#sf21veu

Cybersecurity-Oriented Network Traffic Analysis

Luca Deri, Matteo Biscosi
ntop

Martin Scheu
Switch

#sf21veu

Hello!

Luca

Matteo

Martin

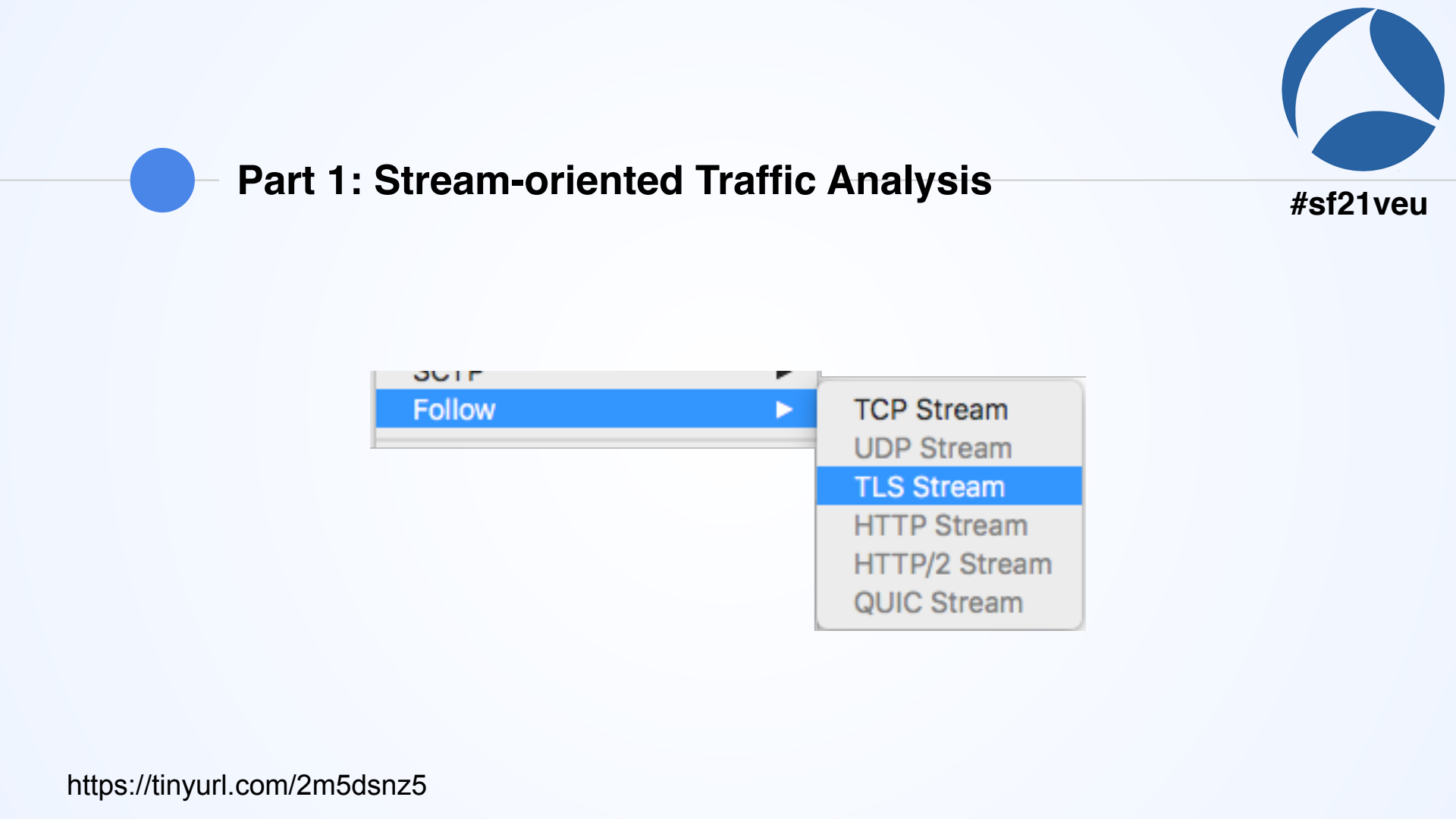
https://tinyurl.com/2m5dsnz5

#sf21veu

Agenda

- Part 1: Luca
  - Stream-oriented Traffic Analysis: nDPI + Wireshark
    - Technologies: Extcap, Lua Wireshark Scripts
- Part 2: Matteo
  - Scripting Wireshark for Cybersecurity
    - Technologies: Lua + Tshark
- Part 3: Martin
  - Cybersecurity in Industrial Control Systems
    - Technologies: PyShark, Wireshark

https://tinyurl.com/2m5dsnz5



Part 1: Stream-oriented Traffic Analysis

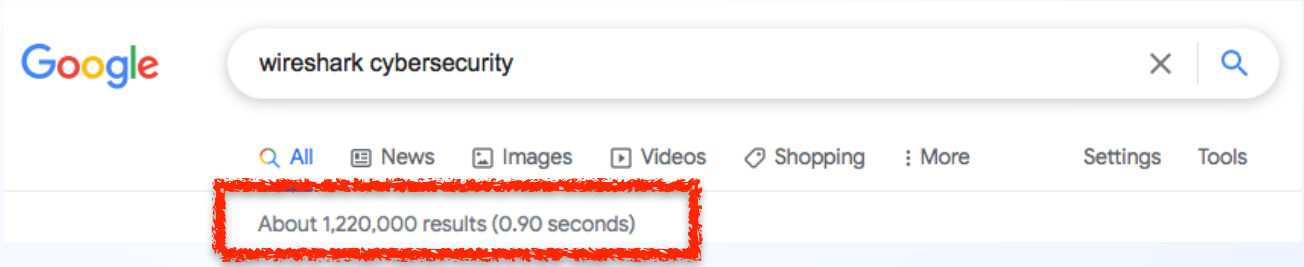
#sf21veu

https://tinyurl.com/2m5dsnz5

#sf21veu

Wireshark in Cybersecurity: Introduction

- Wireshark is a popular tool in cybersecurity:
  - Capture and filter traffic.
  - Diagnose attacks and visualize packet payload content.
  - Analyzing encrypted traffic packets.



https://tinyurl.com/2m5dsnz5

- Martin Mathieson, Snort Alerts in Wireshark, #SF16EU
- Maher Adib, Know Abnormal, Find Evil, #SF18US
- Brad Duncan, Analyzing Windows malware traffic with Wireshark, #SF19US

https://tinyurl.com/2m5dsnz5

Wireshark in Cybersecurity: Challenges

#sf21veu

- Wireshark has been designed to dissect traffic in detail, and provide security analysts user friendly features to this complex task.
- Analysing a cybersecurity accident is a challenging task that requires very advanced technical skills that only very few experts have.
- Goal: how we can lower the bar in order to allow Wireshark to be used in cybersecurity more easily?

https://tinyurl.com/2m5dsnz5

Cybersecurity Traffic Analysis Fundamentals [1/3]

- NIDS (Network Intrusion Detection System) is a software application that monitors a network for malicious activity or policy violations and sends alert when a suspicious event is detected.
- IPS (Intrusion Prevention System) is similar to an NIDS with the difference that when a security violation is detected, network traffic is blocked.
- IDS = Monitoring, IPS = Monitoring+Enforcement.

Wireshark

https://tinyurl.com/2m5dsnz5

Cybersecurity Traffic Analysis Fundamentals [2/3]

#sf21veu

- Security violations can be detected with two main methods:
  - Signature-based: detection happens by searching specific traffic patterns in packet headers or content.
  - Anomaly-based: model (often using machine learning techniques) good traffic and compare current traffic against the model to spot violations.

https://tinyurl.com/2m5dsnz5

Cybersecurity Traffic Analysis Fundamentals [3/3]

```
alert tcp any any -> any [443,465] (msg:"Detected non-
TLS on TLS port"; flow:to_server; app-layer-protocol:!
tls; threshold: type limit, track by_src, seconds 90,
count 1; sid:210003; rev:1;)
```

Example of a Suricata (popular signature-based open source NIDS) compatible rule that generates an alert when non-TLS traffic is detected on TCP ports 443 or 465.

More rules at:
https://rules.emergingthreats.net/open/suricata/rules/

Retrospective #SF17EU

- In 2017 we have presented our integration of nDPI (https://github.com/ntop/nDPI) with the purpose of enhancing Wireshark application protocol detection.
- The goal was to provide better traffic visibility to the Wireshark user community.

https://tinyurl.com/2m5dsnz5

Wireshark in Cybersecurity: Goal

- As follow-up to #SF17EU we now present how we have enriched Wireshark in order to:
  - Automatically detect popular network traffic issues without manually digging into huge packet traces.
  - Provide high-level traffic summaries that will ease analysis of network traffic by pre-processing traffic and highlighting flows that need attention.

Cybersecurity Analysis with Wireshark [1/2]

#sf21veu

- Goal: extend Wireshark with stream-oriented traffic analysis techniques for spotting suspicious communications.
- How: complement native traffic analysis with security-oriented features that can be used to detect popular malware patterns.
- Why: ease security analyst work by automatically identifying relevant traffic streams that can be used as starting point for detecting security violation.

https://tinyurl.com/2m5dsnz5

Cybersecurity Analysis with Wireshark [2/2]

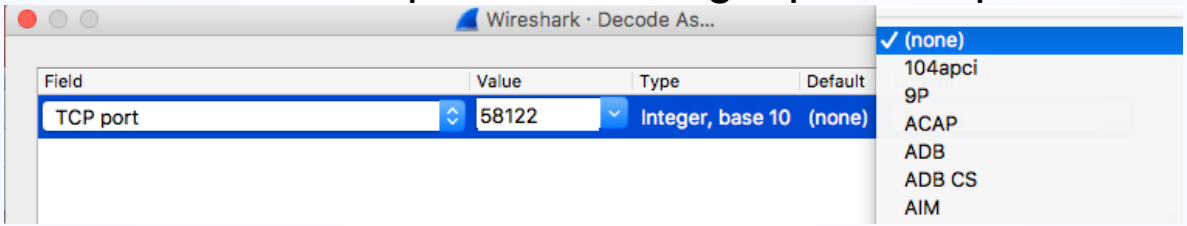
- Wireshark is able to natively cluster traffic in network streams (also known as flows): packets that share the same tuple key: VLAN, Protocol, IP/Port src/dst.
- Idea: complement native stream-oriented analysis with security oriented analysis

https://tinyurl.com/2m5dsnz5

#sf21veu

Deep Packet Inspection [1/3]

- Wireshark detects protocols using a protocol/port association



- Deep Packet Inspection (DPI) is the ability to inspect packet payload in order to detect the application protocol (e.g. TLS.Netflix) and extract deleted metadata (e.g. TLS certificate information) regardless of the IP/protocol/port used by the stream carrying such packets.

https://tinyurl.com/2m5dsnz5

#sf21veu

Deep Packet Inspection [2/3]

- In 2012 ntop decided to develop our own GNU LGPL DPI toolkit order to build an open source DPI layer.
- Protocols supported exceed 250+ and include:
  - P2P (BitTorrent)
  - Messaging (Viber, Whatsapp, Telegram, Facebook)
  - Multimedia (YouTube, Last.gm, iTunes)
  - Conferencing (Skype, Webex, Teams, Meet, Zoom)
  - Streaming (Zattoo, Disney, Netflix)
  - Business (VNC, RDP, Citrix)

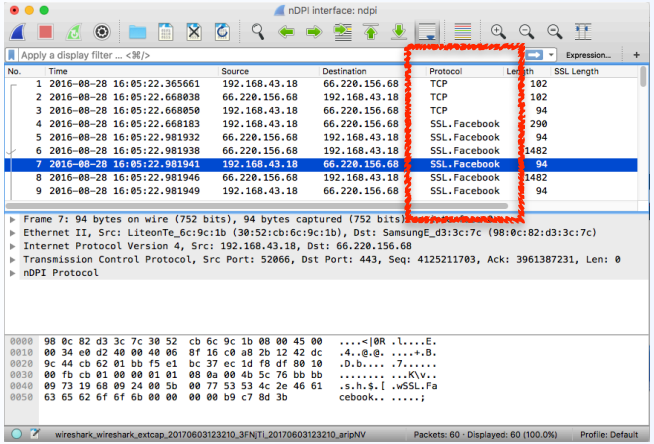
https://tinyurl.com/2m5dsnz5

Deep Packet Inspection [3/3]

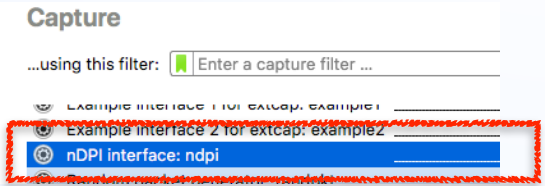
#sf21veu

Lua nDPI script to visualise protocol information

and create traffic reports



Capture live traffic or read a pcap file



- Presented at #SF17EU.

https://tinyurl.com/2m5dsnz5

#sf21veu

Wireshark in Cybersecurity

- Wireshark is the most popular tool for packet-oriented cybersecurity traffic analysis.
- What is missing is the ability to drive security analysts towards relevant issues without having to manually analyse large traces…
- …and simplify data analysis by extending native packet analysis with additional techniques (e.g. DGA detection) that are not present in Wireshark.

https://tinyurl.com/2m5dsnz5

nDPI and Cybersecurity

#sf21veu

- Analyses encrypted traffic to detect issues hidden but un-inspectable payload content.
- Extracts metadata from selected protocols (e.g. DNS, HTTP, TLS..) and matches it against known algorithms for detecting selected threats (e.g. DGA hosts, Domain Generated Algorithm).
- Associates a “risk” with specific flows to identify communications that are affected by security issues.

https://tinyurl.com/2m5dsnz5

#sf21veu

nDPI Flow Risks

- HTTP suspicious user-agent
- HTTP numeric IP host contacted
- HTTP suspicious URL
- HTTP suspicious protocol header
- TLS connections not carrying HTTPS (e.g. a VPN over TLS)
- Suspicious DGA domain contacted
- Malformed packet
- SSH/SMB obsolete protocol or application version
- TLS suspicious ESNI usage
- Unsafe Protocol used
- Suspicious DNS traffic
- TLS with no SNI
- XSS (Cross Site Scripting)
- SQL Injection
- Arbitrary Code Injection/Execution
- Binary/.exe application transfer (e.g. in HTTP)
- Known protocol on non standard port
- TLS self-signed certificate
- TLS obsolete version
- TLS weak cipher
- TLS certificate expired
- TLS certificate mismatch
- DNS suspicious traffic
- HTTP suspicious content
- Risky ASN
- Risky Domain Name
- Malicious JA3 Fingerprint
- Malicious SHA1 Certificate
- Desktop of File Sharing Session
- TLS Uncommon ALPN

https://tinyurl.com/2m5dsnz5

From Flow Risk to Score

- Flow traffic analysis is too granular and it needs to be consolidated into:
  - Network Interface
  - Host/Network/Customer.
  - ASN/country
- In essence that is the pillar for creating a (client/server) numerical score that can be quickly used to spot issues (network, security…).

https://tinyurl.com/2m5dsnz5

#sf21veu

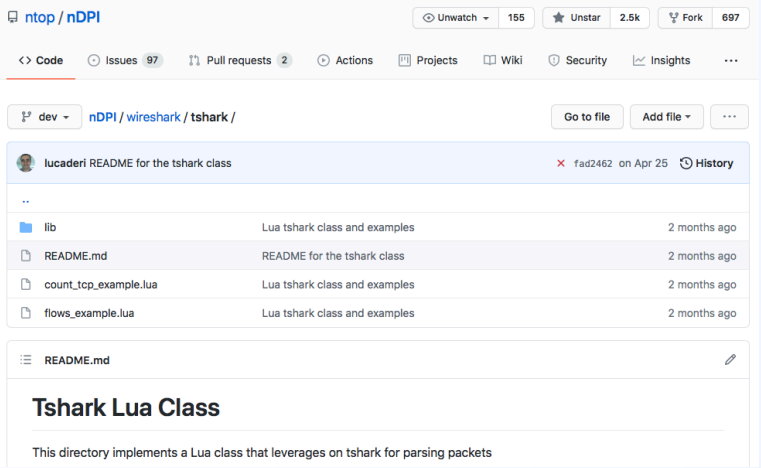
Demo

https://tinyurl.com/2m5dsnz5

#sf21veu

Bonus Track: What about Lua outside Wireshark?

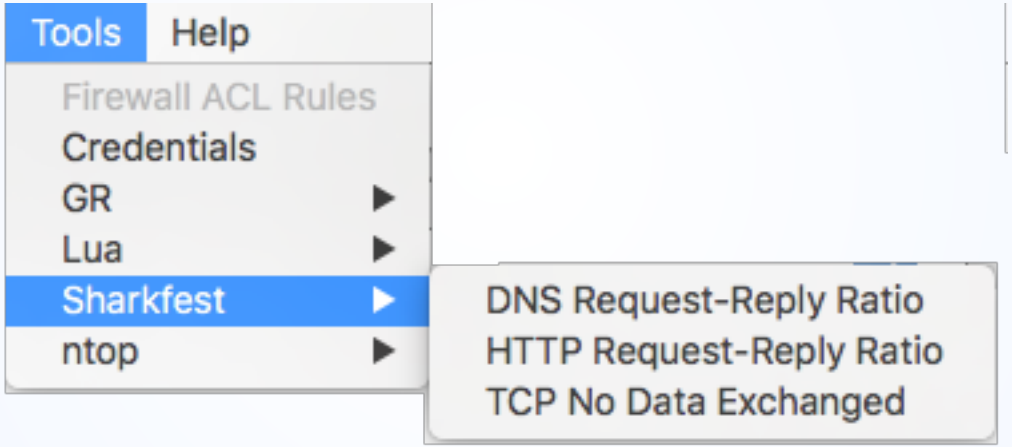
- At ntop we love Lua. Pyshark is a popular library but it is written in Python.
- If you like Lua and OOP, using our Lua Tshark class, to write your Wireshark-based scripts.



https://tinyurl.com/2m5dsnz5

#sf21veu

Part 2: Scripting Wireshark for Cybersec



https://tinyurl.com/2m5dsnz5

Lua Scripts in Wireshark [1/2]

#sf21veu

- It is possible to develop Lua scripts in Wireshark
  - Simple and lightweight programming language
  - Lua can be used to write dissectors, taps, and capture file readers and writers
  - Popular topic at Sharkfest

https://tinyurl.com/2m5dsnz5

Lua Scripts in Wireshark [2/2]

#sf21veu

- Good way to analyze traffic by using Wireshark filters
  (available here: https://www.wireshark.org/docs/dfref/)
- Analyze flows and hosts traffic to detect possible attacks, suspicious traffic or possible crash/errors

https://tinyurl.com/2m5dsnz5

Some Cybersecurity Scripts

#sf21veu

- During this talk the following scripts will be covered in detail:
  - HTTP Request/Reply Ratio (HTTP Server down / crashed)
  - DNS Request/Reply Ratio (DNS Flood Attack, DNS server down / crashed ...)
  - TCP No Data Exchanged (SYN Scan, SYN Flood)
  - Scanning Attacks

https://tinyurl.com/2m5dsnz5

#sf21veu

Brief Configuration Setup

- *init.lua* controls whether or not Lua scripts are enabled via the enable_lua variable
- to run Wireshark with lua scripts:
  -X lua_script:file.lua
- The Lua code is executed after all protocol dissectors are initialized and before reading any file.

```
-- init.lua
--
-- initialize wireshark's lua
--
--  This file is going to be executed before any other lua script.
--  It can be used to load libraries, disable functions and more.
--
-- Wireshark - Network traffic analyzer
-- By Gerald Combs <gerald@wireshark.org>
-- Copyright 1998 Gerald Combs
--
-- SPDX-License-Identifier: GPL-2.0-or-later

-- Set enable_lua to false to disable Lua support.
enable_lua = true

if not enable_lua then
    return
end

-- If set and Wireshark was started as (setuid) root, then the user
-- will not be able to execute custom Lua scripts from the personal
-- configuration directory, the -Xlua_script command line option or
-- the Lua Evaluate menu option in the GUI.
run_user_scripts_when_superuser = true
```

https://tinyurl.com/2m5dsnz5

Develop a Script in Lua [1/2]

#sf21veu

- Register the Menu Entry (register_menu)
- Create the Window and add the Listener (TextWindow.new | Listener.new)
- Analyze each packet (tap.packet)
- Draw the results (tap.draw | window:clear)

https://tinyurl.com/2m5dsnz5

#sf21veu

Develop a Script in Lua [1/2]

```
-- This function will be called once for each packet
function tap.packet(pinfo,tvb)
        http_table = processPackets(pinfo,tvb, http_table)
end
```

```
-- This function will be called once every few seconds to update our window
function tap.draw(t)
    tw:clear()

    for flow in pairs(http_table) do
                local requests = http_table[flow]["requests"]
                local replies = http_table[flow]["replies"]
    local ratio = 0
```

```
-- Register the menu Entry
register_menu("Sharkfest/HTTP Request-Reply Ratio", httpReqRepRatio, MENU_TOOLS_UNSORTED)
```

```
-- Declare the window we will use
local tw = TextWindow.new("HTTP Request/Reply Ratio")

local http_table = {}

local tap = Listener.new();
```

https://tinyurl.com/2m5dsnz5

#sf21veu

Develop a Script in Lua [2/2]

- Cleanup functions

```
-- This function will be called whenever a reset is needed
-- e.g. when reloading the capture file
function tap.reset()
        tw:clear()
        http_table = {}
end
```

```
local function removeListener()
        -- This way we remove the listener that otherwise will remain running indefinitely
        tap:remove();
end

-- We tell the window to call the remove() function when closed
tw:set_atclose(removeListener)
```

```
-- Ensure that all existing packets are processed.
retap_packets()
```

https://tinyurl.com/2m5dsnz5

HTTP/DNS Request/Reply Ratio

- Really important indicator: usually it should be 1 reply per request (1:1)
- Having a ratio different from 1 could be an indicator of various problems:
  - Server unreachable: DNS/HTTP server could be victims of some attacks and could be down or could be crashed
  - Server reachable with low ratio: the Network could be congested or the server too slow to respond

https://tinyurl.com/2m5dsnz5

TCP No Data Exchanged [1/2]

- Usually a TCP flow should exchange data: if no data (from layer 4 ISO-OSI and upwards) is exchanged then there could be a problem (e.g.):
  - TCP SYN Scan
  - TCP SYN Flood

https://tinyurl.com/2m5dsnz5

#sf21veu

TCP No Data Exchanged: TCP SYN Scan [2/2]

- SYN scanning is the most common type of port scanning
- Uses a SYN scan to determine the status of ports on the remote target
- SYN-scanning sends the first packet only (the one marked with the SYN flag - TCP). It waits for either a RST, ACK or SYN,ACK response.
  - RST,ACK: nothing is running on the port
  - SYN,ACK: a service is known to be running on the port.

https://tinyurl.com/2m5dsnz5

#sf21veu

Demo

https://tinyurl.com/2m5dsnz5

Part 3: Cybersecurity in Industrial Control Systems

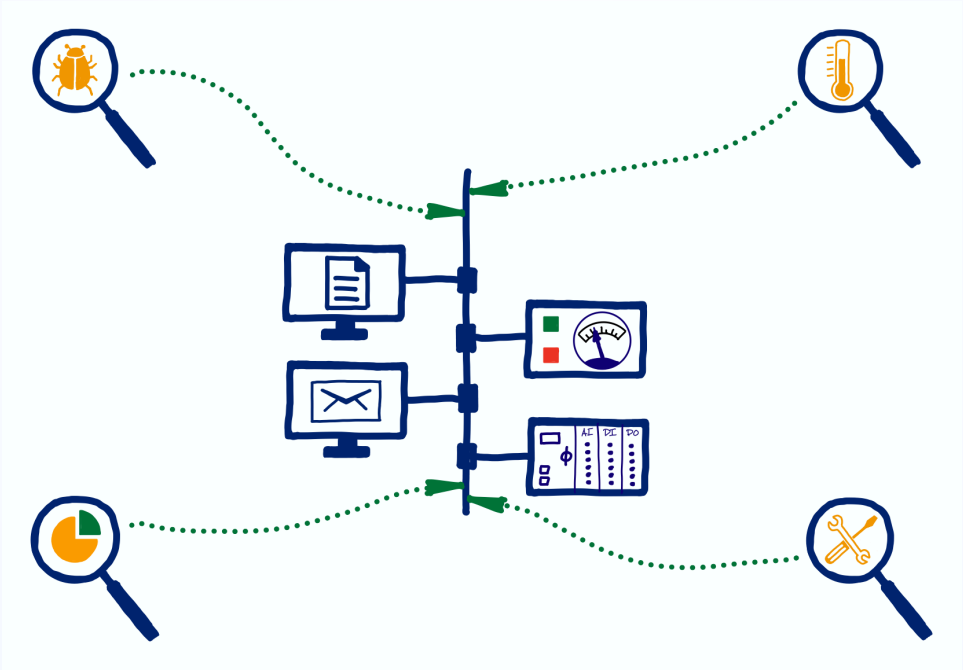
#sf21veu

https://tinyurl.com/2m5dsnz5

#sf21veu

Why monitoring?

Things which do not belong there



Health

Performance

Configuration errors

https://tinyurl.com/2m5dsnz5

Specific Characteristics of Industrial Network Data

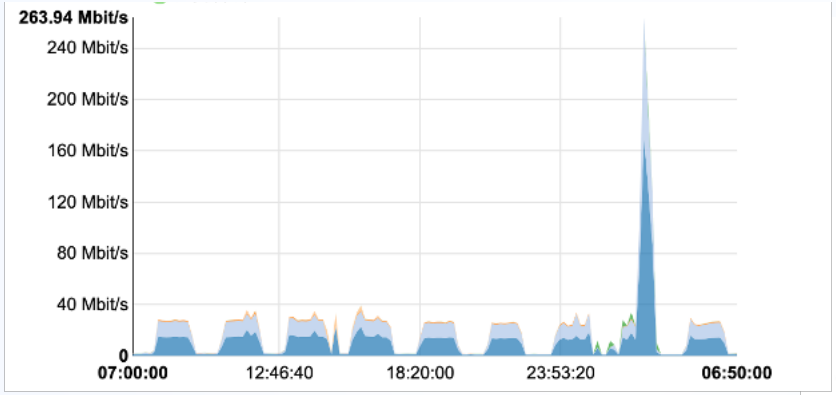
- Long lived flows (> 12h)
- Protocol implementation are vendor / system integrator specific
- Goodput typically below 50%
  (IP + TCP compared to payload data)

https://tinyurl.com/2m5dsnz5

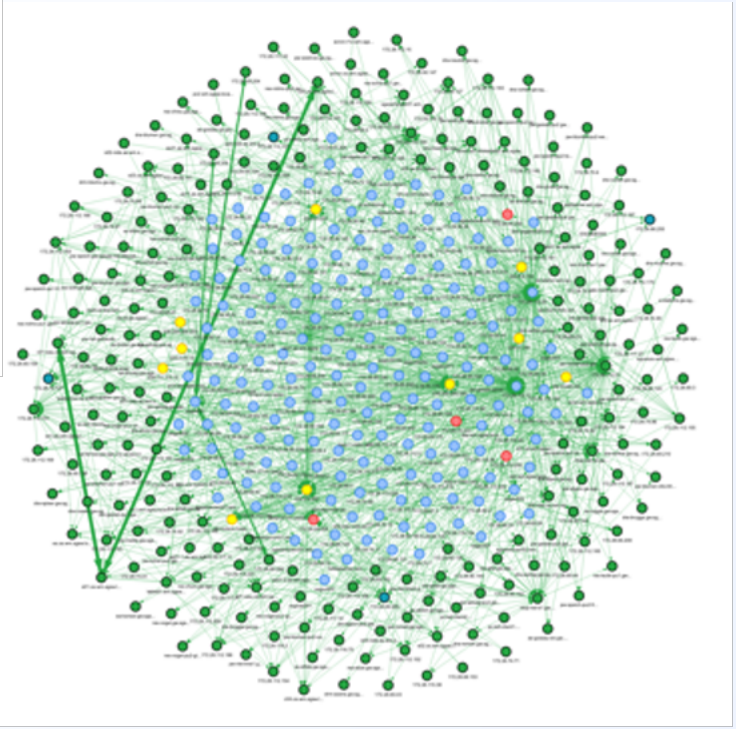
#sf21veu

Cybersecurity - Baselining

how does "normal" look like?



Application	Total Bytes	Percentage
HTTP	881.84 GB	48.0 %
Unknown	374.75 GB	20.4 %
SMBv23	171.28 GB	9.3 %
s7comm	158.66 GB	8.6 %
SSDP	60.94 GB	3.3 %
RDP	59.99 GB	3.3 %
ICMP	28.85 GB	1.6 %
MDNS	27.13 GB	1.5 %
NTP	14.0 GB	0.8 %
IEC60870	12.97 GB	0.7 %



https://tinyurl.com

#sf21veu

Industrial Protocol - IEC 60870-5-104

- 60870 standards are developed by IEC
- Widely used in:
  - electrical energy distribution
  - water / waste water processing
- IP / TCP based

https://tinyurl.com/2m5dsnz5

#sf21veu

Industrial Protocol - IEC 60870-5-104

No.	Time	Source	Destination	Protocol	Length	ntop Extensions	nDPI Protocol Interp
1	0.000000	167.180.175.212	167.180.191.69	IEC60870	123		✓

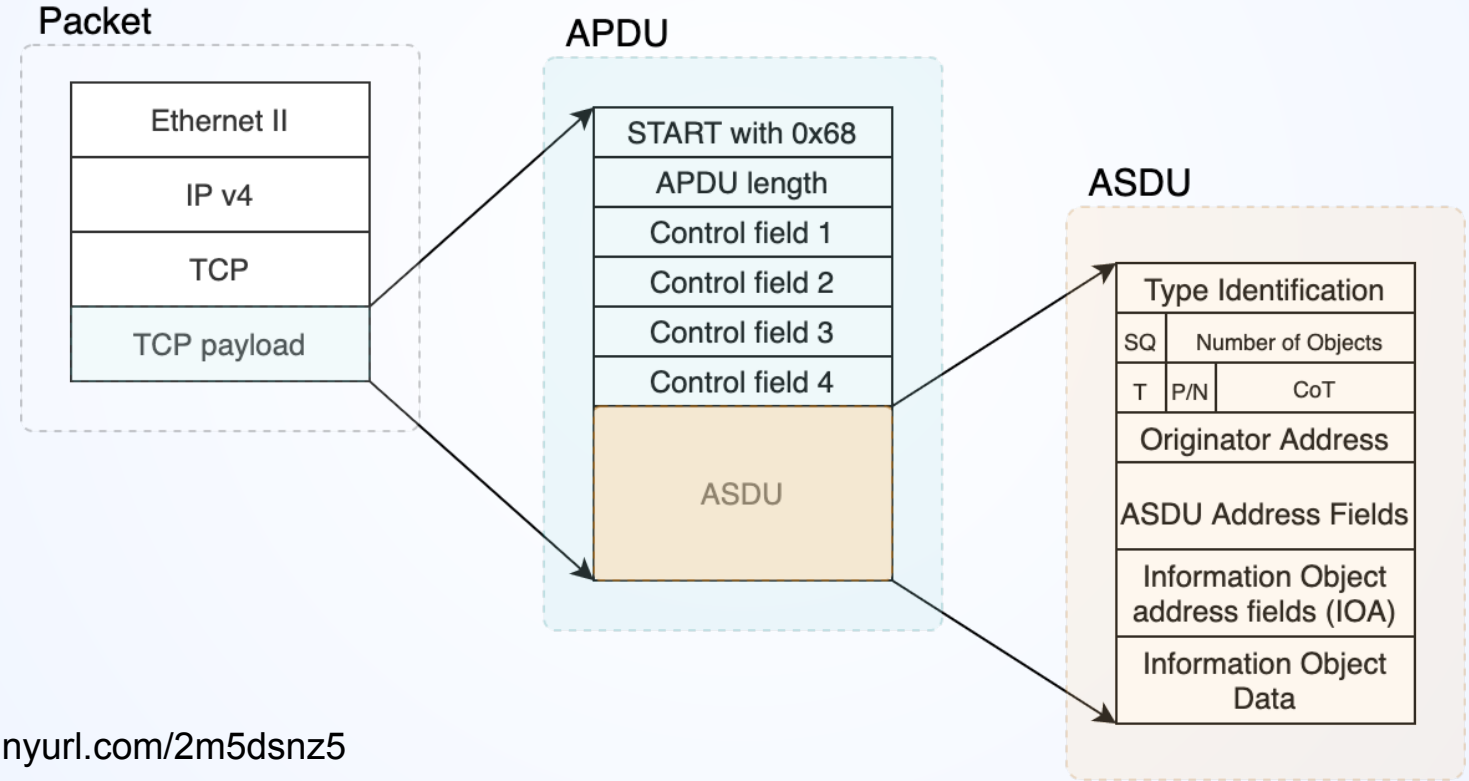
```
▸ Frame 1: 123 bytes on wire (984 bits), 123 bytes captured (984 bits) on interface /tmp/wireshark_ex
▸ Ethernet II, Src: 00:66:aa:d1:32:c2 (00:66:aa:d1:32:c2), Dst: 00:44:66:fc:29:af (00:44:66:fc:29:af)
▸ 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 21
▸ Internet Protocol Version 4, Src: 167.180.175.212, Dst: 167.180.191.69
▸ Transmission Control Protocol, Src Port: 2404, Dst Port: 64971, Seq: 1, Ack: 1, Len: 27
▸ IEC 60870-5-104: -> I (22380,132)
▸ IEC 60870-5-101/104 ASDU: ASDU=69 M_ME_TF_1 Spont   IOA=1290 'measured value, short floating point
```

```
0000  00 44 66 fc 29 af 00 66  aa d1 32 c2 81 00 00 15   ·Df·)··f ··2·····
0010  08 00 45 00 00 43 08 df  00 00 3b 06 b8 53 a7 b4   ··E··C·· ··;··S··
0020  af d4 a7 b4 bf 45 09 64  fd cb 15 49 84 49 e5 dd   ·····E·d ···I·I··
0030  a8 59 50 18 20 00 36 c2  00 00 68 19 d8 ae 08 01   ·YP· ·6· ··h·····
0040  24 01 03 00 45 00 0a 05  00 00 3a 2d 44 00 35 5f   $···E··· ··:-D·5_
0050  9b 09 52 0c 0c 19 68 09  24 00 00 00 f5 00 00 00   ··R···h· $·······
0060  00 00 00 00 00 00 00 49  45 43 36 30 38 37 30 00   ·······I EC60870·
0070  00 00 00 00 00 00 00 77  fb 28 8a                  ·······w ·(·
```

https://tinyurl.com/2m5dsnz5

#sf21veu

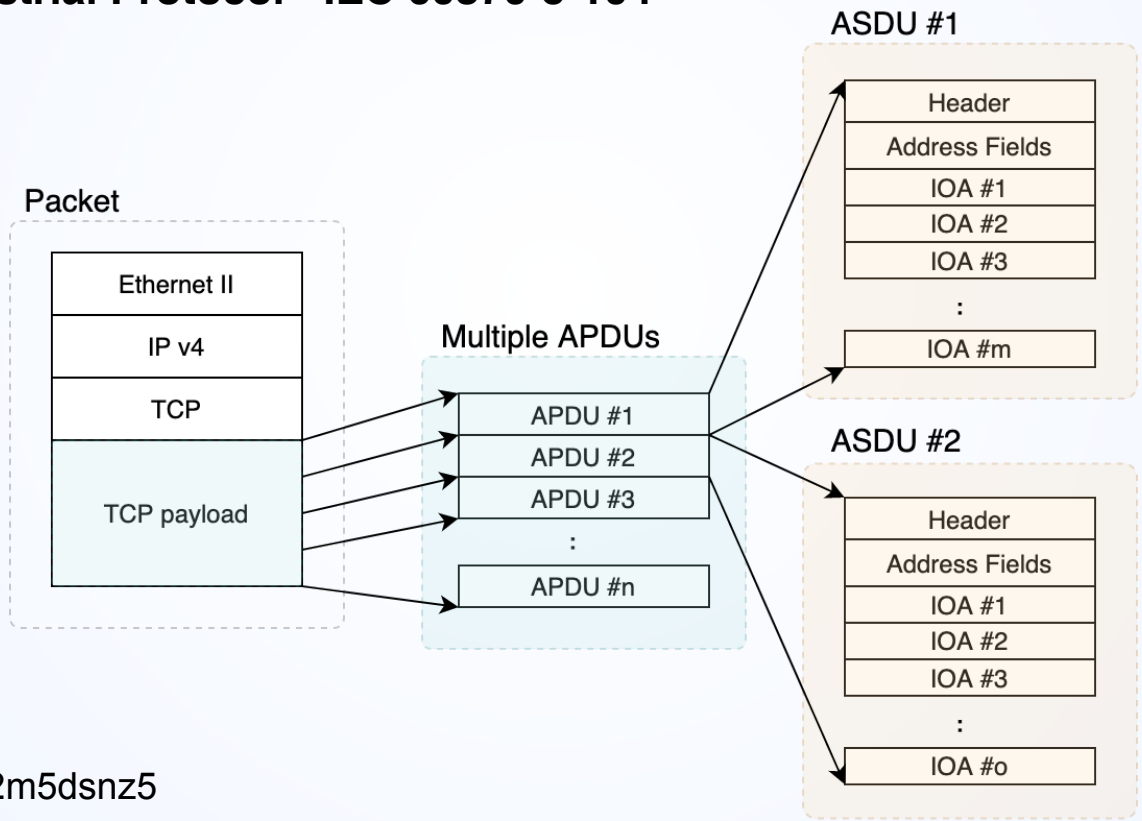
Industrial Protocol - IEC 60870-5-104



https://tinyurl.com/2m5dsnz5

#sf21veu

Industrial Protocol - IEC 60870-5-104



https://tinyurl.com/2m5dsnz5

#sf21veu

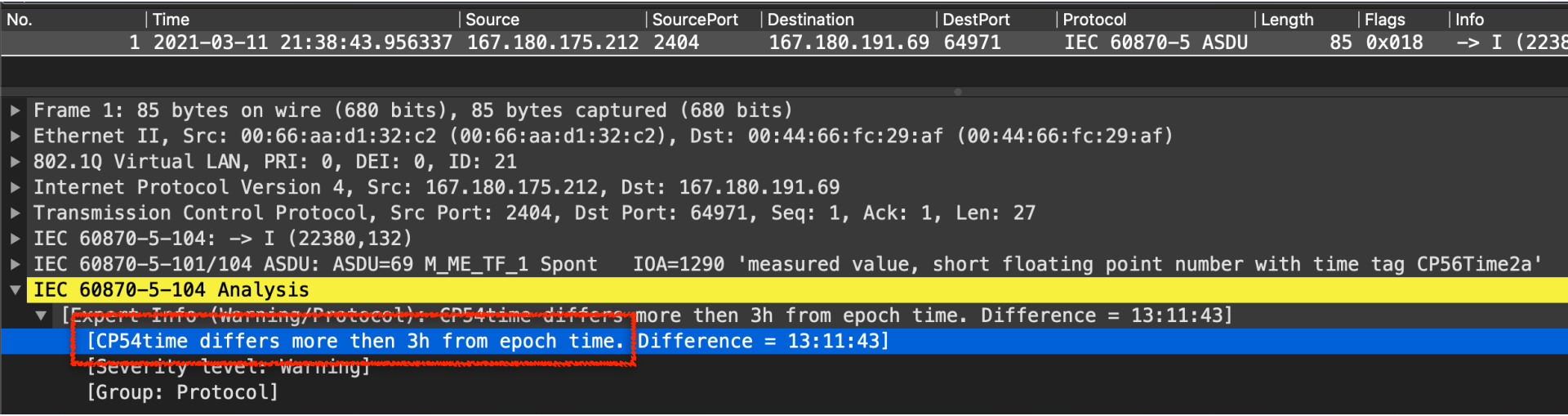
Expert Info: IEC 60870-5-104 Analysis

- Helps operators to identify protocol issues
- Easy to deploy
- Algorithm can be used for network monitoring

https://tinyurl.com/2m5dsnz5

IEC 60870-5-104 - Configuration Issue

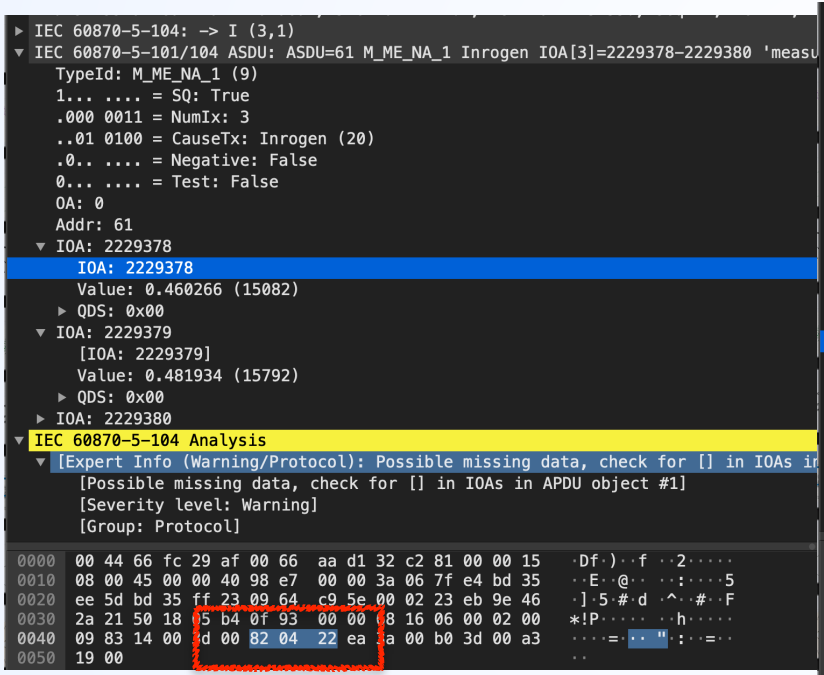
#sf21veu

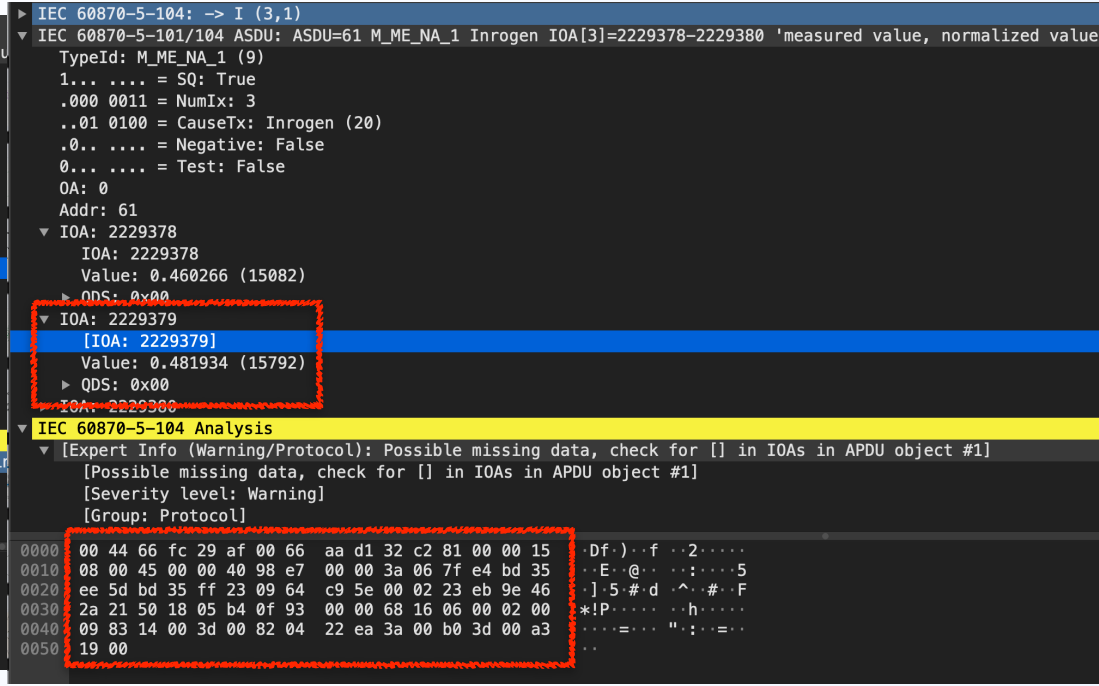


https://tinyurl.com/2m5dsnz5

IEC 60870-5-104 - Configuration Issue

#sf21veu

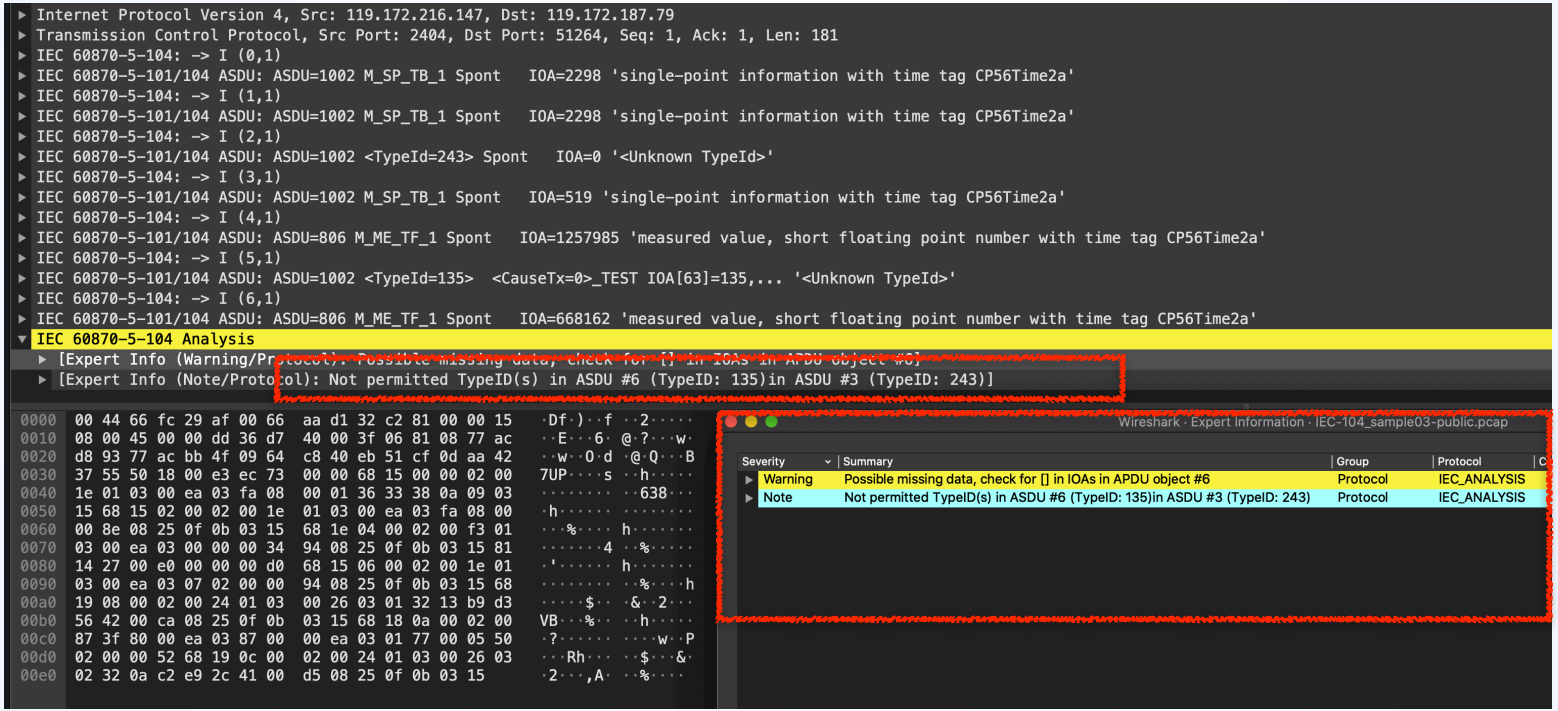




https://tinyurl.com/2m5dsnz5

IEC 60870-5-104 - Configuration Issue

#sf21veu

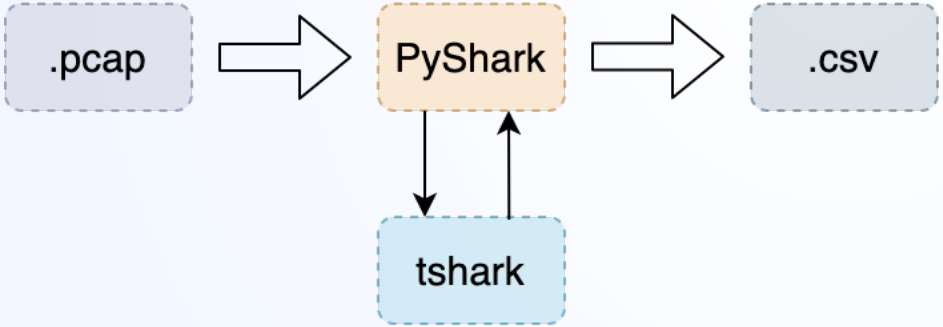


https://tinyurl.com/2m5dsnz5

#sf21veu

PyShark

- Replace MAC / IP address by ID#
- Extract necessary protocol fields (same names as in Wireshark, e.g. tcp.payload)



Challenges

- Different protocol implementations
- Site specific configurations

https://tinyurl.com/2m5dsnz5

PyShark

#sf21veu

Flags	SrcID	SrcPort	DstID	DstPort	_m	Type	TypeID	APDU_Num	NumIx	CauseTx	ORG	ASDUAddr	Tx	Rx	UType	:tA_N	IOA	IOA Value	CP54Time
0010	1	63685	2	2404															
0018	3	2404	1	63708		I	36	1	1	3	0	219	11956	727	-	1	946	431.015625	2012-7-2_0:23:23.2
0018	4	2404	1	63684		I	36	1	1	3	0	804	20013	740	-	1	945	78.0	2021-3-10_17:44:1
0018	5	2404	6	51498		U	0	1	0	0	0	0	0	0	TESTFR act				
0018	6	51498	5	2404		U	0	1	0	0	0	0	0	0	TESTFR con				
0018	7	2404	1	63687		I	36	1	1	3	0	801	20594	740	-	1	286	2.200000047683716	2021-3-10_17:44:1
0010	5	2404	6	51498															
0018	8	2404	6	49994		U	0	1	0	0	0	0	0	0	TESTFR act				
0018	6	49994	8	2404		U	0	1	0	0	0	0	0	0	TESTFR con				
0010	1	63681	9	2404															
0018	9	2404	1	63681		I	9	1	3	3	0	60	26193	1	-	1	293	1	
0018	9	2404	1	63681		I	9	1	3	3	0	60	26193	1	-	2	299	2857	
0018	9	2404	1	63681		I	9	1	3	3	0	60	26193	1	-	3	303	2857	
0018	1	63681	9	2404		S	0	1	0	0	0	0	0	26194	-				
0010	9	2404	1	63681															
0010	10	2404	6	49975															
0010	1	63708	3	2404															
0010	1	63684	4	2404															
0010	1	63687	7	2404															
0018	7	2404	1	63687		I	36	1	1	3	0	801	20595	740	-	1	282	115.70000457763672	2021-3-10_17:44:1
0018	7	2404	1	63687		I	36	2	1	3	0	801	20596	740	-	1	281	14.0	2021-3-10_17:44:1
0018	7	2404	1	63687		I	36	3	1	3	0	801	20597	740	-	1	954	115.70000457763672	2021-3-10_17:44:1
0018	7	2404	1	63687		I	36	4	1	3	0	801	20598	740	-	1	283	66.9000015258789	2021-3-10_17:44:1

https://tinyurl.com/2m5dsnz5

Conclusions

- Code can be found at: https://github.com/ntop/nDPI/tree/dev/wireshark

- Thanks everyone for listening!

https://tinyurl.com/2m5dsnz5