5-2023

# Improvements to Passive Fingerprinting of Operational Technology Environments

Lawrence Wellman
*Boise State University*

# Improvements to Passive Fingerprinting of Operational Technology Environments

by

Lawrence Wellman

A project completed in fulfillment

of the CORe 591 requirements for the degree of

Master of Science in Cyber Operations and Resilience

Boise State University

May 2023

# Improvements to Passive Fingerprinting of Operational Technology Environments

Lawrence Wellman
Cyber Operations and Resilience Program
Boise State University

***Abstract*— This paper explores the effectiveness of three network tools for analyzing network traffic and highlights their reliance on network ports to fingerprint TCP and UDP network protocols. Considering this limitation, the paper introduces protoDetect, a novel tool demonstrating a possible solution for identifying Operational Technology (OT) network protocols.**

***Keywords—Operational Technology, ICS, Network Fingerprinting***

## I. Introduction

**Motivation.** Operational Technology (OT) networks are essential for monitoring and controlling critical infrastructure environments such as power grids, manufacturing plants, and water treatment systems. The disruption of these environments can have significant and immediate effects on the citizens and corporations that rely on their continued operation.

Systems in an OT environment are often designed and installed by third parties and operated by a combination of vendors and onsite engineers. They are often left in place until they fail to work, resulting in many generations of devices. The vendor may no longer support these devices or still be operating. The security of OT networks can be improved by identifying and classifying the network traffic generated by the various OT devices. Recognizing the multiple connections, systems, and applications in an environment is a preliminary step in security monitoring, penetration testing, incident response, system backup, and recovery [1]. This research paper proposes a novel method for passively fingerprinting Industrial Control System (ICS) network traffic, targeting incident responders looking for solutions that are not environment-specific specifically without relying on the network port.

**Limitation.** Existing passive fingerprinting techniques used by tools in the field require protocols to either be on a default and expected port or that the operator will know and be able to configure tools where the default port has been modified. Individual tools may be able to scan and identify specific protocols on nonstandard or all ports [2]. However, this capability has only been identified for a limited number of widespread protocols or is available only as vendor-proprietary tools.

**Approach.** This paper reviews three network traffic parsing tools and finds they cannot effectively parse traffic transmitted on nonstandard ports. To overcome this limitation, a novel approach is proposed using methods demonstrated in [3]–[6]. Further, this paper presents a series of techniques to classify network traffic by extracting unique traits identified in the data sections of the protocols. This is then demonstrated through the development of protoDetect, which incorporates unique fingerprints for Modbus [3], S7Communication (S7Comm) [4], Distributed Network Protocol 3 (DNP3) [5], and Ethernet/IP [6].

**Contribution.** The main contribution of this research is showing that currently, available network tools cannot identify traffic configured to run on a different port. It also presents a tool for fingerprinting connections irrespective of the port transmitted on.

**Roadmap.** Section 2 provides an overview of existing work. Section 3 looks at the results of commonly used tools and shows that they fail to identify protocols where the ports have been manipulated. Section 4 describes and illustrates the tool's results developed to address this issue. Section 5 discusses the developed tool's shortcomings and future work and concludes in Section 7.

## II. Related Work

Interest in device fingerprinting is growing in the academic and industrial research communities. This research is commonly split between active methods which send traffic to the device to determine its type and protocols. Active fingerprinting can be done by sending intentionally manipulated packets to identify variations in the device's response to commands that return device information. Typical examples of active fingerprinting are Nmap [7], Xprobe2 [8], or PLCScan [9]. Passive fingerprinting relies on monitoring device emanations to identify variances that can be used to identify unique attributes of the device. For example, discrepancies in individual packet flags, communications delays and other timing, session interactions, and packet application data. Typical tools for passive fingerprinting include p0f [10] and GrassMarlin [11]. Research in [12] examines network flows extracted from collected network traffic and separates ICS from non-ICS devices.

The described research primarily focuses on identifying specific devices rather than specifically looking at identifying particular protocols. [13] looks at methods for classifying traffic on the fly using the first five packets to identify traffic

that may be attempting to hide by changing its ports, such as peer-to-peer (p2p) traffic.

## III. Review of Commonly Used Tools

This section looks at Wireshark, GrassMarlin, and SynSaber OT PCAP Analyzer, specifically each tool's capability to identify network protocols. It identifies that each tool relies primarily on network ports to determine protocol and that if the port is changed, they fail to recognize the protocol successfully.

Wireshark is a free and open-source network packet analyzer with many uses, including troubleshooting networks, examining potential security issues, verifying network applications, and debugging protocol implementations [14]. It can capture network traffic and save it as a packet capture (PCAP). Wireshark can parse approximately 3000 protocols [15].

Wireshark has two primary methods for protocol identification. The first is a link dissector that defines when traffic should be forwarded to the dissector. This is generally implemented by adding a dissector with a field, value, and a handle, e.g., "tcp.port, 22, ssh_handle". Multiple dissectors will often be added to support various ports for the same protocol handler. The second method is a heuristic dissector, which will check bytes in incoming packets to match conditions defined in the protocol description. This can be used to identify protocols that contain easily identifiable attributes such as a magic number or message IDs. Heuristic dissectors are used for approximately 271 parsers. However, this is commonly combined with link dissectors. For example, in cases where multiple protocols overlap on network ports and thus still rely on the network port for identification. Thus, of the 3000 protocols supported by Wireshark, only around 64 protocols don't rely on the network port for protocol identification.

GRASSMARLIN is a Java-based program released by the NSA. The tool aims to provide situational awareness of industrial control systems but has received few updates since its initial release in 2016. GRASSMARLIN leverages an older version of Wireshark to process network traffic but includes 119 fingerprints to identify different OT devices and protocols. These protocols are defined in XML files. The only file which does not leverage the port in its fingerprinting is the Operating System which leverages TTL to identify different operating systems.

SynSaber released a free OT PCAP Analyzer in 2023 [16]. This tool runs a local web server that allows users to upload a PCAP; it will then provide an overview of the vendor and protocols identified in the PCAP. This tool's source is unavailable, so the protocol identification method is unknown.

Additionally, to test these tools' capability to identify protocols, two datasets of PCAPs were used. The first is a collection of ICS PCAPs from various online locations provided on GitHub [17]. The second is a set of PCAPs used to test parsers developed for Malcolm, also retrieved from GitHub [18].

A modified version was created for each PCAP file which changed UDP and TCP ports by adding 2001 to each port. The files were altered using Scapy, a Python library for reading and packet manipulation, which was used to read the files, make the modifications to the source and destination ports and then recalculate the checksum and length of the packet. 2001 was selected because it pushed all well-known ports (0, 1023) outside this range. Adding 1 to each network port handles cases where default ports are often changed by adding the number to itself, for example, by chaining port 22 to port 2222 or port 80 to port 8080. Finally, this simple modification allowed the researcher to reverse the change while reviewing the files.

Then for each PCAP, the original file was loaded into the three tools, and then the modified version was loaded into the tool to identify any changes in protocol detection. For Wireshark, the packet dissections were exported as a CSV file then the protocol column was compared. For GrassMarlin and SynSaber, the tools were manually loaded, and then differences were compared manually.

## IV. protoDetect: ICS Protocol Detection

This section presents protoDetect, a command line tool developed for PCAPs, and identifies network protocols. This tool takes a directory or a file as input and writes connections to a log file. This output can be used to configure security tools with the port used for various protocols used. Three fields were identified and used to fingerprint each protocol. These included a protocol identifier or specific bytes that help identify a particular protocol, length fields, and cyclic redundancy check (CRC) or checksum. Where possible, a combination of these fields was used.

### A. *Modbus TCP*

Modbus was originally a serial protocol developed by Modicon to communicate with its programmable logic controllers (PLCs) [19]. It has been adapted to communicate over multiple methods, including Modbus TCP, a variant that defaults to port 502.

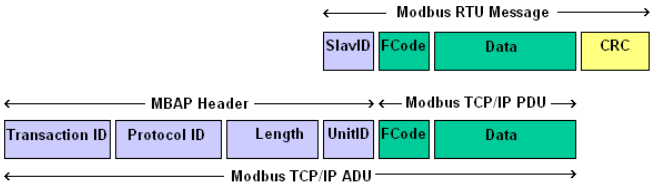


Image 1: Modbus Protocol Description [20]

The Modbus protocol fingerprint uses two methods to detect the protocol. The first is the protocol identifier "0x0000". However, two bytes, especially all zero bytes, may overlap with other protocols and lead to a high false positive rate. The second method uses the length field, which counts the bytes following the length field to the end of the packet. Combining these two methods reduced false positives in the given data set.

### B. *S7Comm*

S7Comm is a Siemens proprietary protocol that runs between PLCs in the Siemens S7-300/400 family [21].

S7Comm consists of a combination of protocols, including, at a minimum, ISO transport services on top of the TCP (TPKT) and connection-oriented transport protocol (COTP). S7Comm, by default, uses port 102.

The S7Comm protocol fingerprint uses similar methods to the Modbus protocol. The protocol identifier in the header is the hex bytes “0x32”. The TPKT layer contains a length layer that counts the entire length of the data section.

### *C. DNP3*

DNP3 is a set of protocols used in process automation and typically used by electric and water companies [22]. The default port for DNP3 is 20,000.

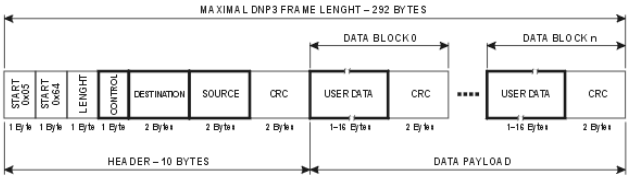


Image 2: DNP3 Protocol Description [23]

This protocol contains several reoccurring structures allowing it to be identified confidently. In the header, the protocol starts with a protocol identifier “0x0564”. The length field is calculated by adding the bytes in the control, destination, source fields, and user data sections. Each user data section is 16 bytes, with the final block containing the remaining bytes. The header and user data sections contain a CRC, which can be verified. For the current data set, sufficient accuracy was achieved by looking for the protocol identifier and calculating the header CRC.

### *D. Ethernet/IP or CIP*

Ethernet/IP is one of a few protocols that utilize CIP and is one of the leading industrial protocols used in the United States [24]. The default ports for Ethernet/IP are TCP 44818 and UDP 2222.

This protocol contains a protocol identifier of hex “0x 00000000” and a length field. However, unlike other fingerprinted protocols, CIP uses zero-length packets containing only the header. This renders the fingerprint for these two fields unable to identify these specific packets as CIP reliably.

## V. DISCUSSION

This section will discuss the tool's results and looks at future work that can improve the research findings. One comparison is included below of Wireshark and protoDetect.

| | Wireshark | protoDetect |
|---|---|---|
| Modbus | 99472 | 99478 |
| DNP3 | 38 | 24 |
| S7comm | 106421 | 212794 |
| Ethernet/IP | 0 | |

Table 1: Number of Packets for Unchanged 4SICS-GeekLounge-151022.pcap

| | Wireshark | protoDetect |
|---|---|---|
| Modbus | 0 | 99478 |
| DNP3 | 0 | 24 |
| S7comm | 0 | 212794 |
| Ethernet/IP | 0 | 0 |

Table 2: Number of Packets identified for Modified 4SICS-GeekLounge-151022.pcap

In Table 1, we can see variations in the number identified by protoDetect when it comes to individual packets. The primary reason for variations was that Wireshark does not count packets with TCP issues, such as TCP Retransmission or TCP duplicate ACKs. However, a few misidentified packets were also identified where protoDetect’s fingerprints were not specific enough, and future revisions could be improved. These issues are mitigated because protoDetect lists the recognized protocols and the host and ports they communicate on. This list is a culmination of the fingerprinting of each packet; thus, misidentifications of a single packet will generally not result in the entire communication being misclassified.

protoDetect relies on a fingerprint being developed for each protocol. A good fingerprint requires multiple verifiable fields available in the protocol. The CRC was the most reliable field of the areas used as it involves a block of bytes to match a specific value. Future work should add protocols and methods to generate automatic fingerprints for a protocol found within a PCAP that currently does not have a defined fingerprint. Originally the protoDetect was developed using Golang, and issues were identified in the gopacket library for handling packets that did not adhere to TCP specifications. This resulted in the script being rewritten in Python using Scapy. While this proved to be more reliable, it did come at a significant time cost to analyze a file. Future work should identify methods to speed up the protocol analysis.

## VI. CONCLUSION

This paper examines the challenges of fingerprinting network protocols in an operational technology network. It reviews the current capabilities of network capture analysis tools and identifies their reliance on network ports to fingerprint TCP and UDP network protocols. It then proposes a potential solution that could be applied to run a first pass identifying OT network protocols and identifying which port they are communicating with. The proposed methods are then demonstrated using protoDetect. There is then a discussion of future works that could help overcome shortcomings in the presented tool.

## References


[1] "The 18 CIS Controls," *CIS*. https://www.cisecurity.org/controls/cis-controls-list/ (accessed Jun. 25, 2022).

[2] "Configuring HTTP Inspection on All Ports." https://sc1.checkpoint.com/documents/R81/WebAdminGuides/EN/CP_R81_DataLossPrevention_AdminGuide/Topics-DLPG/Configuring-HTTP-Inspection-on-All-Ports.htm (accessed Mar. 27, 2023).

[3]"Modbus_Messaging_Implementation_Guide_V1_0b.pdf." Accessed: May 07, 2023. [Online]. Available: https://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf

[4] "78028908_SIMATIC_Comm_DOKU_v23_e.pdf - CPU-CPU Communication with SIMATIC Controllers - ID: 78028908 - Industry Support Siemens." https://support.industry.siemens.com/cs/document/78028908/cpu-cpu-communication-with-simatic-controllers?dti=0&lc=en-GE (accessed May 07, 2023).

[5] "Overview of DNP3 Protocol." https://www.dnp.org/About/Overview-of-DNP3-Protocol (accessed May 07, 2023).

[6] "EtherNet/IP: Industrial Protocol White Paper." Institute of Electrical and Electronic Engineers, 2001. Accessed: May 07, 2023. [Online]. Available: https://literature.rockwellautomation.com/idc/groups/literature/documents/wp/enet-wp001_-en-p.pdf

[7] "Nmap: the Network Mapper - Free Security Scanner." https://nmap.org/ (accessed Jun. 26, 2022).

[8] binarytrails, "binarytrails/xprobe2." May 19, 2022. Accessed: Jun. 26, 2022. [Online]. Available: https://github.com/binarytrails/xprobe2

[9] "Google Code Archive - Long-term storage for Google Code Project Hosting." https://code.google.com/archive/p/plcscan/ (accessed Jun. 26, 2022).

[10] "p0f v3." https://lcamtuf.coredump.cx/p0f3/ (accessed Jun. 26, 2022).

[11] "GRASSMARLIN." NSA Cybersecurity Directorate, Jun. 20, 2022. Accessed: Jun. 25, 2022. [Online]. Available: https://github.com/nsacyber/GRASSMARLIN

[12] I. Chakraborty, B. M. Kelley, and B. Gallagher, "Industrial control system device classification using network traffic features and neural network embeddings," *Array*, vol. 12, p. 100081, Dec. 2021, doi: 10.1016/j.array.2021.100081.

[13] V. Sowinski-Mydlarz, J. Li, K. Ouazzane, and V. Vassilev, "Threat intelligence using machine learning packet dissection," *Trans. Comput. Sci. Comput. Intell.*, Jul. 2021, Accessed: May 07, 2023. [Online]. Available: https://www.springer.com/series/11769

[14] "Chapter 1. Introduction." https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html (accessed Mar. 27, 2023).

[15] "Wireshark · Display Filter Reference: Index," *Wireshark*. http://localhost:3000/docs/dfref/default.html (accessed Mar. 28, 2023).

[16] "SynSaber Launches a Free OT PCAP Analyzer Tool for the ICS Community," *SynSaber | Industrial Cybersecurity*. https://synsaber.com/news-and-events/free-ot-pcap-analyzer-tool-from-synsaber/ (accessed Mar. 28, 2023).

[17] J. Smith, "ICS-pcap." Feb. 02, 2023. Accessed: Feb. 06, 2023. [Online]. Available: https://github.com/automayt/ICS-pcap

[18] "Malcolm-PCAP." mmguero-dev, Mar. 08, 2023. Accessed: Mar. 28, 2023. [Online]. Available: https://github.com/mmguero-dev/Malcolm-PCAP

[19] "Modbus," *Wikipedia*. Apr. 24, 2023. Accessed: May 07, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Modbus&oldid=1151570531

[20] "About Modbus TCP | Simply Modbus Software." https://www.simplymodbus.ca/TCP.htm (accessed May 07, 2023).

[21] "S7comm." https://wiki.wireshark.org/S7comm (accessed May 07, 2023).

[22] "DNP3," *Wikipedia*. Mar. 20, 2022. Accessed: May 07, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=DNP3&oldid=1078136308

[23] "DNP 3," *RACOM*. https://www.racom.eu/eng/support/prot/dnp3/index.html (accessed May 07, 2023).

[24] "EtherNet/IP," *Wikipedia*. Mar. 22, 2023. Accessed: May 07, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=EtherNet/IP&oldid=1146034135