ICS/SCADA Device Recognition: A Hybrid Communication-Patterns and Passive-Fingerprinting Approach

Alaa T. Al Ghazo
Department of Electrical and
Computer Engineering
Iowa State University
Ames, IA 50014, USA
alghazo@iastate.edu

Ratnesh Kumar
Department of Electrical and
Computer Engineering
Iowa State University
Ames, IA 50014, USA
rkumar@iastate.edu

***Abstract*—The Industrial Control System (ICS) and Supervisory Control and Data Acquisition (SCADA) systems are the backbones for monitoring and supervising factories, power grids, water distribution systems, nuclear plants, and other critical infrastructures. These systems are installed by third party contractors, maintained by site engineers, and operate for a long time. This makes tracing the documentation of the systems' changes and updates challenging since some of their components' information (type, manufacturer, model, etc.) may not be up-to-date, leading to possibly unaccounted security vulnerabilities in the systems. Device recognition is useful first step in vulnerability identification and defense augmentation, but due to the lack of full traceability in case of legacy ICS/SCADA systems, the typical device recognition based on document inspection is not applicable. In this paper, we propose a hybrid approach involving the mix of communication-patterns and passive-fingerprinting to identify the unknown devices' types, manufacturers, and models. The algorithm uses the ICS/SCADA devices's communication-patterns to recognize the control hierarchy levels of the devices. In conjunction, certain distinguishable features in the communication-packets are used to recognize the device manufacturer, and model. We have implemented this hybrid approach in Python, and tested on traffic data from a water treatment SCADA testbed in Singapore (iTrust).**

***Index Terms*—Security, Cyber-Physical Systems, Internet of Things, Industrial Control systems, Device recognition, SCADA fingerprinting,**

I. Introduction

ICS/SCADA systems are specialized distributed/networked systems consisting of devices and protocols that interact with the physical systems through sensors, actuators, controllers, monitors, etc. Such systems used to be on isolated networks. However, due to the increase in popularity and advancements of wireless networking and cloud technologies, ICS/SCADA systems have begun to expand their connectivity to the cloud; the extent of such connectivity can vary from system to system. Benefits of connecting to the internet/cloud are substantial (e.g., practically unlimited computing and storage, remote accessibility, etc.), but such connectivity entails the possibility of introducing security vulnerabilities, as the system is no longer isolated.

To catch any existing vulnerabilities in an ICS/SCADA system, a complete understanding of devices, practices, protocols, and applications must be established to be able to enumerate the previously identified vulnerabilities for that set of components, and develop the defenses against the potential attacks (such as providing patches to remove the existing vulnerabilities). To develop such desired knowhow of a legacy system, one important pre-requisite is the identification of the devices (e.g., Programmable Logic Controller (PLC), Human Machine Interface (HMI)), protocols, and applications/services on the network. This is also referred to as device fingerprinting, and if proper documentations were available, a simple solution could be to refer to those. However, per the DHS report on common cybersecurity vulnerability in ICS/SCADA [1], it turns out that in many instances, the documentation and implementation differ owing to the inadequately documented changes and updates, rendering the referral to any such documentation erroneous/incomplete. This situation is also encountered in traditional IT infrastructures, and a typical solution is to resort to network fingerprinting techniques. This can be either active or passive fingerprinting.

Active fingerprinting attempts to identify the devices on the network by actively requesting information from the devices: For example it may send probing packets to an IP address and analyze any response [2]. One may attempt to implement such an active approach for an ICS/SCADA network, but it raises two concerning issues. Firstly, it may add to the traffic load of the network, whereas most ICS/SCADA systems have congestion constraints with strict latency requirements, and their violation may raise certain safety issues. Secondly, some legacy devices on an ICS/SCADA system are not designed to robustly support any unsupported packets, and their introduction may set the device into an unpredictable state. These concerns make the active fingerprenting for ICS/SCADA systems circumspect, and so not preferred by ICS/SCADA users, as applying it requires further analysis to assess the impacts on the systems.

Passive fingerprinting uses a network sniffer to capture traffic already generated by the system devices, and analyzes this traffic to identify the devices. For passive fingerprinting to function appropriately, it is pertinent to define the distinguishing features in the network traffic to facilitate device identity. IT infrastructure network traffics have known distinguishable

978-3-903176-15-7 © 2019 IFIP

features that can be used to identify the network devices [3]. In an ICS/SCADA, a network does not typically provide *explicit* signatures for a passive fingerprinting, as its equipment are often simple and unable to offer extra-functional services. The passive fingerprinting task can be approached by first inferring the device type, that can reduce the amount of features needed to identify the devices in the network and thereby improve its accuracy. We use the communications-pattern to first identify the device hierarchy level to reveal its type, and next use the passive fingerprinting for an enhanced device recognition.

ICS/SCADA systems have specific network topology requirements, two of which are very useful for our approach. (1) Each device is only allowed to communicate with pre-programmed devices. (2) The devices follow a chain-of-command based communication-pattern to prevent any command conflict. Such topological information about communication can be used to identify the control hierarchy level of a device within that chain-of-command, where the levels of the ICS/SCADA hierarchy help identify a device type (SCADA station, HMI, PLC, etc.).

Accordingly, we introduce a two-step hybrid approach that uses both the communication-patterns and device-fingerprints (constructed through passive fingerprinting techniques). Our two-step algorithm first uses the network traffic to explore the ICS/SCADA communication-patterns and identify the control hierarchy, and next a fingerprint for each device obtained from its communication packets and is compared with the learned ones in the reference database (explained in next sentence) to determine the exact model details of the device. To learn the reference database, the traffic of a network with known devices is gathered and analyzed to identify the device specific values in certain fields of the communication packets.

We implemented this hybrid device recognition approach in Python, and tested it on traffic data from a water treatment testbed, reported in [4]. Our reference fingerprint database was constructed from the traffic data reported in [4]–[6]. [4] described a water treatment system controlled by Allen Bradly PLCs, HMI, and SCADA station. [5] presented a sub system consisting of Direct Logic 205, Siemens S7 1200, and a SCADA station. [6] provided one-to-one communication between a SCADA station and Siemens S7 1500, S7 1200, S7 300, and S7 400 PLCs. In other words, our reference dataset represents a variety of ICS/SCADA devices from different vendors to be able to validate the generality of our approach.

The main contributions of the work presented here are:

1) To the best of our knowledge, for the first time, a hybrid communication-pattern and passive-fingerprinting approach is proposed to identify the ICS/SCADA devices' type, manufacturer, and model.
2) To the best of our knowledge, for the first time, the ICS/SCADA communication-pattern is used to identify the device control hierarchy level, and next to determine the device type.
3) The paper identified a set of features in the communication packets that can be used to distinguish among the devices based on their control hierarchy level, while not relying on the availability of special packets such as the SYN packets.
4) We present a software implementation, and its validation using a real-world water treatment CPS example from iTrust [4].

The remainder of this paper is structured as follows: Section I-A reviews the related work. Section II describes our hybrid approach for device recognition. Section III describes the CPS SCADA Water case study and the experimental results. Section IV summarizes and presents certain future directions.

A. *Prior work on ICS/SCADA Recognition*

A limited number of studies exist that proposed approaches to identify the ICS/SCADA components. [7] proposed identifying the devices based on specific SCADA protocol functions (e.g., in Modbus protocol there is a function that returns the device information). Such approaches are limited to the specific protocol used, and do not generalize. The SCADA search engine Shadon [8], that can discover the devices connected to the internet, has inspired works as [9] and [10] to use port scanning methods to identify SCADA devices. However, these approaches employ active fingerprinting, and if applied without proper study and preparation, can interrupt and even "freeze" some devices, as mentioned earlier.

[11] proposed the use of TCP/IP information to construct a device fingerprint. [12] generated a device fingerprint based on the response time. This approach is only device-specific, and not device-model-specific, thus offering limited resolution of devices. [13] identified the type of a device based on the ICS/SCADA communication-patterns, but is limited to two layer systems. The ICS/SCADA device fingerprinting problem was also considered in [3], where the author explored the existing approaches, without proposing a new one, the paper concluded that current tools are not fully compatible with ICS/SCADA systems. [14] evaluated Shadon search engine and concluded that Shadon search engine can be a threat to ICS/SCADA systems as it can identify Internet-facing industrial control devices; The paper did propose a mitigation technique to protect from Shadon, and did not propose a new device recognition tool. In this paper, we improve the reliability and the accuracy of passive device recognition by proposing an original communication-pattern and TCP/IP passive-fingerprinting hybrid approach.

II. Proposed Hybrid Approach to Device Recognition

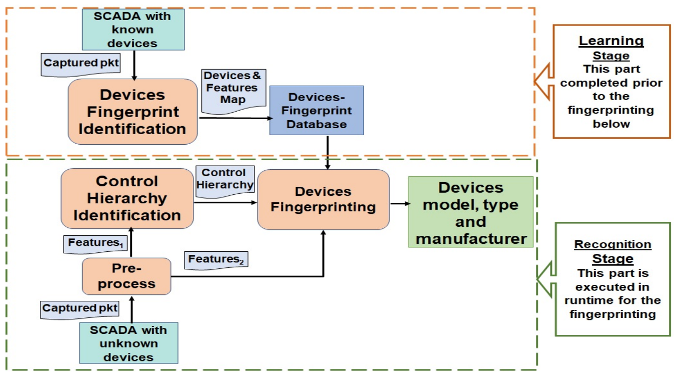

SCADA with known devices
Captured pkt
Devices Fingerprint Identification
Devices & Features Map
Devices-Fingerprint Database
Learning Stage
This part completed prior to the fingerprinting below
Control Hierarchy Identification
Control Hierarchy
Devices Fingerprinting
Devices model, type and manufacturer
Features$_1$
Features$_2$
Pre-process
Captured pkt
SCADA with unknown devices
Recognition Stage
This part is executed in runtime for the fingerprinting

Figure 1: Devices Recognition Hybrid Approach

Figure 1 depicts the architecture of the proposed hybrid approach for device recognition in an ICS/SCADA network. The approach is based on the TCP/IP passive fingerprinting where the communication packets are captured for analysis and device recognition. This is a two-stage process: the first stage is a learning stage where a reference fingerprints database is created from an ICS/SCADA system containing a known set of devices, using the identified values of certain features in the communication packets of the devices. Creating such a database requires identifying the reliable distinguishable features in the captured TCP/IP packets that can be used to generate a distinguishable fingerprint for each device. The second stage is a recognition stage where packets captured from a SCADA system with unknown devices are processed, and passed into a two-step algorithm: a control hierarchy identification step and a device fingerprinting step.

ICS/SCADA systems control hierarchy follows the standard architecture shown in Figure 2 [15]. The hierarchy is imposed to prevent conflicts in the control commands, requiring each node in the system to communicate only with its direct parent, its immediate children, or with nodes at the same level. This characteristic can be used to identify the ICS/SCADA devices' level using our control hierarchy identification algorithm (see Algorithm 1 in Section II-B). Knowing the device control hierarchy level, allows us to infer the device type (e.g., local controller level is a PLC, plant supervisory controller level is either HMI or SCADA station, production coordinator controller level and control center level are SCADA stations).

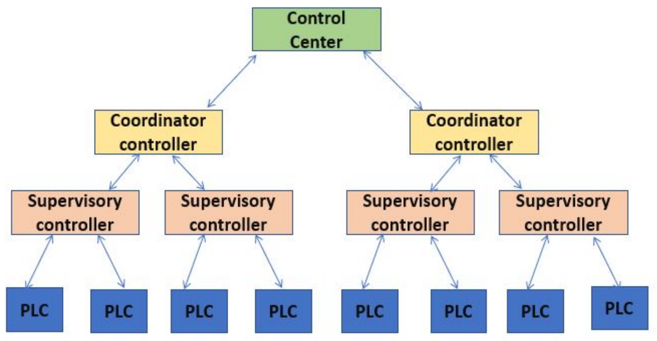


Figure 2: Standard ICS/SCADA Control Hierarchy

The above step of control hierarchy level identification is useful when ICS/SCADA devices in the same system are from the same vendors and use same programmer protocols. In such cases, different types of ICS/SCADA devices have similar TCP/IP communication packet features (Section II-A gives an example of such a case), making it challenging to distinguish devices based solely on communication packets. However, by first inferring their control hierarchy, we can identify a device type (PLC vs. HMI), and this together with the distinguishable features of the communications packets (the second step) then further pinpoints the device.

The packet processing in the recognition stage is simply to extract the features from the TCP/IP packets and store those into two matrices: $Features_1$ and $Features_2$. $Features_1$ has two columns for each packet and the total number of rows equals the number of packets captured. Column 1 records the source of the packet and column 2 records the destination. $Features_2$ has 3 columns for each packet captured, and again the total number of rows is the number of packets captured. Each column records a distinguishable packet feature, described below. To be able to fully recognize all the devices, the captured packets need to span all the devices in the ICS/SCADA network, and we assume that to be the case.

A. Learning Stage: Building Reference Database

The first stage in device recognition is to build a reference fingerprints database by analyzing the captured packet from an ICS/SCADA network with *known* devices, and identifying the values of certain features in the communication packets that are unique to the devices. To identify such distinguishable features in the captured network traffic, we examined data generated from several different ICS/SCADA devices Allen Bradley Logix 1765 PLC, Siemens S1200 PLC, Siemens S1500 PLC, Siemens S7 300 PLC, Siemens S7 400, and DirectLogic 205 PLC, reported in [4]–[6]. We examined a total of 20 TCP/IP packet features including: frame length, vendor MAC ID, TCP-segment length, IP identification numbers, Total IP packet length, time to live, and tcp-window size. These 7 features have been plotted below in Figure 3 to illustrate our fingerprinting approach. Among these 7 features 3 were found to be unique to a device. These 3 features are described in bit more detail next.

When a packet is generated, the source of the packet specifies the duration a packet remains valid, called time-to-live, and denoted (TTL). The TTL values for the packets from the same source/model are always the same, and so serve as one distinguishing feature. Another feature that a packet source specifies is the packet ID, that it increases in the same fixed increments, and that increment amount is different for different source/model. Thirdly, each generated packet also specifies the source MAC address, in which first four bytes form the vendor ID. The vendor ID in the MAC address can be used to identify the device manufacturer using the databases that enlist the devices' vendors and their MAC vendor IDs (for instance the OUI Lookup Tool from Wireshark [16]). Generally, vendors use more than one MAC vendor ID, specific to a factory or manufacturing time period. This can be used to further differentiate among the devices from the same vendor. An example for this is Siemens S1200 (by Siemens Numerical Control Ltd) versus S1500 (manufactured by Siemens AG); S1200 MAC vendor ID is 001C06 while S1500 MAC vendor ID is 001B1B.

The analysis results of packet feature values for different packets, and of different devices, are plotted in Figure 3, where each diagram is for a single device, and in each diagram, the x-axis is discrete corresponding to the 7 features mentioned above, and the y-axis is also discrete showing the values of the packet-features (encoded as numeric values). Eg, packets from the device S7 1500 are plotted in the first diagram. It can be seen that several of the ICS/SCADA packet features from a same device vary from packet-to-packet, except for the 3 distinguishable features mentioned above that are constant, and turn out to be adequate to differentiate among the 6 different PLCs. These 3 features are: Time To Live

(TTL), the difference in the IP.IDs of two consecutive packets (which we denote as, $IP.ID_{diff}$), and the vendor MAC ID ($Vendor_MAC$). Note given that TTL, $Vendor_{MAC}$, and $IP.ID_{Diff}$ can take 255 (1 byte), 4.294×10^9 (4 bytes), and 65.535×10^3 (2 bytes), values respectively. Thus using these 3 features, one can potentially distinguish a total of $255 \times (4.294 \times 10^9) \times (65.535 \times 10^3) = 7.176 \times 10^{16}$ number of devices.

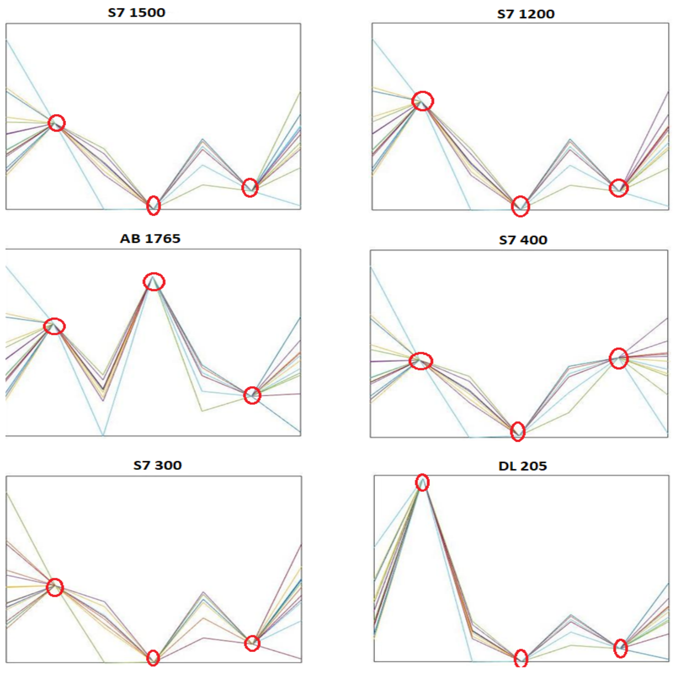


Figure 3: PLCs Data Analysis

Table I summarizes the results from our analysis of the communication packets and consolidating those into the 3 distinguishing features TTL, $IP.ID_{diff}$, and $Vendor_{MAC}$, that can be used for look up to identify the device manufacturer and model. This table reports the communications-pattern based reference database created during the learning stage of our approach for identifying different ICS/SCADA devices.

SCADA/ICS device	TTL	$IP.ID_{diff}$	$Vendor_{MAC}$
Logix 1765 PLC	64	256	001d9c
Direct Logic 205	32	1	00E062
S-1200	30	1	001C06
S-1500	30	1	001B1B
S7-300	30	1	000e8c
S7-400	128	3	000e8c

Table I: ICS/SCADA Distinguishable Features Summary

Note while the above 3 packet features are constant across the packets from the same device, there can still exist more than one device with identical set of 3 features. An example pair is the Logix 1765 PLC versus the Allen Bradly HMI; the two devices are identical with respect to the aforementioned 3 TCP/IP features as shown in Figure 4. However, such devices are not placed at the same control hierarchy, and so we use the extra information about the control hierarchy to further pin-point a device.

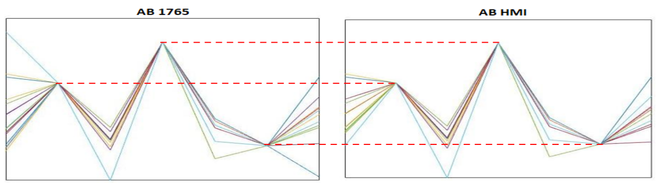


Figure 4: Logix 1765 PLC and the Allen Bradly HMI Data Analysis

B. Recognition Stage

This stage is used to recognize the unknown devices in a SCADA system by monitoring the network communications packets in the run-time, by first deciphering the control hierarchy, and next utilizing the packet features of Table 1 to identify the devices on the network.

To maintain a high level of reliability and safety, the ICS/SCADA systems architecture and communication follow the IEC 6113 [17] and the IEC 62264 [18] standards. Using such standardized control hierarchy architecture as a basis, we identify the ICS/SCADA devices' level as formulated in Algorithm 1 below.

The algorithm proceeds in bottom-up fashion, by identifying the bottom-most (also, called level-0) devices of the control hierarchy, and once the devices of a level have been identified, the devices of a next level can be identified as the destinations of the former. To identify the level-0 devices, first the devices with identical destinations are grouped into a same group. Then the level-0 devices correspond to the members of those groups whose destinations are not larger than those of any other groups. Once the level-0 devices are recognized, finding the higher levels is easier: Given level-k devices for some $k \geq 0$, their set of destination devices, that do not belong to any lower levels, form the level-$(k+1)$ devices. The algorithm terminates when all the devices are already accounted for in one of the levels.

Here we use the example in Figure 5 as a running example to illustrate these ideas. Initially, using the $Features_1$ data, that contains the list of all source-destination pairs, we compute for each source node s, its set of destination devices D_s. In Figure 5 for example, $D_{s_1} = \{s_1, s_2, s_3, s_6, s_7\}$.

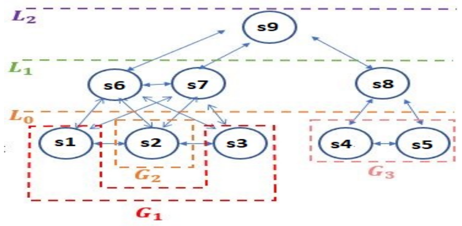


Figure 5: Control Hierarchy Identification example

Next we identify the sources that are at the bottom of the control hierarchy. For this, the sources with *identical destinations* are grouped together into their own single groups.

Then the sources in those groups are in the bottom of control hierarchy if their destinations are no larger than those of any other groups. For example, since $D_{s_1} = D_{s_3}$, we group those into the group, $G_1 = \{s_1, s_3\}$. Similarly, $G_2 = \{s_2\}$, $G_3 = \{s_4, s_5\}$, $G_4 = \{s_9\}$, etc. Next, the groups whose destinations are not larger than any of the other groups' destinations are identified. So for example, the destinations of G_1 and G_2 are smaller than those of group $G_5 = \{s_6, s_7\}$ whose destinations are $D_{s_6} = D_{s_7} = \{s_1, s_2, s_3, s_6, s_7, s_9\}$, whereas the destinations of G_3 includes s_8 that is not a destination for G_1 or G_2, etc. Similarly, the destination of G_3 namely, $\{s_4, s_5, s_8\}$, are smaller than the destination of G_6, namely $\{s_4, s_5, s_8, s_9\}$, etc. With such grouping and destination comparison, the sources in G_1, G_2, G_3, are identified to be the lowest level groups in the control hierarchy, and this is recorded in $L_0 = G_1 \cup G_2 \cup G_3$.

Once the level-k sources in L_k get recorded for any $k \geq 0$, the level-$(k+1)$ sources in L_{k+1} are identified as the union of those groups which contain a destination of a level-k source, and which do not belong to the level-k or lower. So for example, $L_1 = G_5 \cup G_6$, and next $L_2 = G_4$. The above steps are formalized in the following algorithm.

Algorithm 1 Control Hierarchy Identification Algorithm

```
1: Input: Set of packets
2: Output: sources S, control levels $L$, control level
   mapping $C : S \rightarrow L$ that maps each source $s \in S$
   to its control level $C(s) \in L$.
3: main
4: for each source $s$, add it to a set $S$, and also construct its
   destination $D_s$
5: compare $D_s$ for all $s \in S$
6: group sources with same destinations into groups $G_i$'s
7: $\forall i : G_i \subseteq L_0$ if $\forall j : G_j \not\subset G_i$
8: $\forall i, \forall k \geq 0$: $G_i \subseteq L_k + 1$ if $(\exists s \in L_k : D_s \subseteq G_i) \wedge (\nexists l \leq$
   $k : G_i \subseteq L_l)$
9: Terminate when all $s \in S$ are mapped to some level $k \geq 0$.
```

Once the control hierarchy levels have been identified using the above algorithm, the TCP/IP packet features (TTL, $IP.ID_{diff}$, $Vendor_{MAC}$) stored in the $Features_2$ dataset are compared to the reference database to identify the device manufacturer and model.

III. A CPS Case Study: Water Treatment CPS

To demonstrate the applicability of our algorithm to a real-world cyber-physical system (CPS), a water treatment system was chosen as a case-study (Figure 6). The system is based on a fully-functional water treatment CPS testbed in the iTrust laboratory [4]. The system consists of Programmable Logic Controllers (PLCs), Human Machine Interfaces (HMIs), Supervisory Control and Data Acquisition (SCADA) workstation, and a SCADA Server. The 6 PLCs control the 6 stages of water treatment, while the HMI and the SCADA station monitor and coordinate between the PLCs. The SCADA station reports to and coordinates with the SCADA server. These devices and their control hierarchy are shown in Figure 6.

The Secure Water Treatment testbed dataset comprises of data from 11 days of continuous operation. 7 days worth of

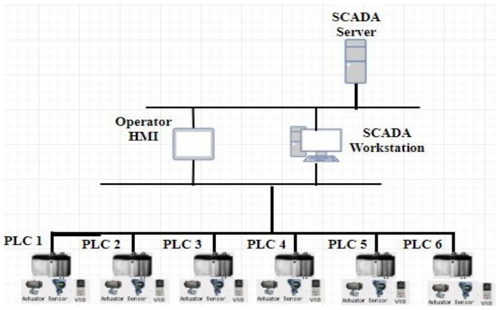


Figure 6: SCADA Architecture of the Water Treatment System

data was collected under normal operation, while 4 days worth of data was collected under attack scenarios. For the device recognition purposes, only one full control cycle of data is needed, and so for our purposes the network traffic for 4 hours of normal operation was sufficient, and was analyzed. The information in the pcap captured files were extracted and stored in CSV files using Tshark [19].

Initially, a preprocessing step was applied to the collected packets to obtain the $Features_1$ and $Features_2$ matrices. The algorithm started by enumerating the set of sources S and the destinations D_s for each source $s \in S$. 9 members were found as sources. Next, $s = 192.168.1.10$ was randomly selected, and its destination set was compared to the destinations of other devices. As a result, 192.168.1.10, 192.168.1.20, 192.168.1.30, 192.168.1.40, 192.168.1.50, and 192.168.1.60 were grouped in G_1. Next, $s = 192.168.1.100$ was selected, and comparing its destinations to those of the other remaining devices resulted in $G_2 = \{192.168.1.200, 192.168.100\}$. Finaly, $s = 192.168.1.201$ was selected; it being the last source in S, it was finalized that $G_3 = \{192.168.1.201\}$. G_1 is the only group that does not contain the other groups as a subset, so $L_0 = G_1$. G_2 contains the destination for G_1 devices, so $L_1 = G_2$. Similarly, G_3 contains the destinations for G_2 devices, and so $L_2 = G_3$. At this point all sources in S have been mapped to a control level, so the algorithm terminated. The result from our Python encoding of Algorithm 1 is shown in Figure 7.

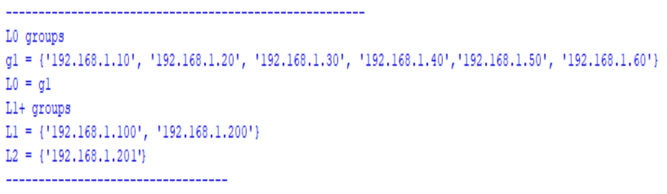
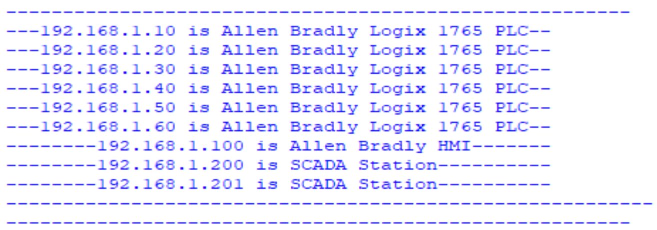

```
--------------------------------------------------------
L0 groups
g1 = {'192.168.1.10', '192.168.1.20', '192.168.1.30', '192.168.1.40','192.168.1.50', '192.168.1.60'}
L0 = g1
L1+ groups
L1 = {'192.168.1.100', '192.168.1.200'}
L2 = {'192.168.1.201'}
-----------------------------------
```

Figure 7: Water Treatment CPS control hierarchy identification output

Based on the control hierarchy identification outcomes, the bottom level devices, 192.168.1.10, 192.168.1.20, 192.168.1.30, 192.168.1.40, 192.168.1.50, and 192.168.1.60 can be identified to be PLCs. The level-1 devices, 192.168.1.100 and 192.168.1.201 can either be a SCADA station or a HMI device. Finally, the top level device, 192.168.1.200 is identified as a SCADA station. Subsequently, the vendor ID was used to distinguish between the HMI and the SCADA station: the vendor ID for both 192.168.1.201 and 192.168.1.200 were Microsoft, while the vendor ID for 192.168.1.100 was Rockwell Automation. Based on this, 192.168.1.201 and 192.168.1.200 were identified to be SCADA stations, while 192.168.1.100 was identified as an Allen Bradly HMI. The device fingerprinting step constructed a fingerprint for each PLC in the form of $Feature_2$ matrix, which was then compared against the lookup table (Table I) to identify their manufacturer and model. This step then revealed that the PLCs are the Allen Bradly Logix 1765 PLC. A log of these results is shown in Figure 8.



```
-------------------------------------------------------
---192.168.1.10 is Allen Bradly Logix 1765 PLC--
---192.168.1.20 is Allen Bradly Logix 1765 PLC--
---192.168.1.30 is Allen Bradly Logix 1765 PLC--
---192.168.1.40 is Allen Bradly Logix 1765 PLC--
---192.168.1.50 is Allen Bradly Logix 1765 PLC--
---192.168.1.60 is Allen Bradly Logix 1765 PLC--
--------192.168.1.100 is Allen Bradly HMI-------
--------192.168.1.200 is SCADA Station----------
--------192.168.1.201 is SCADA Station----------
---------------------------------------------------------
-------------------------------------------------------
```

Figure 8: Water Treatment CPS device recognition output

IV. Concluding Remarks

We presented a first-of-a-kind ICS/SCADA systems device recognition approach based on passive fingerprinting of network data, where a two-stage process was proposed. The first stage is a learning stage where a reference fingerprints database was created. The second stage is a two-step online stage algorithm: the first step identifies the control hierarchy based on the ICS/SCADA communication patterns; the second step identifies a device by creating its fingerprint and comparing it to the reference database of the first stage to identify the device manufacturer and model. Identifying the control hierarchy enhances the device recognition capability since it discriminates further, beyond the reference fingerprinting based discrimination. Also, we fully implemented our approach in Python and demonstrated the validity of the proposed solution through a real-life water treatment SCADA system.

Discovering the devices included in ICS/SCADA systems is an essential first step toward improving their overall cybersecurity. A next step would be to implement the proposed approach along with systems security analysis and mitigation tools, to enhance their overall security and defense strategies against potential cyberattacks.

Acknowledgment

The research was supported in part by the National Science Foundation under the grants, NSF-CCF-1331390, NSF-ECCS-1509420, and NSF-IIP-1602089, and Rockwell Collins through the NSF-IUCRC, S2ERC. The authors thank Yazan Okasha for the help with the implementation of the algorithm, and iTrust team for sharing the Water Treatment System SCADA data, that we used for validating our proposed approach.

References

[1] T. Nelso and M. Chaffin, "Common cybersecurity vulnerabilities in industrial control systems," *Control Systems Security Program. Washington DC: Department of Homeland Security (DHS), National Cyber Security Division*, 2011.

[2] S. A. Oliva and B. Crowe, "Network system and method for automatic discovery of topology using overhead bandwidth," Nov. 25 2003, uS Patent 6,654,802.

[3] M. Caselli, D. Hadžiosmanović, E. Zambon, and F. Kargl, "On the feasibility of device fingerprinting in industrial control systems," in *International Workshop on Critical Information Infrastructures Security*. Springer, 2013, pp. 155–166.

[4] iTrust. (2018) Secure water treatment tesetbed. Accessed:9/4/2018. [Online]. Available: https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/

[5] L. Hansson. (2018) Capture files from 4sics geek lounge. Accessed:9/4/2018. [Online]. Available: https://www.netresec.com/index.ashx?page=PCAP4SICS

[6] T. Yardley. (2018) Ics pcaps. Accessed:9/4/2018. [Online]. Available: https://github.com/ITI/ICS-Security-Tools/tree/master/pcaps

[7] S. Gordeychik, "Scada strangelove or: How i learned to start worrying and love nuclear plants," 2013.

[8] J. C. Matherly, "Shodan the computer search engine," *Available at [Online]: http://www. shodanhq. com/help*, 2009.

[9] J. B. Radvanovsky and J. Brodsky, "Project shine: What we discovered and why you should care," *10th Sans ICS Security Summit, Orlando, FL*, 2015.

[10] T. Kiravuo, S. Tiilikainen, M. Särelä, and J. Manner, "Peeking under the skirts of a nation: finding ics vulnerabilities in the critical digital infrastructure," in *European Conference on Cyber Warfare and Security*. Academic Conferences International Limited, 2015, p. 137.

[11] J. François, H. Abdelnur, R. State, and O. Festor, "Ptf: Passive temporal fingerprinting," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*. IEEE, 2011, pp. 289–296.

[12] A. R. Beyah, I. David Formby, and P. Srinivasan, "Device fingerprinting for cyber-physical systems," Feb. 15 2018, uS Patent App. 15/556,136.

[13] S. Jeon, J.-H. Yun, S. Choi, and W.-N. Kim, "Passive fingerprinting of scada in critical infrastructure network without deep packet inspection," *arXiv preprint arXiv:1608.07679*, 2016.

[14] R. Bodenheim, J. Butts, S. Dunlap, and B. Mullins, "Evaluation of the ability of the shodan search engine to identify internet-facing industrial control devices," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 2, pp. 114–123, 2014.

[15] T. Bangemann, S. Karnouskos, R. Camp, O. Carlsson, M. Riedl, S. McLeod, R. Harrison, A. W. Colombo, and P. Stluka, "State of the art in industrial automation," in *Industrial Cloud-Based Cyber-Physical Systems*. Springer, 2014, pp. 23–47.

[16] Wireshark. (2018) Oui lookup tool. Accessed:9/4/2018. [Online]. Available: https://www.wireshark.org/tools/oui-lookup.html

[17] PLCopen. (2013) International standard iec 61131 applies to programmable controllers (plc). Accessed:9/4/2018. [Online]. Available: http://www.plcopen.org/pages/tc1_standards/iec61131-1/

[18] I. E. C. (IEC). (2016) Iec 62264-2 enterprise-control system integration. Accessed:9/4/2018. [Online]. Available: http://www.plcopen.org/pages/tc1_standards/iec61131-1/

[19] Wireshark. (2018) tshark dump and analyze network traffic. Accessed:9/4/2018. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html