How to Dissect Unusual Protocols for Troubleshooting OT Security

nozominetworks.com/blog/how-to-dissect-unusual-protocols-for-troubleshooting-ot-security/

By June 22, 2021

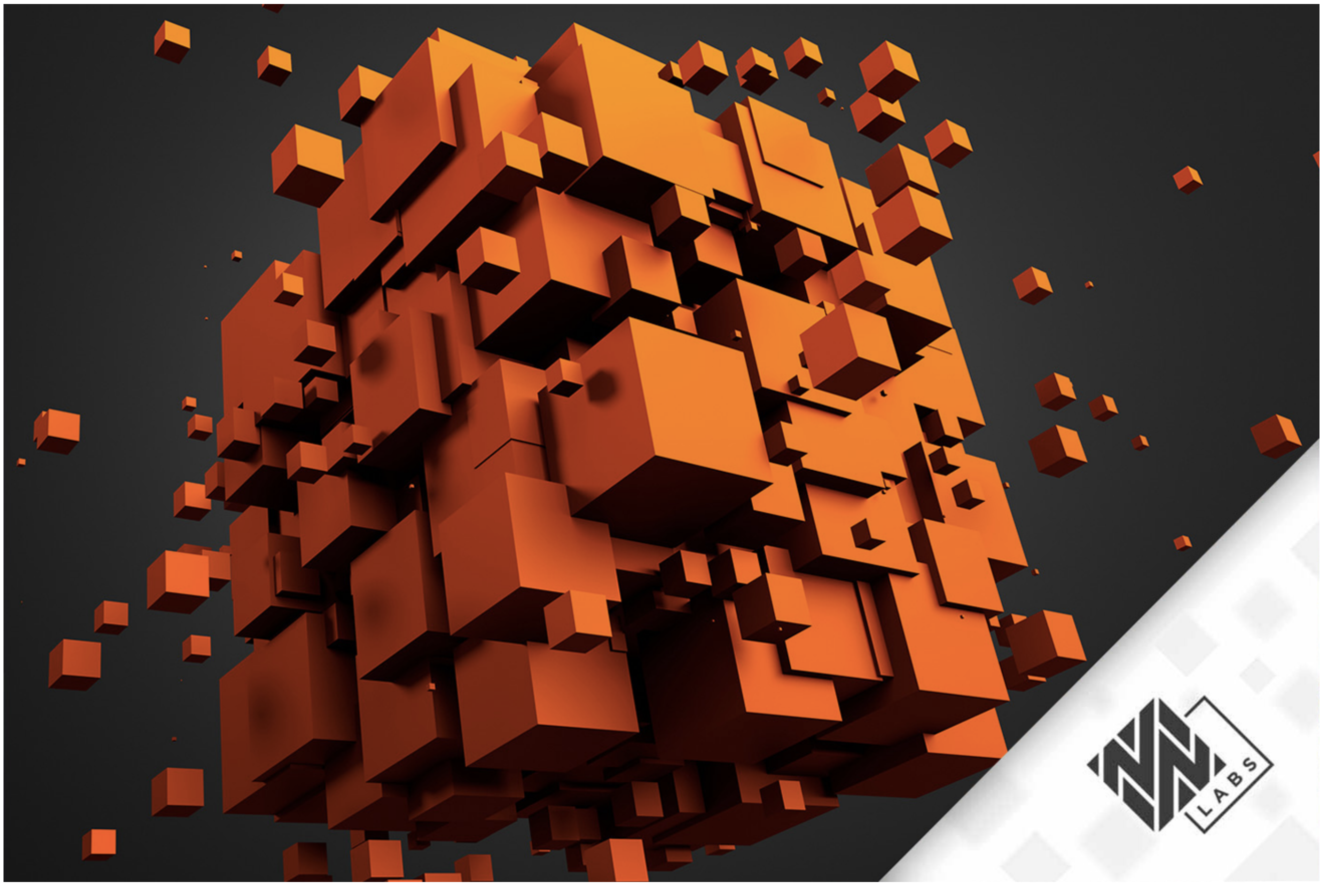
As OT security researchers, the Nozomi Networks Labs team continually works to understand OT and IoT device processes and their security risks. This includes understanding how assets like embedded controllers communicate with each other and their workstations. To do so, we need to reverse undocumented protocols.

Unfortunately, the process of dissection can't be standardized due to the unknown layers of complexity put in place by most vendors when they design systems. This is where the experience of security researchers can make a big difference.

This is the first of a series of articles from Nozomi Networks Labs where we'll demonstrate how to use Lua APIs to instruct Wireshark to properly dissect an undocumented protocol.

In this blog, we look at the steps involved in developing a dissector for a real-world use case, the well-known Cisco Nexus protocol. We've also posted a plug-in on GitHub to help the security community at large.

Whether you're a security researcher yourself, or you manage networks for an asset owner, this methodology and plug-in will help you troubleshoot networking issues and improve overall OT security.



It can be tedious and time-consuming to gain a complete understanding of the unknown protocols used by different security device vendors, but it’s possible to use Lua APIs to more easily dissect them.

Determining How to Dissect Unknown Protocols

One of tools commonly used to kick-start the exploration process for an unknown protocol is Wireshark.

With the right set of traffic/Pcaps – generated by forcing a specific type of communication between controllers – we can start analyzing the protocol.

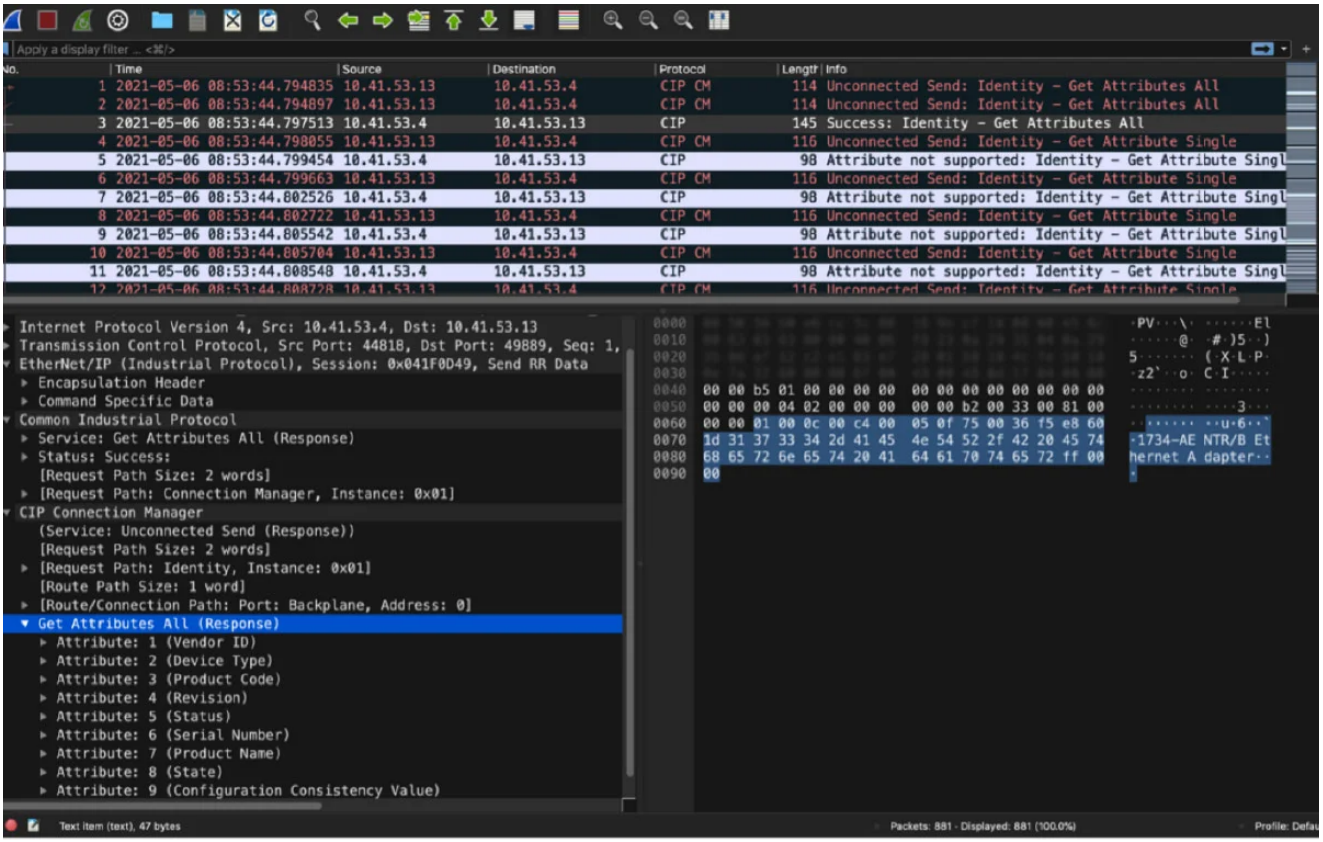
We begin by focusing our attention on patterns. Normally, when we start the protocol reverse engineering process, we’re confronted by an unknown language. We need to identify key elements that will help us understand the communication structure step-by-step (lengths, function codes, sequence numbers, crc, etc.).


Reverse engineering of protocols begins with focusing on unknown language patterns. Plugin scripts written in the Lua language can help dissect packets and validate research findings.

While making assumptions during this phase, it helps to leverage one of Wireshark’s capabilities called plugins. These are scripts written in the Lua programming language that instruct the tool to dissect each packet using our findings. Plugins also allow us to validate findings with the collected and/or live traffic.

TIP: You can also create your own dissectors directly using the native Wireshark C language in cases where performance needs to be fine-tuned.

- Windows: `%APPDATA%\Wireshark\plugins`
- Unix: `~/.local/lib/wireshark/plugins`
- Mac: `~/.config/wireshark/plugins`



Rockwell Ethernet/IP dissection

As you can imagine, the processes involved in gaining a complete understanding of an unknown communication language can be tedious and time-consuming. However, the end result is shareable knowledge (in the form of a plugin) that's highly useful to others.
For example, utility operators can benefit enormously from tools like this every time they need to troubleshoot specific scenarios within a normal industrial process operation.

Rather than describe how to develop a Lua dissector from scratch (there's already a lot of easily-found documentation out there, see the tip below), we'll dive into some specific non-standard real world use cases and how to deal with them.

TIP: For those just starting their Lua dissection journey, take a look to Mika's tech blog: [Creating a Wireshark dissector in Lua – Part 1 (the basics).](#)

Registering a New Dissector

Ethertype and Recall

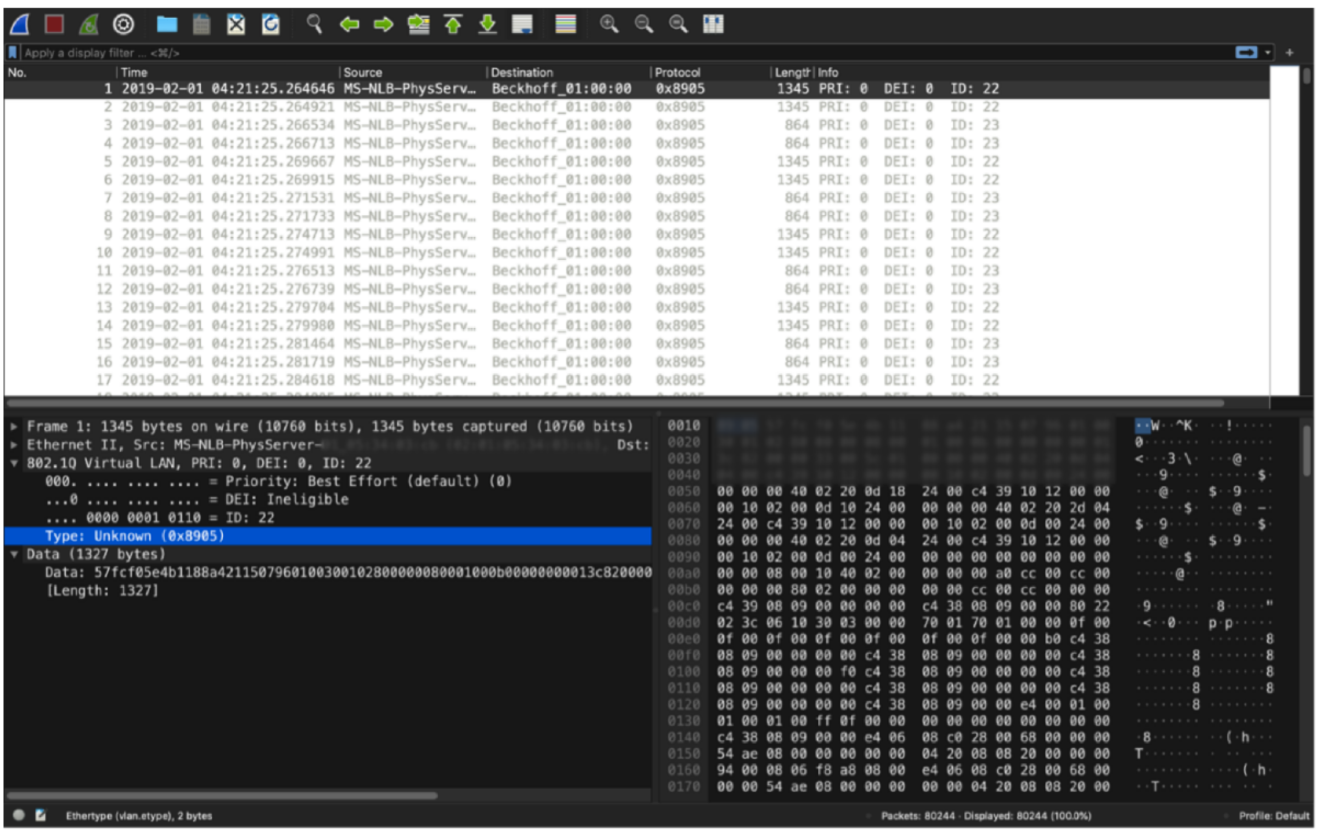
Our first challenge involves dealing with protocols that aren't easily accessible by Wireshark – at least not in the standard way we're used to.

Let’s use Cisco Nexus as an example – a well-known protocol used between the NX-OS switch series. We’ll go through the steps involved in developing the related dissector (note we also reverse engineered its inner structure).

First, the research team noticed that Wireshark doesn’t support the dissection of such a protocol. This means we have to do a bit of investigation and packet analysis in order to understand its structure.

As far as Wireshark knows, we have an 802.1Q Virtual Lan frame. This is a commonly used standard for defining VLANs (Virtual LAN) with a pretty basic structure.

The VLAN ID in place [0x8905] is non-standard. This is why the tool classifies it as an “unknown” type. Let’s assume that that could be a good first indicator for a proprietary protocol, and keep it in mind for later.



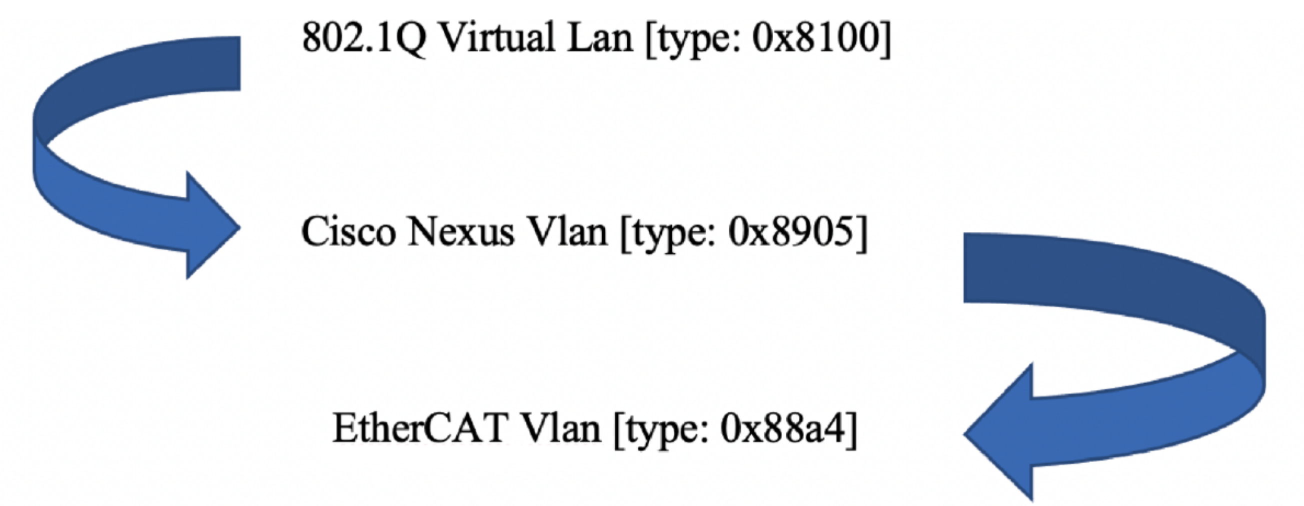
Unknown protocol

After a bit of deeper analysis on some interesting patterns in the “data” content of the packet, we determined that the structure is much simpler than expected. And, from a particular point, it is very similar to another common protocol in the industrial field. This proprietary protocol is actually a combination of a vendor specific layer, plus a well known protocol.

HINT: Do those 0x88a4 bytes ring a bell?

Luckily, the encapsulated protocol is already known by Wireshark. This means that we have to instruct the tool to subscribe (or tag) the unknown layer to properly and accurately detect it every time it's seen over the wire.

Now, recall the discovered encapsulated dissector mentioned earlier.



Ethernet II frame structure

Let's start by writing down some code. First, we have to instruct Wireshark to link to our dissector every time it sees any ethernet packet with the unknown Type ID: 0x8905.
To link properly, we can force the `DissectorTable.get()` function to point in the specified frame area by indicating that we're interested in referencing only the `ethertype` parameter [aka Type ID] and not the standard udp/tcp.port.

```
-- initialize wrapper fields
-- wrapper main function
function cisco_nexus.dissector (buf, pkt, root)
```

```
end
```

```
-- subscribe for Ethernet packets on type 0x8905.
local eth_table = DissectorTable.get("ethertype")
eth_table:add(0x8905, cisco_nexus)
```

Next, we have to initialize a second dissector table that we're going to recall at a specific offset. For this one, we'll use a different approach: Lua gives us the ability to point at a specific known dissector every time we need it by using the get_dissector() function, and then use it through the call() function.

Let's see these functions in action.

Because the nested protocol has a defined VLAN tag within the 802.1Q IEEE standard, we can store the DissectorTable related to it in the `original_vlan_dissector` variable at the initialization section of our script:

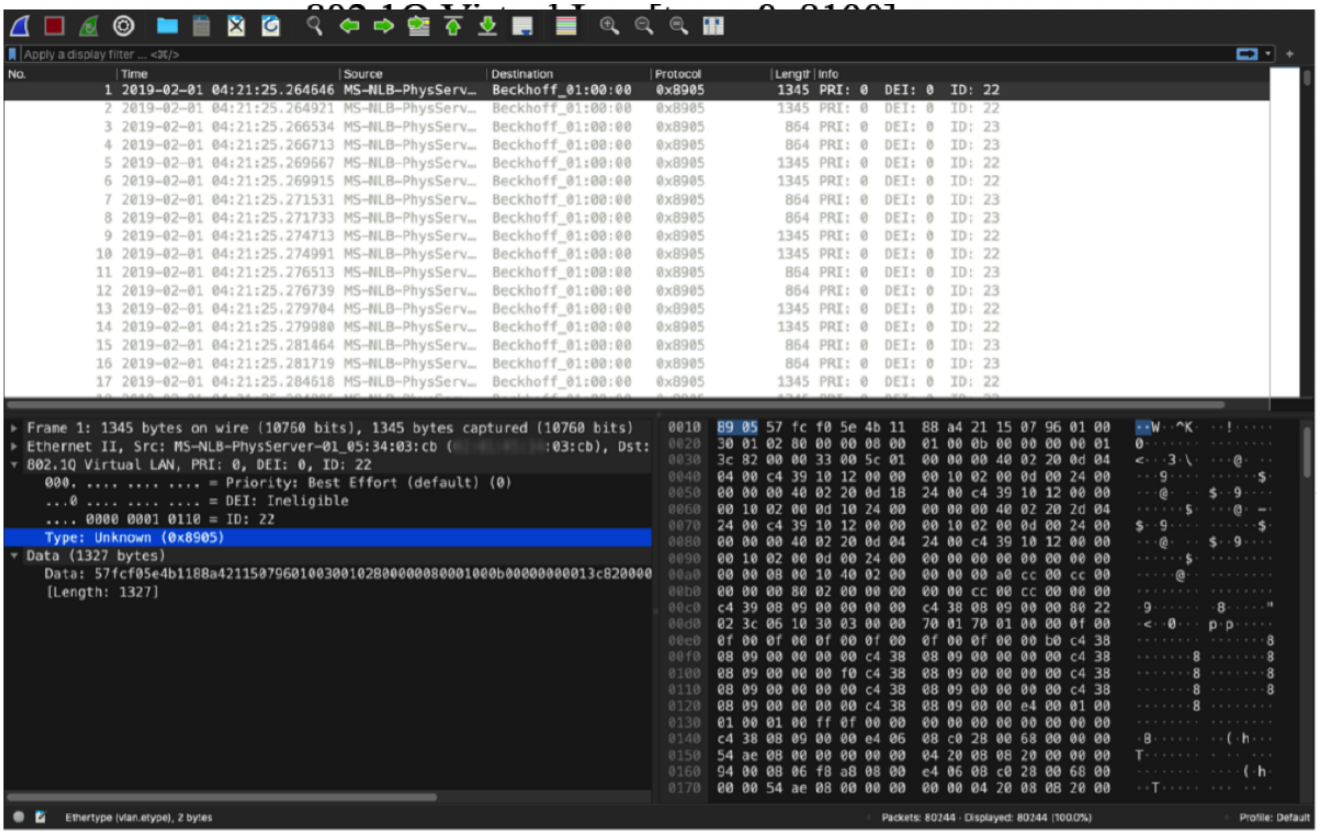
```
-- load 802.1Q Virtual LAN dissector
original_vlan_dissector =
DissectorTable.get("ethertype"):get_dissector(0x8100)
```

and then call it at the right offset after the Cisco Nexus header, within the context of the main dissection function, precisely after the 4 header bytes:

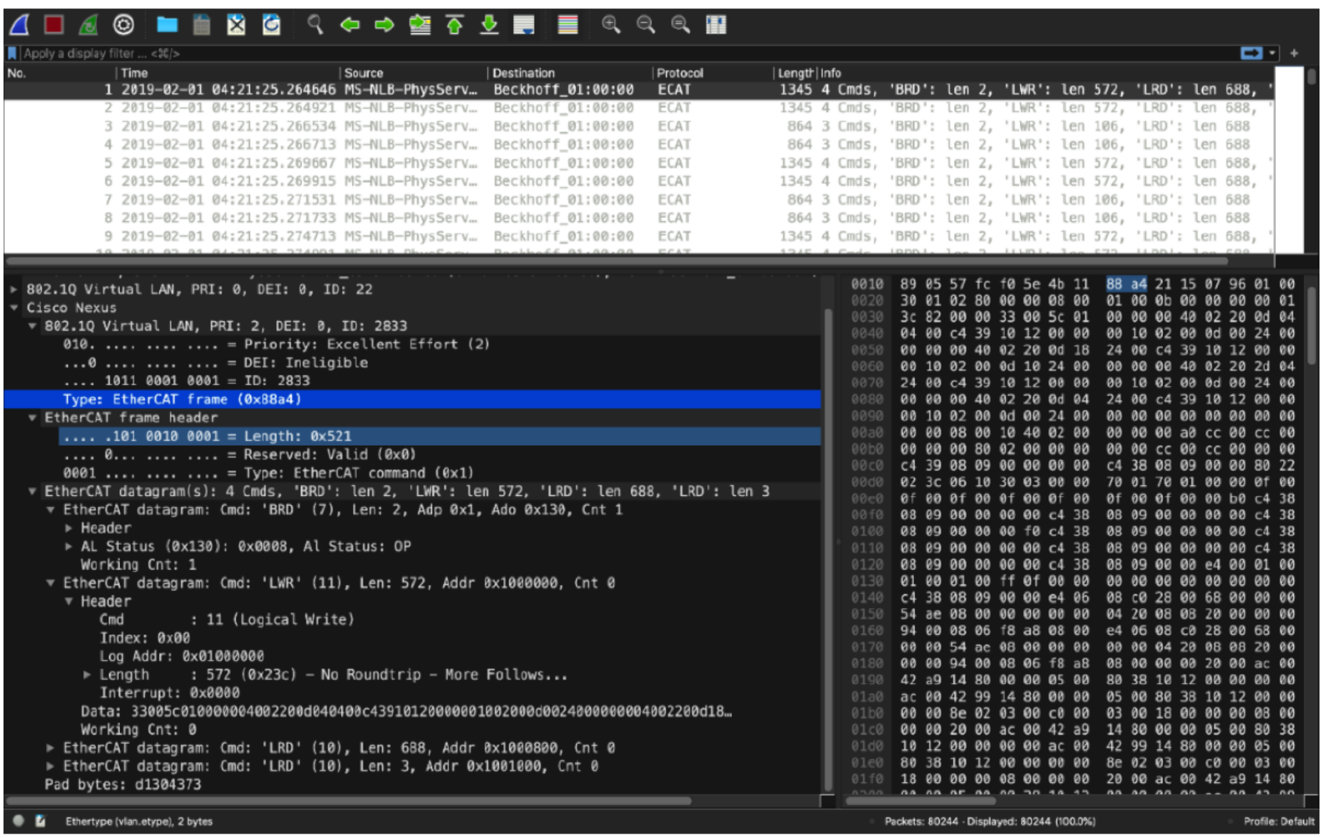
TIP: The first length check is done to ensure that we avoid any 0 byte packets, in case some are found.

```
-- wrapper main function
function cisco_nexus.dissector (buf, pkt, root)
-- validate packet length is adequate, otherwise quit

if buf:len() == 0 then return end
pkt.cols.protocol = cisco_nexus.name

-- create subtree for Cisco Nexus
subtree = root:add(cisco_nexus, buf())

-- subscribes ECAT dissector
original_vlan_dissector:call(buf:range(4,buf:len()-4):tvb(), pkt, subtree)

end
```

The final result is a complete dissection of the entire packet structure.



Before: Detection of an unknown protocol



After: Final dissection of the unknown protocol's inner structure

Using the Cisco Nexus Protocol to Create the Cisco Nexus Dissector Plugin

In this article, we showed how to easily manage known and unknown layer 2 frames using Lua APIs, by instructing Wireshark to properly dissect them while the analysis is being done. To illustrate this, Nozomi Networks researchers used a real-world example of a previously unknown protocol: Cisco Nexus.

During the background analysis, the team found that the proprietary protocol encapsulated a well-defined communication structure. We then leveraged this knowledge to test the following combinations of DissectorTable functions:

- .get(v1)
- .get(v1):get_dissector(v2)
- .get(v1):get_dissector(v2):call(v3,v4,v5)

The outcome of our research is a plugin called the Cisco Nexus Dissector. We've posted it in GitHub to help asset owners troubleshoot activities within their own networks. The global security community can also use it to further their analysis and research projects.

Next month, Nozomi Networks Labs will investigate how to create a plugin for another unknown protocol found in a commonly-used industrial communications equipment. Stay tuned!

Related Content

NOZOMI NETWORKS LABS

TRITON: The First ICS Cyber Attack on Safety Instrument Systems

Understanding the Malware, Its Communications and Its OT Payload

Alessandro Di Pinto, Younes Dragoni, Andrea Carcano

Black Hat USA 2018 - Research Paper

RESEARCH REPORT

TRITON: The First ICS Cyber Attack on Safety Instrument Systems

Understanding the Malware, Its Communications and Its OT Payload

Learn about:

- How the TRITON cyberattack unfolded and why it's so important
- Understanding TRITON through reverse engineering
- The implications of TRITON on industrial control systems security
- Two free community tools that help you defend against TRITON

Download

Related Links

- Research Report: Nozomi Networks Labs OT/IoT Security Report
- Github.com: Nozomi Networks Dissectors: Cisco Nexus
- Blog: Demonstrating the Link Between Functional Safety and ICS Security
- Blog: Colonial Pipeline Ransomware Attack: Revealing How DarkSide Works
- Blog: Black Hat: Understanding TRITON, The First SIS Cyber Attack
- Blog: The Clever Use of Post Dissectors to Analyze Layer 2 Protocols
- Webpage: Nozomi Networks Labs
- Data Sheet: Threat Intelligence

Younes Dragoni

Security Researcher

Younes Dragoni is a member of the World Economic Forum's Global Shaper Community, a worldwide network of young people actively shaping our future through solution building, policy-making and lasting change. His fascination with computer security, and desire to be on the offensive side, began many years ago. Now, as Security Researcher with Nozomi Networks, Younes thrives on hunting down vulnerabilities in automation devices (ICS/SCADA) and examining malicious software to understand the nature of threats to industrial operations.