OIDA

Scan. Fuzz. Assess.

Responsibly, in OT.

```
$ pip install oida-ics[all]
```
· backed by QuellSec

$whoami

Felix Eberstaller aka f0rw4rd

- **Head of Vulnerability Research / R&D** at Limes Security
- **Lecturer** at JKU: pentesting.
- **Black Hat USA 2025** speaker
- **50+ CVEs**, one **CISA KEV** (CVE-2023-4346, KNX BCU key), one **MITRE ATT&CK for ICS** technique (T0892).
- **Tools**: OIDA, knxunlocker, tls-preloader, sthenos-embedded-toolkit.
- **Obmann of QuellSec**, Austrian non-profit.
- `github.com/f0rw4rd` · `blog.f0rw4rd.xyz`

The problem

OT is the technology behind power, water, manufacturing, healthcare, everything with a cyberphysical impact.

- **Dozens of protocols**, IT protocols, product/vendor-specific, typical OT protocols
- Typical issues
  - Most offensive OT tooling is unsupported, unmaintained, clunky
  - Tried to reimplement full stacks
  - One job takes too many tools and too many command lines
  - Hard to use in nearly air-gapped networks

Fifteen years of scattered tooling

- 2008 - 2011: Stuxnet, nmap, modscan, msf
- 2012 - 2014: Project Basecamp, Conpot, Redpoint, many scripts
- 2014 - 2016: larger frameworks e.g. GRASSMARLIN
- 2017 - 2019: ISF, icsmaster, icssploit, all fin ack :-/

The OT/ICS Dynamic Assessment (OIDA) framework

- **One binary**, uniform syntax, written in python3.
- **25+ industrial protocols** under one CLI.
- Syntax is always `oida <protocol> <target>`.
- **Discover, enumerate, fuzz and analyse**.
- Runs **offline and air-gapped**, no agent on the device.
- Also supported on Windows (most features)
- **One install**: `pip install oida-ics[all]` or `oida-ics[modbus]`

00 · Shooting Range

99+ Services to emulate OT/ICS Services and OT CVEs

- **Based on docker** and public hosted
- **Test the tools and your skills**: e.g. can you crack an MMS password
- Used to ensure that OIDA works (part of CI/CD)

```
f0rw4rd@devlocal ~/d/oida> ./services.py list modbus
=== Core Services ===  python services.py up
  GROUP            SERVICE                        PORT(S)                          DESCRIPTION
  ---              ---                            ---                              ---
  modbus           modbus-conpot                  5503                             Conpot Modbus TCP honeypot, Schneider M34
                   modbus-mock                    502                              Modbus TCP server, unit IDs 1-10, coils/r
                   modbus-rtu-tcp                 5030                             Modbus RTU-over-TCP gateway (3 slave unit
                   modbus-sunspec                 5502                             SunSpec Modbus solar inverter simulator,
                   modbus-tls                     802                              Modbus TLS with mTLS client auth

=== CVE Services ===  python services.py up cve
  GROUP            SERVICE                        PORT(S)       CVE                DESCRIPTION
  ---              ---                            ---           ---                ---
  modbus           modbus-cve-2019-14462          5024          CVE-2019-14462     libmodbus out-of-bounds read
                   modbus-cve-2019-14463          5025          CVE-2019-14463     libmodbus out-of-bounds read (write path)
                   modbus-cve-2022-0367           5021          CVE-2022-0367      libmodbus integer overflow
                   modbus-cve-2024-10918          5022          CVE-2024-10918     libmodbus heap buffer overflow
```

01 · Discover

Discovery, optimised for Layer 2

- **Passive**: `oida discovery eth0` listens promiscuously (LLDP, DCP, mDNS, CDP). Nothing transmitted.
- **Active**: `-a` adds active OT probes + ARP + ICMP
  - PROFINET DCP, CodeSys, BACnet, KNX, …
- **PCAP**: `oida pcap` gives the same inventory from a capture.

02 · Analyze

PCAP, offline analysis

Feed a capture, get infos quick!

- **109 listeners**, six categories, over a PyShark/tshark pipeline
- Per-protocol depth: which commands hit which IOAs, any writes attempted
- A per-protocol breakdown, as json easy to process

A capture from a real attack

`oida pcap` on an **Industroyer2-style** IEC 104 capture (2022 Ukraine grid, breakers tripped over IEC 104).

```
(.venv) f0rw4rd@dev2 ~/p/oida> oida pcap tests/fixtures/pcap/iec104/eset_industroyer2_sample1.pcap | grep rw=
PCAP     [*] | 0 | IEC104   | 10.9.9.9 | 10.1.1.1 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=activation common_addr=3 ioa=0 value=QOI=20  |
PCAP     [*] | 0 | IEC104   | 10.1.1.1 | 10.9.9.9 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=actcon common_addr=3 ioa=0 value=QOI=20      |
PCAP     [*] | 1 | IEC104   | 10.9.9.9 | 10.1.1.2 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=activation common_addr=2 ioa=0 value=QOI=20  |
PCAP     [*] | 1 | IEC104   | 10.1.1.2 | 10.9.9.9 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=actcon common_addr=2 ioa=0 value=QOI=20      |
PCAP     [*] | 0 | IEC104   | 10.1.1.1 | 10.9.9.9 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=actterm common_addr=3 ioa=0 value=QOI=20     |
PCAP     [*] | 2 | IEC104   | 10.9.9.9 | 10.1.1.3 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=activation common_addr=1 ioa=0 value=QOI=20  |
PCAP     [*] | 2 | IEC104   | 10.1.1.3 | 10.9.9.9 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=actcon common_addr=1 ioa=0 value=QOI=20      |
PCAP     [*] | 1 | IEC104   | 10.1.1.2 | 10.9.9.9 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=actterm common_addr=2 ioa=0 value=QOI=20     |
PCAP     [*] | 2 | IEC104   | 10.1.1.3 | 10.9.9.9 | C_IC_NA_1     | rw=read type_id=100 (C_IC_NA_1) cot=actterm common_addr=1 ioa=0 value=QOI=20     |
PCAP     [*] | 0 | IEC104   | 10.9.9.9 | 10.1.1.1 | C_SC_NA_1     | rw=write type_id=45 (C_SC_NA_1) cot=activation common_addr=3 ioa=130202 value=ON |
PCAP     [*] | 0 | IEC104   | 10.1.1.1 | 10.9.9.9 | C_SC_NA_1     | rw=write type_id=45 (C_SC_NA_1) cot=actcon common_addr=3 ioa=130202 value=ON     |
PCAP     [*] | 2 | IEC104   | 10.9.9.9 | 10.1.1.3 | C_DC_NA_1     | rw=write type_id=46 (C_DC_NA_1) cot=activation common_addr=1 ioa=1258 value=OFF  |
PCAP     [*] | 1 | IEC104   | 10.9.9.9 | 10.1.1.2 | C_DC_NA_1     | rw=write type_id=46 (C_DC_NA_1) cot=activation common_addr=2 ioa=1104 value=OFF  |
PCAP     [*] | 1 | IEC104   | 10.1.1.2 | 10.9.9.9 | C_DC_NA_1     | rw=write type_id=46 (C_DC_NA_1) cot=actcon common_addr=2 ioa=1104 value=OFF      |
```

> Not just "IEC 104 seen": which command hit which IOA, and whether it was control.

Credentials, straight from a capture

`oida pcap` to parse for credenials:

```
PCAP  [*] | 0 | SMB      | 10.160.65.192   | 192.168.100.168 | SMB3 SessionSetup    | version=SMB3                                                          |
PCAP  [*] | 0 | NETBIOS  | 192.168.100.168 | 10.160.65.192   | NBSS Session Message | type=Session Message len=174                                          |
PCAP  [*] | 0 | SMB      | 192.168.100.168 | 10.160.65.192   | SMB3 SessionSetup    | version=SMB3                                                          |
PCAP  [*] | 0 | NETBIOS  | 10.160.65.192   | 192.168.100.168 | NBSS Session Message | type=Session Message len=132                                          |
PCAP  [*] | 0 | SMB      | 10.160.65.192   | 192.168.100.168 | SMB3 SessionSetup    | version=SMB3                                                          |
PCAP  [*] +---+----------+-----------------+-----------------+----------------------+-----------------------------------------------------------------------+
PCAP  [*] Credentials (1)
PCAP  [*] +----------+--------+--------------------+---------------+-----------------+
PCAP  [*] | Protocol | Type   | Username           | Server        | Client          |
PCAP  [*] +----------+--------+--------------------+---------------+-----------------+
PCAP  [*] | NTLMSSP  | NTLMv2 | NULL\administrator | 10.160.65.192 | 192.168.100.168 |
PCAP  [*] +----------+--------+--------------------+---------------+-----------------+
PCAP  [+] netbios: 2 devices
PCAP  [+] ntlm: 2 devices
PCAP  [+] ntlm hash (NTLMv2): administrator @ 10.160.65.192 [administrator::NULL:26de2c0b3abaaa1c:711d6cb05614bc240ca7e2a38568ff85:0101000000000000e652e41aa7b4d401dac
00060046004f004f000100100044004600530052004f004f00540031000400100066006f006f002e0074006500730074000300220064006600730072006f006f00740031002e0066006f006f002e0074006500
06f002e00740065007300740007000800e652e41aa7b4d40100000000]
PCAP  [+] smb: 1 devices
```

> Plaintext lifted directly; hashes written ready to crack.

03 · Protocols

```
oida <protocol> <target>
```

Modbus, why enumeration is hard

Modbus looks simple. The real-world surface is wider.

- **RTU-via-TCP**: gateways tunnel RTU frames over 502; OIDA probes both framings.
- **Custom FCs**: vendors define codes in 65-72 and 100-110; the only signal is exception 01 versus a response.
- **Encoding**: 32-bit values arrive big-, little- or word-swapped; all three decoded.
- **File transfer** and value mappings: read files off a device, learn what Coil 3 means.
- One IP can be 247 devices

Modbus, live against the mock

```
(.venv) f0rw4rd@dev2 ~/p/oida> oida modbus 127.0.0.1 -p 5502 --scan-fc
[+] OIDA v0.9.9 | powered by @f0rw4rd
MODBUS 127.0.0.1:5502  [*] Connecting to 127.0.0.1:5502
MODBUS 127.0.0.1:5502  [+] Connected via TCP
MODBUS 127.0.0.1:5502  [!] No encryption - Modbus/TCP has no transport encryption (cleartext)
MODBUS 127.0.0.1:5502  [*]     Unit ID: 1
MODBUS 127.0.0.1:5502  [*]     Server ID: OIDA Test Devices-SUNSPEC-MOCK-10KW-1.0.0 (Running)
MODBUS 127.0.0.1:5502  [*] Scanning function codes (1-8,11,12,15-17,20-23,43)...
[*] Progress: 18/18 (100.0%)
MODBUS 127.0.0.1:5502  [*] [Supported Function Codes] (10 found)
MODBUS 127.0.0.1:5502  [*]   FC   1: Read Coils
MODBUS 127.0.0.1:5502  [*]   FC   2: Read Discrete Inputs
MODBUS 127.0.0.1:5502  [*]   FC   3: Read Holding Registers
MODBUS 127.0.0.1:5502  [*]   FC   4: Read Input Registers
MODBUS 127.0.0.1:5502  [*]   FC   7: Read Exception Status
MODBUS 127.0.0.1:5502  [*]   FC  11: Get Comm Event Counter
MODBUS 127.0.0.1:5502  [*]   FC  12: Get Comm Event Log
MODBUS 127.0.0.1:5502  [*]   FC  17: Report Server ID
MODBUS 127.0.0.1:5502  [*]   FC  22: Mask Write Register (Exception 4)
MODBUS 127.0.0.1:5502  [*]   FC  43: Device Information
```

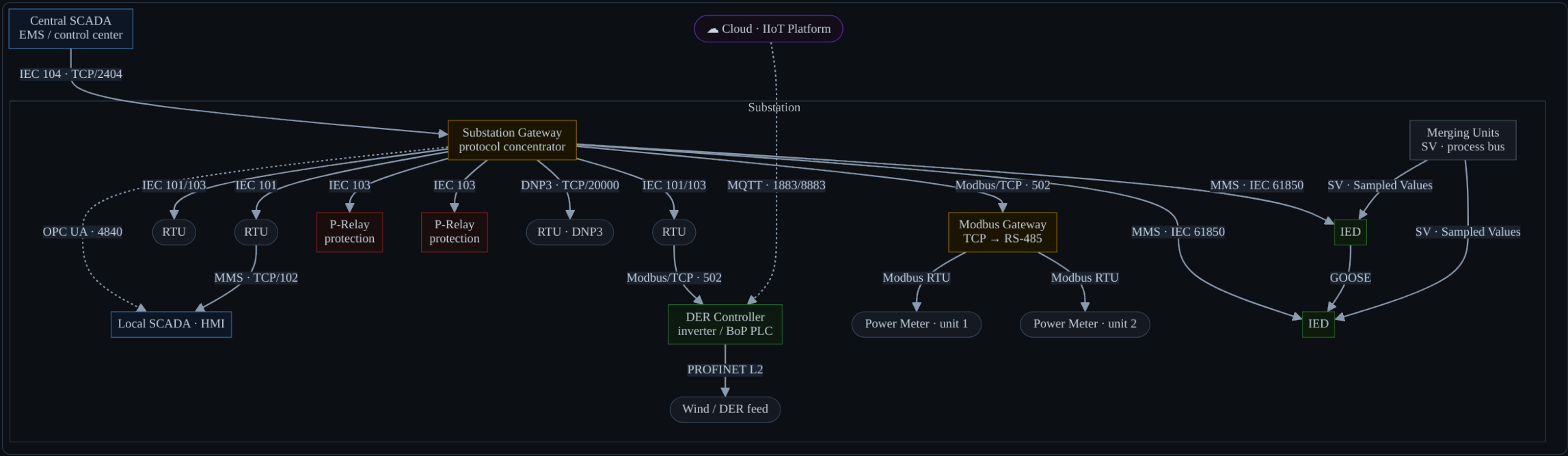
OPC UA, inside the address space

OIDA allows to interact with complex OPC UA data models:

- **Node browse**: recursive `BrowseRequest` from the Objects root.
- **Variable read and write**: values, data types, access levels.
- **Endpoints**: security modes, policy URIs, the server certificate.
- **Anonymous access, Bruteforce, Cert Checks**

Substation Example

Example for an electrical substation:



Coverage of substation protocols

Protocol	Metasploit	ISF	OIDA
IEC 60870-5-104	✓	✗	✓
IEC 61850 MMS	✗	✗	✓
IEC 61850 GOOSE	✗	✗	✓
Modbus	✓	partial	✓
OPC UA	partial	✗	✓
MQTT	partial	✗	✓
Siemens S7 (classic)	✓	✓	✓

04 · Fuzz

A ready-to-go black-box fuzzer.

Black-box fuzzing

A **boofuzz**-based black-box fuzzer with OT protocols implemented:

- **Reach**: 30+ protocol fuzzers, many with no scanner at all.
- **Mutation**: boundary values, type confusion, radamsa-style mangling.
- **Monitoring**: adapts probes to what each protocol implements.

Two fuzzers, two layers

`fuzz` is the standalone one. The scanners carry a second one: `--fuzz`.

- **Standalone**: attacks the protocol implementation: grammar, length fields, state machine.
- **In-scanner**: `$ oida modbus <target> --fuzz -d f32` is **type- and register-aware**, never malforms the framing.
- **The target**: not the parser, the app behind it. What does firmware do when a setpoint reads `NaN`?
- **Register maps**: hits only **writable points**, then restores the originals.

Time to demo



Thank you and check out the tool.

```
$ pip install oida-ics[all]
```

Sponsors



LIMES SECURITY

More sponsors welcome :-)