Industroyer2: Nozomi Networks Labs Analyzes the IEC 104 Payload

nozominetworks.com/blog/industroyer2-nozomi-networks-labs-analyzes-the-iec-104-payload/

By April 27, 2022

In a previous blog we gave a high-level overview of Industroyer2, the latest tool that advanced persistent threat (APT) group Sandworm used to target the Ukrainian power grid. As the name coined by ESET suggests, this sample presents many commonalities with its predecessor Industroyer, which was used in the notorious December 2016 attack on Ukraine's grid. In this blog, we will compare the two samples by providing a side-by-side analysis to uncover the similarities, which helps us gain more insight into the mind of the attacker.
Before we present evidence that will link the two samples, we will first focus on the main task of Industroyer2, namely manipulating IEC-104 Information Object Address (IOA).

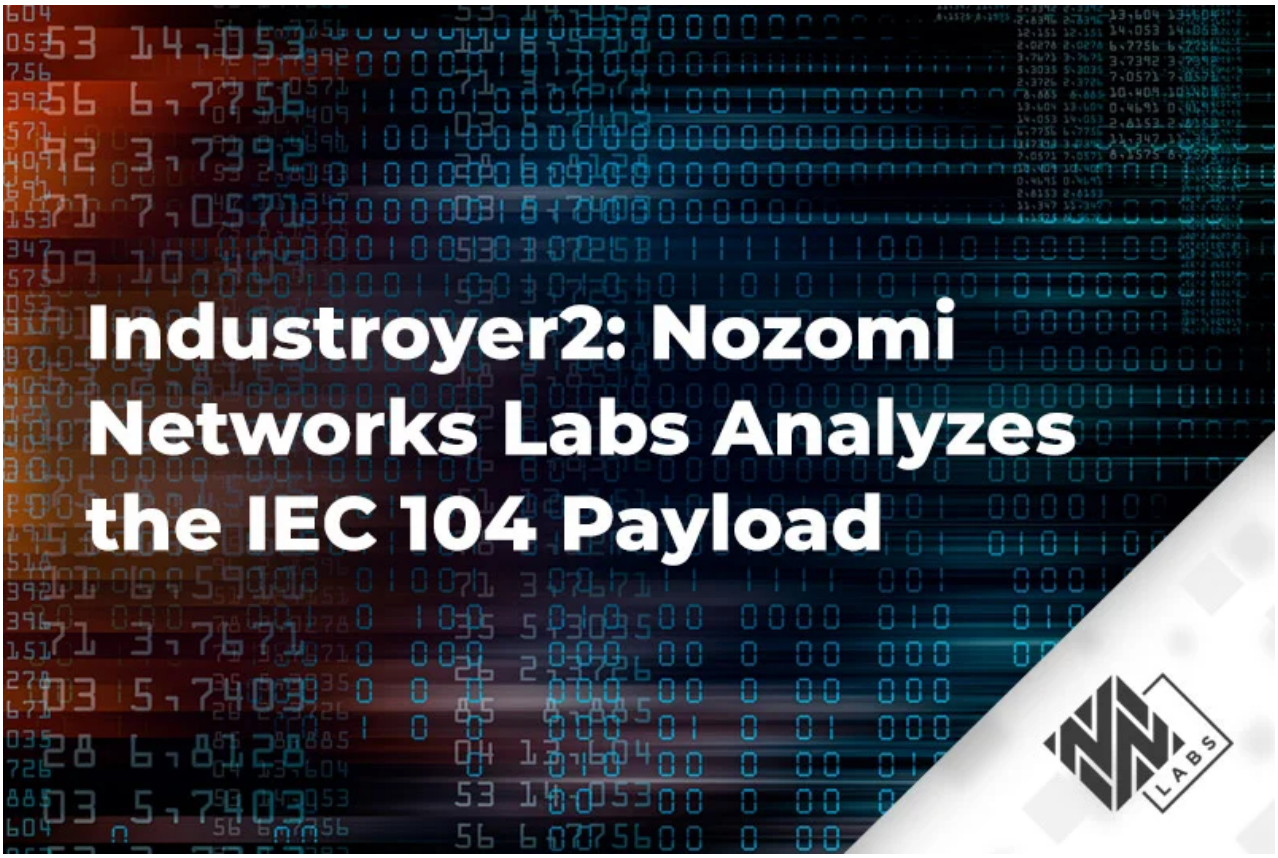
Unlike Industroyer, which targeted IEC-101, IEC-104, IEC 61850 and OPC Data Access mostly through different Dynamic-link libraries (DLLs) (i.e.101.dll, 104.dll, 61850.dll, OPC.exe and OPCClientDemo.dll), the malware dubbed Industroyer2 is a standalone executable which exclusively targets IEC-104.

During the analysis of the sample, we came across something unusual in modern malware: the authors did not bother hiding its activity, nor perform any form of obfuscation. The core of the malware consists of its configuration which, among other parameters described below, contains a hardcoded list of IOAs to manipulate. This configuration is not protected in the executable, rather it is embedded as a regular Unicode string.

This lack of concern for detection on the endpoint suggests that the threat actor had a fairly complete understanding of the security measures deployed in the target environment. At the same time, the hardcoded list of IOAs indicates two things:

1. The operators had a thorough understanding of the Operational Technology (OT) environment; and
2. The Industroyer2 sample is designed to be executed in a privileged environment with direct access to the target devices.

The window between initial access and when Industroyer2 was launched is unknown. However, we can assume that the window between access and attack was within days rather than hours, based on the malicious activity.



Nozomi Networks Labs' analysis of Industroyer2 suggests that the threat actors had a fairly complete understanding of the security measures deployed in the target environment.

IEC-104 Configuration

The sample that we analyzed contains hardcoded configurations for three different stations. For each station the configuration includes:

1. Station Configuration Header: Header to configure the station interaction
2. IOA Configuration Format: Table containing multiple IOAs and their corresponding parameters

Station Configuration Header

The configuration for one of the stations uses the following header:

10.x.y.z 2404 3 0 1 1 PService_PPD.exe 1 "D:\OIK\DevCounter" 0 1 0 0 1 0 0 44

10.x.y.z → Controlled station local IP address

2404 → Controlled station port

3 → ASDU address

0 → Operation mode. If set to 0, the hardcoded table of IOA is used. If set to 1, two arguments need to follow specifying a start and an end IOA number, which are then used to set a range of IOA to iterate through.

1 → Switch that requires 9 arguments:

a) 1 → Boolean, accepted values: 0, 1

b) PService_PPD.exe → Name of the process to be killed

c) 1 → Controls if the filename rename should happen, accepted values: 0 (skip rename), 1 (rename)

d) “D:\OIK\DevCounter” → Folder where the executable targeted for killing and rename is stored

e) 0 → Sleep time in minutes, executed prior to initiating interaction with the station

f) 1 → Sleep time in seconds

g) 0 → Sleep time in seconds

h) 0 → Sleep time in seconds

i) 1 → Boolean, accepted values: 0, 1

0 → If set, it will interact with each IOA again, with SCO/DCO On/Off inverted

44 → Number of IOA following header

IOA Configuration Format

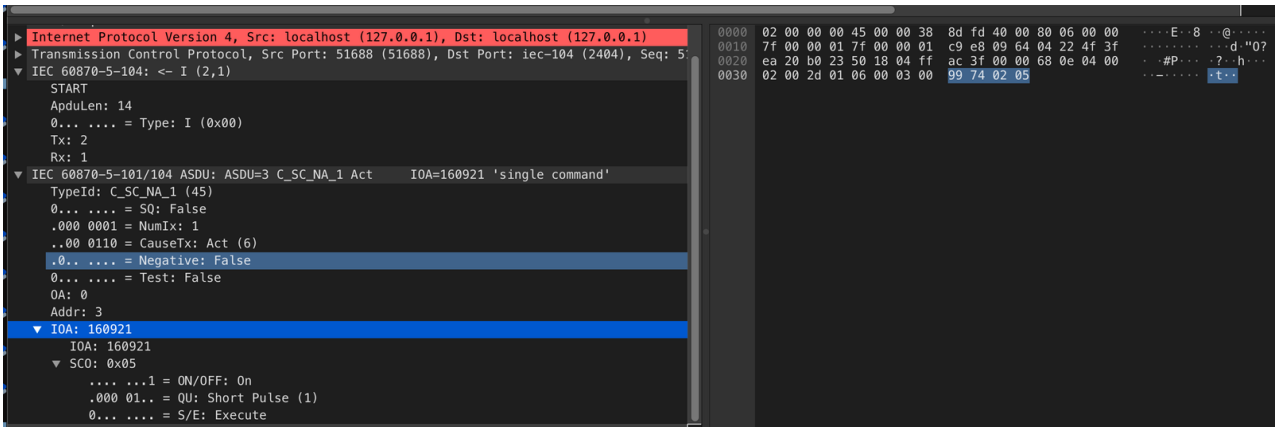


Figure 1. IOA Configuration.

160921 1 0 1 1 2

- 160921 → IOA address
- 1 → Sets single or double command, accepted values: 0 (Double), 1 (Single)
- 0 → Sets SCO/DCO Select/Execute, accepted values: 0 (Execute), 1 (Select)
- 1 → Sets SCO/DCO On/Off, accepted values: 0 (Off), 1 (On)
- 1 → Priority
- 2 → IOA entry index in the configuration list

Operation

iec60870_104 and tcp.dstport == 2404

No.	Time	Protocol	Length	Info	
4	09:57:56.388263	IEC 60870-5-104	50	<- U (TESTFR act)	
8	09:57:57.777530	IEC 60870-5-104	50	<- U (STARTDT act)	
12	09:57:59.168074	IEC 60870-5 ASDU	60	<- I (0,0) ASDU=3 C_IC_NA_1 Act	IOA=0
16	09:58:00.966451	IEC 60870-5-104	50	<- S (1)	
18	09:58:02.152117	IEC 60870-5 ASDU	60	<- I (1,1) ASDU=3 C_SC_NA_1 Act	IOA=130202
22	09:58:03.949065	IEC 60870-5 ASDU	60	<- I (2,1) ASDU=3 C_SC_NA_1 Act	IOA=160921
26	09:58:05.777491	IEC 60870-5 ASDU	60	<- I (3,1) ASDU=3 C_SC_NA_1 Act	IOA=160923
30	09:58:07.574706	IEC 60870-5 ASDU	60	<- I (4,1) ASDU=3 C_SC_NA_1 Act	IOA=160924
34	09:58:09.371291	IEC 60870-5 ASDU	60	<- I (5,1) ASDU=3 C_SC_NA_1 Act	IOA=160925
38	09:58:11.027459	IEC 60870-5 ASDU	60	<- I (6,1) ASDU=3 C_SC_NA_1 Act	IOA=160927
44	09:58:12.684070	IEC 60870-5 ASDU	60	<- I (7,1) ASDU=3 C_SC_NA_1 Act	IOA=160928
48	09:58:14.480261	IEC 60870-5 ASDU	60	<- I (8,1) ASDU=3 C_SC_NA_1 Act	IOA=190202
52	09:58:16.277435	IEC 60870-5 ASDU	60	<- I (9,1) ASDU=3 C_SC_NA_1 Act	IOA=260202
58	09:58:18.075091	IEC 60870-5 ASDU	60	<- I (10,1) ASDU=3 C_SC_NA_1 Act	IOA=260901
62	09:58:19.871360	IEC 60870-5 ASDU	60	<- I (11,1) ASDU=3 C_SC_NA_1 Act	IOA=260902
68	09:58:21.669480	IEC 60870-5 ASDU	60	<- I (12,1) ASDU=3 C_SC_NA_1 Act	IOA=260903
72	09:58:23.465049	IEC 60870-5 ASDU	60	<- I (13,1) ASDU=3 C_SC_NA_1 Act	IOA=260904
76	09:58:25.277649	IEC 60870-5 ASDU	60	<- I (14,1) ASDU=3 C_SC_NA_1 Act	IOA=260905
80	09:58:27.074758	IEC 60870-5 ASDU	60	<- I (15,1) ASDU=3 C_SC_NA_1 Act	IOA=260906
84	09:58:28.870927	IEC 60870-5 ASDU	60	<- I (16,1) ASDU=3 C_SC_NA_1 Act	IOA=260907
90	09:58:30.684995	IEC 60870-5 ASDU	60	<- I (17,1) ASDU=3 C_SC_NA_1 Act	IOA=260908
94	09:58:32.480503	IEC 60870-5 ASDU	60	<- I (18,1) ASDU=3 C_SC_NA_1 Act	IOA=260909
98	09:58:34.277877	IEC 60870-5 ASDU	60	<- I (19,1) ASDU=3 C_SC_NA_1 Act	IOA=260910
102	09:58:36.090148	IEC 60870-5 ASDU	60	<- I (20,1) ASDU=3 C_SC_NA_1 Act	IOA=260911
106	09:58:37.887052	IEC 60870-5 ASDU	60	<- I (21,1) ASDU=3 C_SC_NA_1 Act	IOA=260912
110	09:58:39.684105	IEC 60870-5 ASDU	60	<- I (22,1) ASDU=3 C_SC_NA_1 Act	IOA=260914
114	09:58:41.480858	IEC 60870-5 ASDU	60	<- I (23,1) ASDU=3 C_SC_NA_1 Act	IOA=260915
118	09:58:43.277663	IEC 60870-5 ASDU	60	<- I (24,1) ASDU=3 C_SC_NA_1 Act	IOA=260916
122	09:58:45.074448	IEC 60870-5 ASDU	60	<- I (25,1) ASDU=3 C_SC_NA_1 Act	IOA=260918
130	09:58:46.871366	IEC 60870-5 ASDU	60	<- I (26,1) ASDU=3 C_SC_NA_1 Act	IOA=260920
134	09:58:48.668731	IEC 60870-5 ASDU	60	<- I (27,1) ASDU=3 C_SC_NA_1 Act	IOA=290202
138	09:58:50.464988	IEC 60870-5 ASDU	60	<- I (28,1) ASDU=3 C_SC_NA_1 Act	IOA=338501
142	09:58:52.293361	IEC 60870-5 ASDU	60	<- I (29,1) ASDU=3 C_DC_NA_1 Act	IOA=1401
146	09:58:54.090015	IEC 60870-5 ASDU	60	<- I (30,1) ASDU=3 C_DC_NA_1 Act	IOA=1402
150	09:58:55.886795	IEC 60870-5 ASDU	60	<- I (31,1) ASDU=3 C_DC_NA_1 Act	IOA=1403
154	09:58:57.699431	IEC 60870-5 ASDU	60	<- I (32,1) ASDU=3 C_DC_NA_1 Act	IOA=1404
158	09:58:59.496114	IEC 60870-5 ASDU	60	<- I (33,1) ASDU=3 C_DC_NA_1 Act	IOA=1301
162	09:59:01.292947	IEC 60870-5 ASDU	60	<- I (34,1) ASDU=3 C_DC_NA_1 Act	IOA=1302
166	09:59:03.089872	IEC 60870-5 ASDU	60	<- I (35,1) ASDU=3 C_DC_NA_1 Act	IOA=1303
170	09:59:04.886824	IEC 60870-5 ASDU	60	<- I (36,1) ASDU=3 C_DC_NA_1 Act	IOA=1304
174	09:59:06.684084	IEC 60870-5 ASDU	60	<- I (37,1) ASDU=3 C_DC_NA_1 Act	IOA=1201
178	09:59:08.480310	IEC 60870-5 ASDU	60	<- I (38,1) ASDU=3 C_DC_NA_1 Act	IOA=1202
182	09:59:10.277783	IEC 60870-5 ASDU	60	<- I (39,1) ASDU=3 C_DC_NA_1 Act	IOA=1203
186	09:59:12.074263	IEC 60870-5 ASDU	60	<- I (40,1) ASDU=3 C_DC_NA_1 Act	IOA=1204
190	09:59:13.871518	IEC 60870-5 ASDU	60	<- I (41,1) ASDU=3 C_DC_NA_1 Act	IOA=1101
194	09:59:15.669191	IEC 60870-5 ASDU	60	<- I (42,1) ASDU=3 C_DC_NA_1 Act	IOA=1102
198	09:59:17.465027	IEC 60870-5 ASDU	60	<- I (43,1) ASDU=3 C_DC_NA_1 Act	IOA=1103
206	09:59:19.277313	IEC 60870-5 ASDU	60	<- I (44,1) ASDU=3 C_DC_NA_1 Act	IOA=1104
214	09:59:23.152341	IEC 60870-5-104	50	<- U (STOPDT act)	

Figure 2. IEC 104 Interaction.

After terminating `PServiceControl.exe`, and based on the configuration, `PService_PPD.exe` which is then renamed with `.MZ` appended to its name, the sample begins IEC 104 interaction (Figure 2).
Traffic sent from the malicious sample to the substation is prefixed with MASTER ->> SLV in the sample's debugging output and vice versa. The interaction begins with a sequence of TESTFR frames: `TESTFR act` sent from the malicious sample, which is then acknowledged by a `TESTFR con` frame. TESTFR frames are used to check if there are connectivity problems between two nodes.

The next frame sent to the receiving station is `STARTDT act`, which is a data transfer activation request, expecting back a `STARTDT con` reply as confirmation. Once data transfer is enabled it is followed by interrogation command `(C_IC_NA_1)`.

The next step that the sample performs using its hardcoded station configuration is to iterate through the hardcoded list of IOA per station and send `C_SC_NA_1 or C_DC_NA_1` type frames, depending on each IOA's configuration. Apart from specifying whether an IOA is meant to be used with a single or double command, as detailed in the configuration above, it is possible to specify the bits used to control Single/Double (SCO/DCO), On/Off, and Select/Execute.

Industroyer vs Industroyer2: Side-by-side Analysis

The Industroyer2 sample with `sha256 d69665f56ddef7ad4e71971f06432e59f1510a7194386e5f0e8926aea7b88e00` has an overwhelming number of similarities with the Industroyer sample called 104.dll and `sha256 7907dd95c1d36cf3dc842a1bd804f0db511a0f68f4b3d382c23a3c974a383cad`. This is a strong indication that the same threat actor had access to the source code.

The following screenshot (Figure 3) presents an example of these similarities with a side-by-side comparison of the decompiled main thread of both samples. On the left side we have the Industroyer sample from 2016, while on the right we have the Industroyer2 sample from 2022.

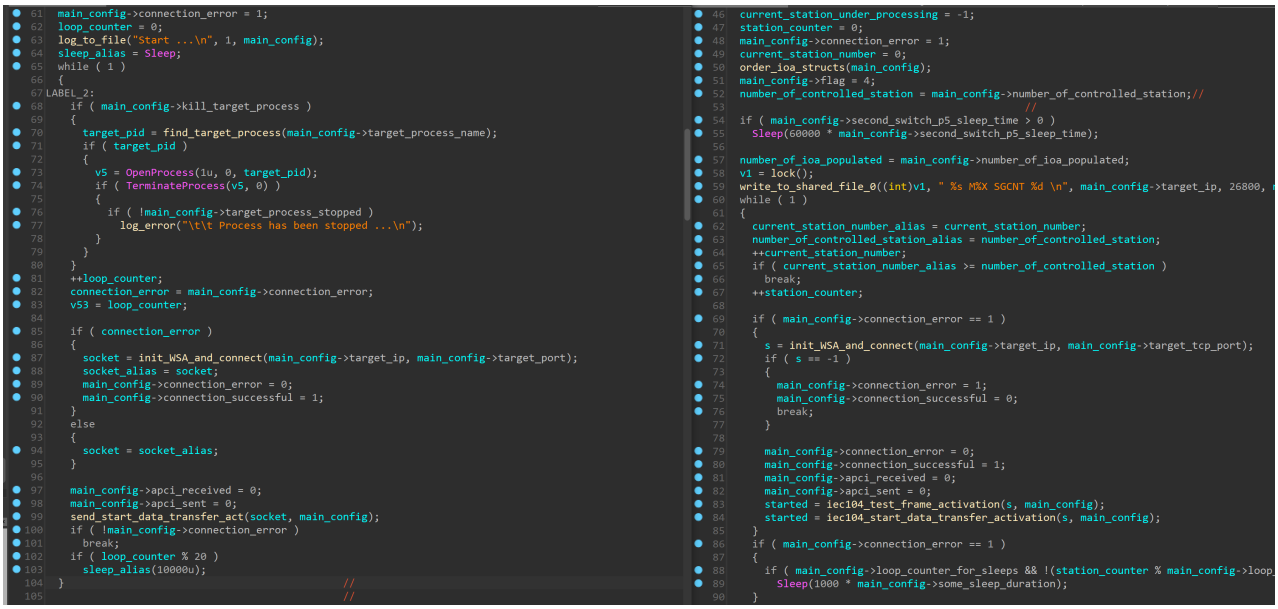


Figure 3. Industroyer (left) and Industroyer2 (right) technical comparison of decompiled main thread of both samples.

The "process killing" functionality that Industroyer implements in the main thread, in Industroyer2 has been factored out into the thread that eventually spawns this main thread. What stands out the most though, is the usage of a structure that we renamed "main_config" in both the samples which stores data used globally throughout the execution. The "main_config" structure as found in Industroyer2 has been updated to hold additional fields, but it shares the same "blueprint" of Industroyer.
The following screenshot (Figure 4) contains the decompiled code responsible for the creation of the `STARTDT` frame. The similarities between the two samples are once again clear, with the "main_config" structure being passed as an argument from one function to

another.

```
int __fastcall send_start_data_transfer_act(SOCKET socket, main_config *main_config)
{
  apci_outer *full_apci_obj; // eax
  apci *apci; // ecx
  int number_of_bytes_recvd; // eax
  char *v7; // ecx
  apci_outer *Block; // [esp+Ch] [ebp-10014h]
  int packet_buffer[16388]; // [esp+10h] [ebp-10010h] BYREF

  Block = operator new(8u);
  full_apci_obj = create_full_apci_obj(Block);
  packet_buffer[0x4003] = 0xFFFFFFFF;
  apci = full_apci_obj->apci;

  *&apci->asdu = 0x30468;
  apci->sent = 0;
  apci->received = 0;
  full_apci_obj->apci->apdu_len = 4;
  full_apci_obj->apci->apci_type = 3;
  full_apci_obj->apci->function_code = 7;
  full_apci_obj->apci->received = 0;
  send_parse_and_log(socket, full_apci_obj, main_config);
  Sleep(dwMilliseconds);
  memset(packet_buffer, 0, 0x10000u);
  number_of_bytes_recvd = outer_recv(main_config, socket, packet_buffer);
  parse_received_packet(packet_buffer, socket, number_of_bytes_recvd, v7, main_config);
  return 0;
}
```

```
int __stdcall iec104_start_data_transfer_activation(SOCKET s, main_config *main_config)
{
  int packet_buffer[16385]; // [esp+0h] [ebp-10014h] BYREF
  int number_of_bytes_recvd; // [esp+10004h] [ebp-10h]
  struct apci_outer *full_apci_obj; // [esp+10008h] [ebp-Ch]
  apci_outer *v6; // [esp+1000Ch] [ebp-8h]
  struct apci **apci; // [esp+10010h] [ebp-4h]

  v6 = (apci_outer *)outer_process_heap_alloc(8u);
  if ( v6 )
    full_apci_obj = create_full_apci_obj(v6);
  else
    full_apci_obj = 0;
  apci = &full_apci_obj->apci;
  set_APCI(full_apci_obj->apci);
  full_apci_obj->apci->apdu_len = 4;
  (*apci)->apci_type = 3;
  (*apci)->function_code = 7;                   // Start Data Transfer Activation
  (*apci)->received = 0;
  send_parse_and_log(s, apci, main_config);
  Sleep(1000u);
  packet_buffer[0x4000] = 4096;
  memset_zero_outer(packet_buffer, 0x10000);
  number_of_bytes_recvd = 0;
  number_of_bytes_recvd = outer_recv(s, main_config, (char *)packet_buffer, 4096);
  parse_recv_and_send(s, (byte *)packet_buffer, number_of_bytes_recvd, " MSTR <<- SLV \t", main_config, 0);
  return 0;
}
```

Figure 4. Industroyer (left) and Industroyer2 (right) technical comparison of decompiled code which creates STARTDT frame.
We can conclude that the threat actor does not aspire to be stealthy and is not concerned about obfuscating the malware activity or the similarities between Industroyer and Industroyer2.

Recommendations

Here are some ways companies can increase their protection:

- **Basic cyber hygiene:** reset passwords, check employee and vendor account/network access and permissions, scan the network for any open ports and close/secure them, etc.
- **Utilize YARA rules** to search for and generate alerts on associated malware activity
- **Use anomaly detection tools** to detect any changes or variations to malware, as well as any irregular activity occurring in OT environments
- **Use an automated firewall** in conjunction with an anomaly detection tool to stop further attack commands
- **Threat hunt for suspicious activity in your network**; this can potentially help to discover attackers early on

We also recommend adhering to CISA's 2017 advisory if those security measures have not been implemented already.

Nozomi Networks will continue to monitor the situation and provide updates on what we are seeing, as well as recommendations the OT industry can use to protect their networks.