClass: {1}", rd.BrowseName, rd.NodeClass);
65.             }
66.         }
67.     }
68. }
```



```
21.  
22.     /* Listing endpoints */  
23.  
24.     UA_EndpointDescription* endpointArray = NULL;  
25.     size_t endpointArraySize = 0;  
26.     UA_StatusCode retval = UA_Client_getEndpoints(client, "opc.tcp://localhost:16664",  
27.                                                 &endpointArraySize, &endpointArray);  
28.  
29.     if(retval != UA_STATUSCODE_GOOD) {  
30.         UA_Array_delete(endpointArray, endpointArraySize, &UA_TYPES[UA_TYPES_ENDPOINTDESCRIPTION]);  
31.         UA_Client_delete(client);  
32.         return (int)retval;  
33.     }  
34.     printf("%i endpoints found\n", (int)endpointArraySize);  
35.     for(size_t i=0;i<endpointArraySize;i++){  
36.         printf("URL of endpoint %i is %.*s\n", (int)i,  
37.                (int)endpointArray[i].endpointUrl.length,  
38.                endpointArray[i].endpointUrl.data);  
39.     }  
40.     UA_Array_delete(endpointArray,endpointArraySize, &UA_TYPES[UA_TYPES_ENDPOINTDESCRIPTION]);  
41.  
42.     /* Connect to a server */  
43.     /* anonymous connect would be: retval = UA_Client_connect(client, "opc.tcp://localhost:16664"); */  
44.     retval = UA_Client_connect_username(client, "opc.tcp://localhost:16664", "user1", "password");  
45.     if(retval != UA_STATUSCODE_GOOD) {  
46.         UA_Client_delete(client);  
47.         return (int)retval;  
48.     }
```

```
1. import sys,time
2. sys.path.insert(0, "..")
3. from opcua import Client
4.
5. if __name__ == "__main__":
6.     client = Client("opc.tcp://140.123.122.129:4840")
7.     try:
8.         res = client.connect()
9.         root = client.get_root_node()
10.        print("Objects node is: ", root)
11.        for child1 in root.get_children():
12.            for child2 in child1.get_children():
13.                for child3 in child2.get_children():
14.                    for child4 in child3.get_children():
15.                        print(child4)
16.        var = client.get_node("StringNodeId(ns=2;s=/Hmi/hmi_chsd_changed)")
17.        print(var.get_properties())
18.        # time.sleep(2)
19.
20.    finally:
21.        client.disconnect()
```

# Questions?