Characterizing Industrial Control System Devices on the Internet

Xuan Feng*†, Qiang Li†, Haining Wang ‡, Limin Sun*
* Institute of Information Engineering, Chinese Academy of Sciences, China
† School of Computer and Information Technology, Beijing Jiaotong University, China
‡ Department of Electrical and Computer Engineering, University of Delaware, USA

***Abstract*—Industrial control system (ICS) devices with IP addresses are accessible on the Internet and play a crucial role for critical infrastructures like power grid. However, there is a lack of deep understanding of these devices' characteristics in the cyberspace. In this paper, we take a first step in this direction by investigating these accessible industrial devices on the Internet. Because of critical nature of industrial control systems, the detection of online ICS devices should be done in a real-time and non-intrusive manner. Thus, we first analyze 17 industrial protocols widely used in industrial control systems, and train a probability model through the learning algorithm to improve detection accuracy. Then, we discover online ICS devices in the IPv4 space while reducing the noise of industrial honeypots. To observe the dynamics of ICS devices in a relatively long run, we have deployed our discovery system on Amazon EC2 and detected online ICS devices in the whole IPv4 space for eight times from August 2015 to March 2016. Based on the ICS device data collection, we conduct a comprehensive data analysis to characterize the usage of ICS devices, especially in the answer to the following three questions: (1) what are the distribution features of ICS devices, (2) who use these ICS devices, and (3) what are the functions of these ICS devices.**

I. Introduction

The number of industrial control system (ICS) devices with computing and communication capabilities is rising rapidly, and they are crucial for infrastructure-critical systems, such as power, oil, and gas pipelines, water distribution, and wastewater collection systems. Supervisory control and data acquisition (SCADA) [1] is often used to control remote ICS devices with coded signals, and these ICS devices are typically computer-based systems with access to the Internet. On one hand, the online ICS devices operate over proprietary protocols and are distributed throughout the whole IPv4 space (four billion IP addresses), and using traditional way to discover all these devices will cost few months. However, searching, characterizing and protecting the exposed ICS devices need to be done in a timely manner. On the other hand, characterizing the usage of these online ICS devices can help us to better manage and protect them. Such a deep understanding of these online ICS devices is highly desirable for the interests of infrastructure-critical systems, but is unfortunately still missing. In this paper, we take a first step to address this challenge and shed light on the usage of online ICS devices.

† Qiang Li is the corresponding author.

We first analyze 17 industrial protocols [2] that are widely adopted in industrial control systems. We send common ICS protocol requests to remote networks and determine whether it runs the ICS protocol. While the idea of ICS measurement seems simple, directly detecting ICS devices in practice is faced with three major challenges. First, one may concern that this measurement could cause interference on the normal operations to ICS devices. Second, the four billion IP addresses is an enormous space to search for ICS devices. As each SCADA protocol operates as the application layer protocol over TCP/IP, there is a need to build standardized and recognized communications to verify ICS devices. This also induces significant time latency in device verification. Third, there is an increasing trend of industrial honeypot deployment because many government, enterprisers, and research institutes attempt to track attack data for countering potential threats.

To tackle these challenges, we propose a real-time, non-intrusive device discovery approach. First, we analyze 17 different kinds of industrial control protocols to select and minimize the probes. Second, we propose a learning model to determine the probability of an ICS honeypot and a heuristic algorithm to verify it with the least number of packets. Third, we do online detection of ICS devices based on the offline training and analysis result. To discover live ICS candidates, we use one stateless packet to filter out unqualified hosts first. If the probability of a remote host being ICS is higher than a pre-specific value, we further use a heuristic algorithm to identify ICS devices. For ethic consideration, we send packets to a remote network by performing standards-compliant handshakes without any malformed payload code. We also clarify the purpose of our measurement by adding reverse DNS entries for our measurement server [3], running a simple web page on port 80 that describes the goals of this research, what data we are collecting, and how to contact us to exclude from our search if remote administrators are unwilling to have their devices discovered.

Based on the protocol analysis and the proposed device discovery approach, we develop our ICS device discovery system using go scripts [4] and validate its effectiveness through real-world experiments. The experimental results demonstrate that our system are able to discover ICS devices with high precision and recall rates. To observe the dynamics of ICS devices in a relatively long run, we deploy the device discovery system in

TABLE I: Category of industrial control protocols

Category	ICS Protocols
TCP-Based	OMRON FINS, HART-IP, Siemens S7, Mddbus, IEC 104, DNP3, EtherNet/IP, Tridium Niagara Fox, PCWorx, ProConOS, CodeSys, Red Lion Crimson V3, General Electric SRTP, CSPV4, Automatic Tank Gauge
UDP-Based	BACnet, OMRON FINS, MELSEC-Q, HART-IP

Amazon EC2 [5] to search the entire IP addresses and discover these online ICS devices for eight times from August 2015 to March 2016. Each time it takes us about 20 hours for the completion of device discovery. Based the collected dataset, we conduct a comprehensive data analysis to characterize the usage of ICS devices on the Internet. In particular, we attempt to answer the following three questions: (1) what are the distribution features of ICS devices, (2) who use these ICS devices, and (3) what are the functions of these ICS devices.

Overall, the major contributions of this work are summarized as follows.

- We have investigated 17 industrial control protocols running on ICS devices and proposed an effective discovery mechanism for searching online ICS devices in the entire IPv4 space.
- We have implemented the discovery mechanism with go scripts and evaluated it in the real-world experiments. We have further deployed it on Amazon EC2 and conducted ICS device discovery for eight times over eight months.
- We have analyzed the collected dataset of ICS devices all over the Internet and characterize the usage and distribution features of these devices in a comprehensive manner.

The remainder of the paper is structured as follows. Section 2 presents our analysis of 17 industrial control protocols. Section 3 describes our ICS device discovery approach and its evaluation. Section 4 details our data analysis and characterization based on the device discovery experiments over eight months. Section 5 surveys related work, and finally, Section 6 concludes.

II. Industrial Control Protocols

In this section, we analyze 17 industrial protocols running on ICS devices, formalize their functions, and show how to discover ICS devices using these protocols.

A. Industrial Control Protocol Analysis

ICS devices typically run a variety of industrial protocols in the application layer. There is abundant information encapsulated in the packet header of these protocols, such as device vendor, type and function. We can use such information to identify ICS devices by analyzing the contents. Table I shows 17 typical industrial protocols that are widely adopted in industrial control systems. By exploiting these 17 protocols, we can find most of devices exposed on the Internet. These industrial control protocols run as application-layer services over standardized TCP/IP protocols. Industrial devices use them to send configurations and control commands for interacting with one another.

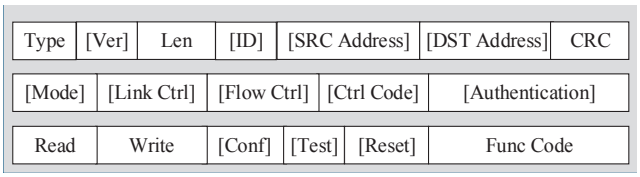
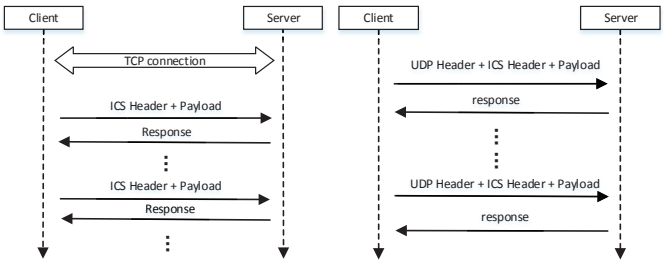


Fig. 1: Functions of industrial control protocol.

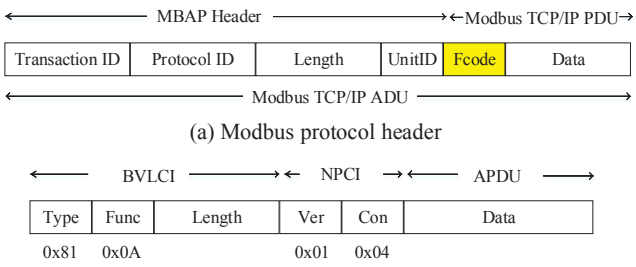


(a) TCP-based. (b) UDP-based.

Fig. 2: ICS communication interactions between the detector and the device based on TCP and UDP.

We analyze these 17 industrial control protocols (Table I). We draw up their most functions used in industrial control protocols. Figure 1 shows the formalized fields in their header format, some optional fields we marked by "[]". Their functions are encapsulated in these fields, and they can be obtained by extracting their response packets from the industrial protocols in the application layer. The first line of header fields indicate the industrial control protocol type, length, version, checksum, as well as their source and destination operation addresses. Some ICS protocols may have several fields related to congestion and flow control functions. The authentication and encryption fields are also needed to secure the communication, but they are only supported by a few ICS protocols. The most important and indispensable field for a industrial control protocol is the application protocol data unit, which sends signal code to acquire the device's status and control industrial devices (operation, start, stop, reset, etc). In addition to a device's status, we also can read some detailed information like device's registration based on the signal code. Note that there are different forms in these fields, and we extract these fields for identifying ICS devices by manually dismantling their signal code.

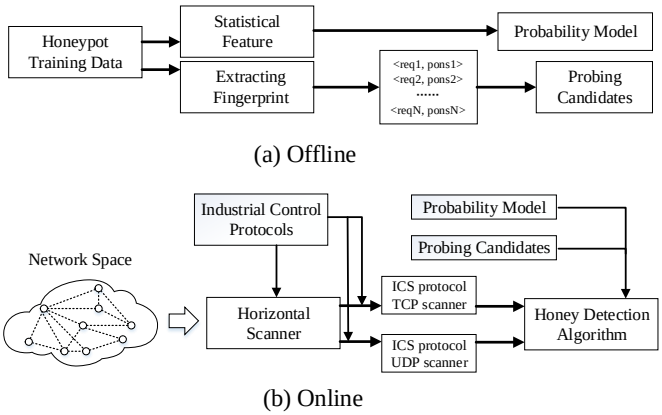
Industrial protocols either operate over TCP or UDP. For TCP-based, Figure 2a shows devices running a TCP-based protocol first establish a standardized TCP connection, and then transmit a probe packet by carrying different kinds of payload. Based on the response to the probe packet, we can identify and enumerate the respondent device's detailed information. Although 17 industrial protocols have different packet structures to encapsulate their payloads, we have stored their signal code in a file for matching comparison. Therefore,



(a) Modbus protocol header

(b) BACnet protocol header

Fig. 3: Two ICS protocol examples.



(a) Offline

(b) Online

Fig. 4: Architecture overview.

we send a probe packet to a remote host with encapsulated specific payload. If the host responds to the request, we extract the response and match it with pre-specific values. If there is a match, we identify it as an ICS device. For UDP-based, Figure 2b shows devices running UDP-based protocol for determining whether a host is an ICS device. Since it is UDP-based, there is no need to establish a connection. We encapsulate all signal code into the packet payload and send it to the remote host. To correctly extract device information, we send different probe packets to enumerate the detailed properties of the device. In general, regardless of transport layer protocols, we create a standardized packet with common payload and then send it to a remote host for identifying ICS devices, without including any malformed payload code.

B. Identifying ICS Devices

Due to the page limit, we only present the illustrations for two ICS protocols: Modbus based on TCP and BACnet based on UDP.

In Modbus, devices running the protocols first need to build the TCP connection before exchanging data with one another. Every time devices transmit data, that data carries a function code, labeled as "Fcode". Figure 3a shows this protocol structure, where "Fcode" occupies 1 byte, specifying a particular function to regulate the purpose of the data, such as 04 (Read Input Register) and 03 (Read Holding Registers). If we send a specific packet satisfying the Modbus protocol standard, we can identify whether the host is the physical device running Modbus from the response. Therefore, we send a probe packet with a payload encapsulating the function code. If the host responds to the function code request, we label the host as a physical device running Modbus.

BACnet is used in building automation and control systems for applications such as heating, ventilating, and air-conditioning, lighting control, access control, and fire detection systems and their associated device. We first identify whether an IP connected device is running BACnet. This works by querying the device with a pre-generated BACnet message as shown in Figure 3b. The BVLCI (BACnet Virtual Link Control Information) set as "0x0A" indicates that this is an Original-Unicast-NPDU and NPCI shows that this is "data expecting a reply." Newer versions of the BACnet protocol will respond with an acknowledgement, and older versions will return a BACnet error message. Presence of either the acknowledgement or the error is sufficient to prove a BACnet capable device is at the target IP address. Then, if an acknowledgement is received, we can change the APDU fields shown in Figure 3b attempting to enumerate several BACnet properties on a responsive BACnet device.

III. Industrial Device Discovery

In this section, we first present the details on how to discover visible industrial control devices on the Internet, and then we verify our discovery approach in real-world experiments.

The overview of our measurement architecture is shown in Figure 4. It contains two parts: an offline probability training model for ICS honeypot detection and an online discovery algorithm for ICS device detection. At the offline stage (Section III-A), our goal lies in using the least number of probing packets to identify an ICS honeypot. Because sending many packets to ICS systems is an intrusive behavior, we should minimize the number of probing packets. We extract statistical features for hosts and train a probability model for every ICS host candidate. We then propose a heuristic algorithm to fingerprint an ICS honeypot based on network fingerprinting technology [6]. At the online stage (Section III-B), we levearge the probing packets to identify industrial devices illustrated in Section II. We use a horizontal scanner to choose the live hosts from 4 billion IP addresses. The horizontal scanner is based on the analysis result of industrial control protocols, where we collect the communication ports of ICS devices and generate specific packets used for probing. As industrial control protocols operate over the application layer, we use specific packets to probe these live hosts and get the detailed information of industrial control devices. After that, we adopt a honeypot detection algorithm based on the proposed probability model and probing candidates identified offline.

A. Offline Training

While we use the method in Section II to identify ICS devices, there are some that may not be real devices. The most important reason is the extensive deployment of ICS

honeypots, such as the Honeynet project [7], Digital Bond's SCADA Honeynet [8] and Conpot [9]. Because of the increasing importance of ICS devices, tracking potential attacks is of great interest to system administrators. Therefore, the number of ICS honeypots is increasing with time. Detecting honeypots is straightforward because they are simply simulations of networking services and have their own implementation details. However, using the off-shelf tools, like Nmap [10], to detect ICS honeypots creates a practical and ethical implications; ICS environments are performing time-critical functions, and it will be intrusive if the tools send too many packets. We need to detect ICS honeypots using the least number of probing packets.

To address this challenge, we propose a two-stage approach to identify a honeypot without sending many packets. In the first stage, we use the naive Bayes classifier [11] to compute the probability whether the ICS device is a honeypot. Bayes is based on the intuition that honeypot information has its own characteristics compared with that of true ICS devices, such as ISPs, owners and locations. The probability $P(y|X)$ is conditioned on several feature variables X, which have the following relation according to conditional probability:

$$P(y|X) \propto P(y)P(X|y)$$

The feature X is a vector consisting of statical information extracted from the IP address of the ICS device, denoted as $X =< x_1, x_2, ..., x_k >$, where $P(y|X)$ is the prior probability that the ICS device is a honeypot under the condition information X. For every statistical feature, we can get additional information from its IP-related or historical characteristics (x_i) without sending any packets. For example, the device's registration information (DNS, ISPs, whois, location, etc.) or the default honeypot configuration could be sourced without probes.

Bayes inference is conditionally independent. We calculate the probability $P(X|y) = \Pi_{x_i \in X} P(x_i|y)$. Each probability $P(x_i|y)$ can be directly calculated by the following:

$$p(x_i|y) = \frac{p(x_i \cap y)}{p(y)}$$

$$p(x_i \cap y) = \frac{N_{x_i \cap y}}{N_{total}}$$

Combining these equations, we can figure out the impact factors of all the characteristics ($\{P(y|x_i)\}$). Given a new host, we can determine its probability $p(y|X)$. If the value is larger than the pre-specified threshold S_{th}, we add the host into the candidate list to verify it at the next stage.

At the second stage, we verify each item in the candidate list with a honeypot fingerprint. There is a tradeoff for the honeypot fingerprint between accuracy and cost. Sending more packet would improve the accuracy of detection, but it also would affect the remote network, especially if they are ICS systems. We propose a heuristic algorithm to generate a honeypot fingerprint. Algorithm 1 shows how to generate a honeypot fingerprint. Given different types of traditional ICS honeypots $F = \{f_1, f_2, ..., f_N\}$ [12], each feature f_i includes a pair of probe and response, $\{(p_1, r_1), (p_i, r_i), ..., (p_k, r_k)\}$, where p_i is the probing packet and r_i is the response. We define the cost as the number of (packet, response) pairs. The accuracy is the value of the relative degree that is pre-define in off-shelf tools. We use the heuristic criterion $h_i = R_i/T_i$ to choose which packet to send to verify an ICS honeypot. The algorithm generates a final cost vector $H = \{h_1, h_2, ..., h_N\}$. Then we choose the lowest-cost top K features to fingerprint the honeypot due to the characteristics of ICS devices. We determine the final features F_{final} as a honeypot fingerprint to verify the ICS honeypot with the least number of packets.

Algorithm 1 Fingerprint generation.

Input: different kinds of ICS honeypot fingerprints, $F = \{f_1, ..., f_N\}$, every f_i has a accuracy R_i
Output: final fingerprint used to identify honeypots, F_{final}

1: **for** (each $f_i = \{(p_1, r_1), ..., (p_i, r_i), ..., (p_N, r_N)\}$ in F) **do**
2: $T_i = \sum_{i=1}^{N} cost(p_i, r_i)$
3: heuristic criterion: $H = \{h_1, h_2, ..., h_N\}$, $h_i = \frac{R_i}{T_i}$
4: Sort(H)
5: choose the lowest-cost top K C_i and its related f_i
6: generate final F_{final}
7: **end for**

B. Online Detection of ICS Devices

We utilize the standard application protocols to discover physical devices running industrial protocols. The online detection needs to be done in real time. In IPv4 space, there are nearly 4 billion IP addresses. As shown in Figure 2, TCP/UDP would introduce certain latencies. Moreover, for the ICS protocol, we need extra probing packets to traverse different fields to retrieve device information. Assuming that each IP address costs several seconds of latency, the time spent over 4 billion addresses will be several months, which is too long in practice. We propose an online detection algorithm, as aforementioned in Figure 4, utilizing industrial protocols to identify devices (Section II) and use the probability model and fingerprints to remove ICS honeypots (Section III-A). Our online detection algorithm is shown in Algorithm 2.

Our algorithm's input is the detection range, and the output is the ICS device list in this range. At the beginning of detection, we first randomize the IP sequence (line 1). If we simply probed every address in numerical order (e.g., 10.0.0.1, 10.0.0.2, 10.0.0.3.), it would cause a burst of consecutive packets, disturbing the remote network. They are used in many previous works [13], [14]. To avoid this, we use the alternative to IP-sequential, a completely random permutation of the address space [15] (e.g., 19.2.112.1, 150.0.0.2, 220.1.2.3.). We target uniformly and randomly over the full range and intermingle probes to many subnets, which reduces the instantaneous load on individual networks and produces an unbiased random sampling.

TABLE II: The cost and relative degree of features

Features	Cost (packet)	Relative degree
Amount of open ports	6	26/297
HTTP configuration	4	9/297
Modbus signal code	5	15/297
S7 signal code	9	15/297

To reduce measurement latency, we also apply the following two existing techniques to filter out unqualified hosts. We send one packet to every address to determine whether there is a live host (line 3). Recent research ZMap [16] suggests that one packet can cover nearly 98% of the detection range and speed up the measurement. We adopt this approach by just sending out one packet to a random address each time to discover physical devices. For each address, we do not build a state connection like a TCP connection. Instead, we use the stateless connection to send probing packets without any waiting (line 4), which can significantly speed up the measurement time.

Once we detect live hosts from the detection range, we adopt industrial control protocols to verify whether these live hosts are ICS devices. The number of candidates is greatly reduced to the order of millions compared with the initial 4 billion. For each candidate, we first conduct a three-way handshake using TCP to build a state connection or straightly seed probes using UDP. To identify industrial protocols, we send a probing packet encapsulating its standard header and common data payload. To get more information, we may seed more packets or traverse special fields to extract more information from devices. For example, to identify S7 protocol, we traverse packets for the correct value of the "DST_TSAP" field and pass through the COTP connection. We then build an S7 communication connection and gain more information from the devices.

In the last stage, we remove the detected honeypots from the candidate list. We use the probability model to calculate the honeypot probability for each ICS candidate. If its value is larger than the pre-defined threshold (line 12), we will further verify it; otherwise, we will include it to the ICS list. After filtering with the Bayes probability, the number of candidates is greatly reduced. Then we use the heuristic algorithm to verify whether the candidate is a honeypot. If it is a honeypot, we remove it from the candidate list and add it to the honeypot list. Finally, our algorithm outputs a list of ICS devices.

C. Discovery Approach Validation

First, we evaluate the performance of our proposed honeypot detection algorithm. We randomly select 617 ICS devices from the whole IPv4 space with a random permutation of the address space. We manually tag their labels among these ICS devices and find 42 ICS honeypots, which are used as our experiment dataset.

For the Bayes probability model, we collect some common features as statistical features. For examples, the host's ISP is the VPS provider, the host's ICS banner has default information, the host can get its own rDNS (reverse DNS) and geo-location. They can be obtained through IP-based online rDNS service, MaxMind's GEOIP [17], and field values of response packets.

We use the first 297 hosts as our training dataset to train our naive Bayes classifier [18], and the remaining part as the test dataset. Figure 5 shows the performance of the probability model under different values of threshold S_{th}. When S_{th} is small, the false negative rate is low but the false positive rate is high, implying that the number of ICS honeypot candidates will increase. When S_{th} becomes large, the situation is reversed. We can use S_{th} to strike a balance between the false positive rate and the false negative rate based on the actual situation.

For ICS honeypot fingerprinting, we use Algorithm 1 to generate the probing packets. Conpot [9], one typical ICS honeypot, is used to verify our approach. There are four features for the honeypot: the amount of open ports, HTTP configuration, Modbus and S7 signal code. We use 281 ICS devices and 16 ICS honeypots as the training dataset. Table II shows the cost and relative degree to extract each feature. The cost is the number of probes sent to remote devices, and the relative degree is the rate of the number of response hosts to the total number of hosts. The test dataset includes 294 ICS devices and 26 ICS honeypots. Table III shows the

Algorithm 2 Online detection of ICS devices.

Input: The list of the detection range, $list$;
Output: The list of ICS devices, $list'$;
1: Using a random algorithm to resort the $list^r = list$;
2: **for** (each IP in $list^r$) **do**
3: send one packet
4: each packet with stateless
5: add each live host into $list'$
6: **end for**
7: **for** (each IP in $list'$) **do**
8: using ICS protocols verifies it
9: add the quantified host into $list'$
10: **end for**
11: **for** (each IP in $list'$) **do**
12: **if** ($p(y_i|X) > S_{th}$) **then**
13: send packet with packets FF with algorithm 1.
14: **if** (get its responses & match the fingerprint) **then**
15: add it into $list_{honeypots}$, remove it from $list'$
16: **end if**
17: **end if**
18: **end for**

TABLE III: Comparison between our generated fingerprints and traditional fingerprints

	Cost (every host)	Accuracy
Traditional fingerprint	20+ packets	100%
Our generated fingerprint	5 packets	95.2%

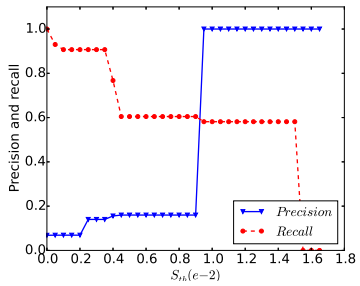


Fig. 5: Precision and recall with different S_{th}.

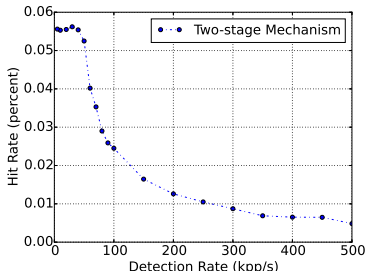


Fig. 6: The hit rate and detection rate for live host discovery.

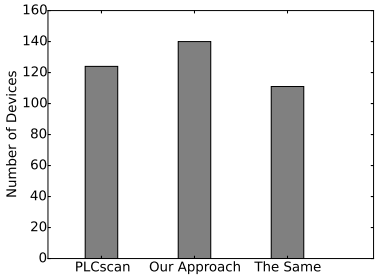


Fig. 7: Comparison of our approach with PLCscan.

performance of our generated fingerprints in the test dataset. The results show that our fingerprints can guarantee a 95.2% accuracy using only one fourth of packets for traditional fingerprinting.

Secondly, we evaluate the the performance of our online detection of ICS devices. We conduct experiments to measure the latency of our online detection approach. The detection time of our approach contains three parts: (1) discovering a live host, (2) verifying ICS protocols, and (3) determining the honeypot, denoted as $T = T_1 + T_2 + T_3$. The time cost of each part is equal to the number $*$ unit time, i.e., $T_i = N_i * u_i$. Note that the magnitudes of the numbers in the three parts are different. For instance, we detect the whole IPv4 space, in which N_1 is 40 billion, N_2 is less than hundreds of thousands, and N_3 is just thousands. Although each item in the determining-the-honeypot stage costs the largest unit of time, its total time latency is neglectful compared with the discovering-the-live-host stage. This is because we evaluate the first stage for discovering live hosts from a huge range of the IPv4 space, which dominates the majority of our detection time.

As shown in Figure 6, there is a tradeoff between the hit rate and the detection rate. The hit rate is equal to $N_{candidates}/N_{total}$, where N_{total} is the total number of IP addresses and $N_{candidates}$ is the number of responding hosts. The higher the hit rate, the less time we spend on detection. The detection rate is the speed of discovering physical devices, (i.e., the latency incurred in our Internet-wide discovery). When the detection rate is 50,000 packets per second, we can achieve a stable hit rate for discovering physical devices. Although ZMap [16] is able to scan the entire IP space under 45 minutes, the time is basically a theoretical bound and only includes host discovery. Our approach collects more information about physical devices than the survival scan by ZMap. A practical issue is that network congestion reduces the hit rate. Our suggestion is to adopt the most stable detection rate while keeping the hit rate high.

Furthermore, we evaluate the precision of our online detection approach in the ICS protocol verification stage. Because device detection occurs in the remote Internet space, there is no ground truth about a device except for its IP address. To gain this truth, we use the traditional tool PLCscan [19] to find physical devices running ICS protocols. It is a utility that was released to identify SCADA devices [1] on the network. We choose a /20 subnet with 4,096 IP addresses to detect physical devices. As shown in Figure 7, our approach is able to detect 140 industrial control devices while PLCscan can merely find 124. We compare the coverage range of the two approaches, and observe that 111 devices are the same in both scans. Using PLCscan results as the ground truth, our precision is 89.5% and recall is 79.3%, as listed in Table IV. However, our approach only takes 5 seconds, while PLCscan needs more than 1 hour. This is because PLCscan needs to traverse every IP address with a state connection and has a complicated packets-probing process of using more than a dozen of packets. By contrast, we use just one packet in the industrial protocol to verify industrial devices. Thus, our approach demonstrates efficient and non-intrusive behaviors.

TABLE IV: The precision and recall of our approach

	Condition positive	Condition negative
Test outcome positive	111	13
Test outcome negative	29	3,972

IV. Data Analysis and Characterization

We have developed a prototype of our device discovery system to detect ICS devices from all over the world by probing the entire IPv4 space. We attempt to understand the usage characteristics of ICS devices in the cyberspace and answer the following questions: "What are the distribution features of ICS devices?", "Who use ICS devices?, and "What are the functions of ICS devices?"

A. Data Collection

We have implemented 17 industrial control protocols [2] as shown in Table I. Each ICS protocol runs over the application layer, and we verify it by sending a pre-speciality data payload. The parsing scripts of industrial control protocols are implemented using the go language [4]. For TCP-based industrial control protocols, we establish three-way handshakes then communicate in the application layer and verify whether they are running the ICS protocols. For UDP-based industrial control protocols, we send pre-generated UDP packets that have

TABLE V: ICS devices discovery at the Internet scale

Begin Time	Device Count	Industrial Control Protocols
2015-08-31	23,983	Modbus,Siemens S7
2015-09-02	24,050	Modbus,Siemens S7
2015-09-05	23,956	Modbus,Siemens S7
2015-10-23	76,817	Modbus,Siemens S7, EtherNet/IP, BACnet, Tridium Niagara Fox, Crimson Red Lion
2015-10-28	20,524	Modbus
2015-10-29	20,514	Modbus
2015-11-30	21,185	Modbus
2016-03-17	141,008	BACnet, MELSEC-Q, OMRON FINS, HART-IP, Siemens S7, DNP3, EtherNet/IP, Tridium Niagara Fox, PCWorx, Red Lion Crimson V3, General Electric SRTP, Automatic Tank, Mddbus, IEC 104, ProConOS, CodeSys,CSPV4

payloads with special industrial control protocol structures. The response packets are used to match the ICS protocol structure.

In the horizontal scanning stage, we send a single TCP packet that has "SYN" filed in its header and a "NULL" data payload to determine whether a remote host is live. Note that we need to send a pre-generated UDP packet when protocols are based on UDP. By reusing the stateless connection and IP address randomization algorithms of the network scanning tool, we send a probe packet to every IP address with a certain communication port, such as "20000" and "502". If the host responds with "SYN-ACK" in the TCP header or responds to the pre-generated packet, we put this host into the set of candidates; otherwise, we discard it. The candidates are stored as a JSON file to be handled in the next phase.

In the honeypot detection stage, we calculate the probability of each live host from these statistical features. If the probability is larger than the pre-threshold value, we remove it from the ICS list and add it to the honeypot list. We use the heuristic algorithm to send probing packets to verify whether it is a honeypot. Finally, the detailed information about physical devices is also stored in a JSON file. To speed up these processes, we use a pipeline to connect the two stages to guarantee that every host in the candidate list can be identified by the go scripts.

We have deployed our prototype system running on a Cloud computing server inside Amazon EC2 [5]. The server runs Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-48-generic x86_64) with 2 vCPU, 8GB of memory and 450Mbps bandwidth. After probing the entire IP space, we fetch the data set from the server and use Python to pre-process and extract useful information (such as IP, timestamp, and response information). Using the prototype system, we have conducted the experiments for eight times from August 2015 to March 2016, and the details are shown in Table V. Each time, we have exhaustively searched the entire IP address space (close to 3.7 billion addresses). We excluded both reserved/unallocated IP space from IANA [20] and those IPs that send emails

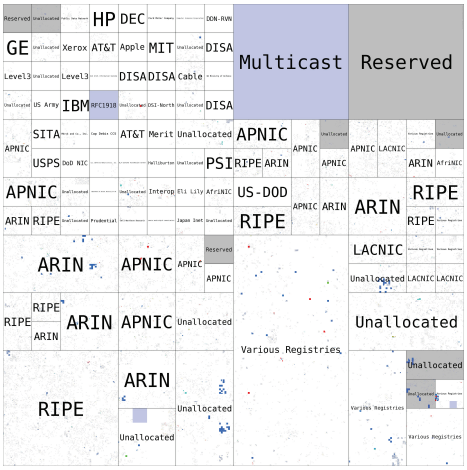


Fig. 8: Physical devices distribution map as a form of the Hilbert curve.

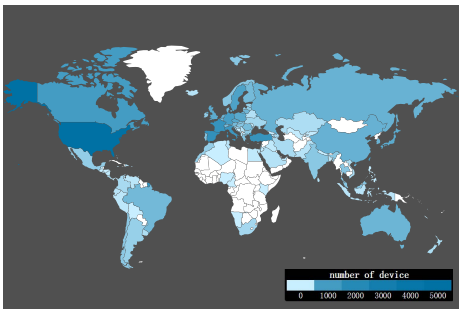


Fig. 9: ICS devices distribution in world map.

to complain about our scanning activities. All together, we added about 610 million IP address to our blacklist to exclude them from the Internet-wide search. In Table V, for each data collection, we used 50,000 packets per second, and each experiment was finished in about 20 hours.

B. What are the distribution features of ICS devices?

We first investigate the IP-layer distribution of these ICS devices. We adopt the Hilbert curve [21] as the form of ICS device distribution map in the Internet space. It is a type of space-filling curve visualization of mapping the entire 32-bit IPv4 address space in two dimensions. As shown in Figure 8, many ICS devices have an irregular distribution. Although ICS device distribution is irregular in the Internet space, we would like to find out whether there is any correlation between spatial location and device distribution.

Furthermore, we explore the spatial association with ICS devices. We use MaxMind's GEOIP [17] database to map between IP addresses and their locations. This database has a mapping between IP addresses and their locations at the city level. More fine-grained location information is unavailable and also unnecessary for analyzing ICS devices in our study. Figure 9 shows the ICS devices' spatial distribution in the world map. The darker blue color shows that there are more ICS devices deploying in this domain. We can also see that

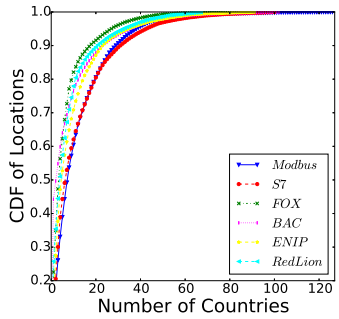


Fig. 10: Location distribution of ICS devices at country-level.

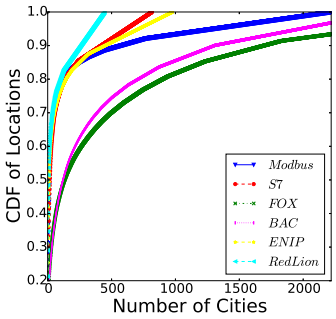


Fig. 11: Location distribution of ICS devices at city-level.

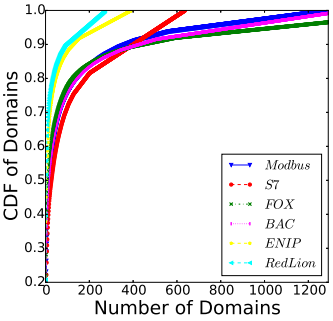


Fig. 12: Domain distribution of ICS devices.

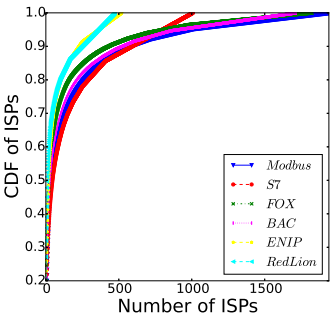


Fig. 13: ISP distribution of ICS devices.

their location distribution is not a uniform distribution. There are in total more than 128 countries with devices running ICS. We further analyze the ICS device distribution at the country level. As shown in Figure 10, more than 84% of the Modbus devices are located in the top 20 countries, 68% of the Modbus devices are in the top 10 countries, and there is a long-tail effect in their ICS device distribution. Other ICS devices exhibit similar distribution characteristics. At the city-level, Figure 11 shows a long-tail effect in ICS device distributions. The three ICS protocols, Modbus, Fox and BACnet, have the largest coverage. More than 47% of Modbus devices are in the top 10 cities, 60% of the Modbus are in the top 20 cities, and the top 200 cities can include 85% of the Modbus devices. Redlion has the least coverage, with only 69 countries having physical devices running Redlion. The long-tail effect of the ICS device location distribution is more prominent at the city-level than at the country-level. RedLion, S7 and ENIP have a more obvious long-tail effect than the others. These observations might be due to fact that in more economically developed countries, there are more industrial infrastructures where S7 and Modbus are the typical SCADA [1] devices.

We have not observed any distribution of ICS devices at the IP level, but there is an obvious long tail effect of location distribution of these devices. This is because IP addresses are allocated by IANA and are continually backwards compatible.

C. Who use ICS devices?

To find out who use these devices, we utilize online rDNS to map each IP address to its domain name and use offline Geolocation compatible with python to map each IP address to its server provider and geo-location. An obvious problem is that rDNS does not associate each IP address with a domain name. This is because many device owners do not register rDNS with their IP addresses. In our data set, nearly half of the IP addresses have a null domain name. Different from rDNS, the Geolocation library has an IP address for every server provider and location. To find out domain names, we leverage both rDNS and the Geolocation library.

TABLE VI: Top 10 Countries and its GDP

	Country/Region	Rank Of GDP	Number of Devices
1	United States	1	47,818
2	China	2	13,828
3	Canada	10	5,497
4	France	6	3,786
5	United kingdom	5	3,726
6	Italy	8	3,543
7	Germany	4	3,322
8	Poland	25	2,899
9	Russian	12	2,860
10	South Korea	11	2,796

Each domain name has several successive strings with dots, (e.g., *"static-71-170-4-117.dllstx.fios.verizon.net"*). We extract the last two strings as a domain name from our data set, and there are almost 1,300 domains. We extract the antepenultimate string to illustrate the enterprise name in the domain name. As shown in Figure 12, more than 66% of Modbus devices are in the top 50 domains (owner names), and 83% of Modbus devices are in the top 200 domains. There is also a long-tail effect for the owners of these ICS devices. We also get server providers of Modbus devices. As shown in Figure 13, there are almost 2,000 server providers, obviously a long-tail effect in their distribution over server providers.

To further discuss the relationship between ICS devices and economic development, we refer to the data from the International Monetary Fund's World Economic Outlook (WEO) [22], April 2015. Countries are sorted by nominal GDP estimates from financial and statistical institutions in this data. Intuitively, a more developed country has more ICS devices, which is indicated by Table VI. The top 10 device countries are all in the top 25 with the highest GDP. ICS devices are naturally parts of industrial control systems, whose number is an important indication of economic development. Therefore, there is a positive correlation between ICS device distribution and economic development. However, some other aspects of these ICS devices, such as temporal dimension, temporal-spatial

TABLE VII: Types of devices running Modbus and EtherNet/IP in Internet-wide range

	Device Type	Count		Device Type	Count
Modbus	programmable logic controller	861	EtherNet/IP	communications adapter	1,989
	solar panel	620		programmable logic controller	1,852
	water flow controller	436		generic device (deprecated)	402
	scada controller	212		human-machine interface	68
	light controller	32		safety discrete I/O device	15
	power controller	13		generic device (keyable)	6
	network analyer	6		AC Drive	3
	power monitor	1		generic type	2,853
	generic type	17,546			

dimension, or other ICS devices running other protocols, have not yet been studied in this work. We will conduct further investigations in the future.

D. What are the functions of ICS devices?

Different ICS protocols have different ways to identify and extract ICS device functions. We extract this information (device type) based on the ICS protocol payload format. Modbus and EtherNet/IP are standard communication protocols for ICS, and we could infer their functions based their detailed information. For instance, Table VII shows our inference of the functions of ICS devices with Modbus and EtherNet/IP.

Other protocols are not standard communication protocols but particular protocols for certain industrial domains or used for certain companies. For instance, BACnet and Tridum Niagara Fox are mainly used in automation control, DNP3 and IEC 104 are usually used in electric power systems, Red Lion Crimson v3 is used as ICS HMI touch panels, Siemens S7 is used in traditional PLC (Programmable Logic Controller), and PCworks is the consistent engineering software for all controllers from Phoenix Contact.

We could also infer a device's function based on the protocols themselves and the detailed information of the devices. However, the challenge is that many ICS devices refuse to respond to the question of what they are being used for, an it is not an option to send too many packets to remote ICS devices.

V. Related Work

Active probing. Active probing is the process of identifying network services by transmitting certain packets toward network hosts and extracting their responses. Nmap [23] is a popularized tool to identify the OS of a remote host based on differences between TCP implementations [6]. It has also been used to track specific devices based on device clock skews in the physical layer and extract service information in the application layer [24].

Our work is closely related to the active probing technique that detects banners in the application layer. Banner grabbing is routinely performed during the penetration testing of a network, in order to identify software version or type advertised in application properties during a connection attempt. Previous work [25] demonstrates that it can be done in real-time to remotely determine the applications or services running on a particular host of interest. Similar to banner grabbing with active probing, our work analyzes 17 Industrial protocols that operate as application layer protocols. We send standardized packets to remote networks and determine ICS devices if their responses satisfy industrial protocol properties. Our work differs from others in that we first filter out unqualified remote hosts and then build normal ICS protocol connection activities. Our approach is less intrusive with the least number of probe packets, and achieves high accuracy.

Scanning. Prior work demonstrates the use of Internet-wide scanning for discovering live hosts, called horizontal scanning. Since there are 4 billion IPv4 addresses in total, horizontal scanning is time consuming. The network reconnaissance research [16], [26]–[29] mainly focuses on speeding up the scanning performance. Over time, the scanning does get progressively faster—from 4 months down to 30 days, one day, and more recently, 45 minutes. Xie et al. [26] proposed the UDmap algorithm to identify and analyze hosts across the entire IP address space. Heidemann et al. [27] explored the visible Internet to characterize the edge nodes and evaluate their usages via an active scanner within 30 days. Hong et al. [29] searched the entire Internet and then classified IP addresses as popular or unpopular. Leonard et al. [28] implemented IRLscanner as an Internet-wide detection mechanism with an improved speed of 24 hours. The focus of IRLscanner is on typical application layer protocols such as HTTP and SMTP, while the focus of our work is on ICS device detection. Durumeric et al. [16] proposed ZMap for Internet-wide host detection with the speed of 45 minutes. However, such a high speed often cannot be achieved in practice due to network congestions. Similar to ZMap, our approach sends just one packet to every IP address but aims to extract device information as much as possible. Different from these previous scanning studies, our work is to identify ICS devices by leveraging Internet-wide scanning.

Industrial control system measurement. An industrial control system is a classic case of cyber physical systems associated with the physical world. For remote access, industrial control devices should be access-controlled by a firewall or a virtual private network (VPN). However, SCADA-enabled devices expose themselves in the public Internet, enabling attackers to remotely locate and subsequently exploit vulnerabilities. There are several previous research works focusing on exploiting the cyberspace scanning for security event detection [30], [31]. Igure et al. [32] presented serious security problems about SCADA-enabled devices that network reconnaissance could discover. Durumeric et al. [31] detected a security event (an OpenSSL flaw "Heartbleed") at the Internet scale. If similar problems arise in an industrial control system, a real-time discovery mechanism is needed. Our work is a first step to identify ICS devices and characterize their usage on

the Internet-wide range.

The most related work to ours is the very recent work by Formby et al. [33], who adopted fingerprinting methods to discover two types of ICS devices. They used passive probing to extract the static and low-latency nature of ICS as their signatures to improve their discovery accuracy. By contrast, we use active probing to investigate ICS devices in the Internet-wide range. Our approach is able to probe any network subsets to detect and measure ICS devices. We could also incorporate their passive measurement techniques to improve our own measurement performance.

VI. Conclusions

ICS devices with IP addresses are accessible on the Internet and play a crucial role for critical infrastructures. In this paper, we have discovered ICS devices in a real-time and non-intrusive manner and characterized their usage on the Internet. Specifically, we have analyzed 17 Industrial protocols widely used in ICS devices and adopted a standardized communication to extract ICS device information with normal packet payload. A probability model is trained to find an ICS honeypot candidate, and a honeypot fingerprint is generated by our heuristic algorithm. We have proposed an online algorithm to discover ICS devices all over the Internet. In the experiments conducted from August 2015 to March 2016, we have attempted to detect visible ICS devices as many as possible within the entire IPv4 space. Based on the ICS device data collection, we have performed a comprehensive data analysis to characterize the usage of ICS devices on the Internet, especially in the answer to the following three questions: (1) what are the distribution features of ICS devices, (2) who use these ICS devices, and (3) what are the functions of these ICS devices.

VII. Acknowledgments

We are grateful to our shepherd Ranveer Chandra and anonymous reviewers for their insightful feedback. This work was supported in part by the National Natural Science Foundation of China (Grant No. U1536107), National Key Research and Development Program (No. 2016YFB0800202), Innovation Foundation of the Chinese Academy of Sciences (No. CXJJ-16M118), and the "Strategic Priority Research Program" of the Chinese Academy of Sciences,Grant No.XDA06040101.

References

[1] S. A. Boyer, *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.

[2] Ics-cert year in review 2015. [Online]. Available: https://ics-cert.us-cert.gov/Year-Review-2015

[3] Internet-wide ics devices scaning research. [Online]. Available: https://www.cpsteam.org/

[4] The go programming language. [Online]. Available: https://golang.org/

[5] Amazon elastic compute cloud (amazon ec2). [Online]. Available: https://aws.amazon.com/ec2/

[6] D. E. Comer and J. C. Lin, "Probing tcp implementations," in *Usenix Summer*, 1994, pp. 245–255.

[7] The honeynet project. [Online]. Available: https://www.honeynet.org/

[8] Digital bonds scada honeynet. [Online]. Available: http://www.digitalbond.com/tools/scada-honeynet/

[9] Conpot. [Online]. Available: https://github.com/mushorg/conpot/

[10] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, 2009.

[11] C. M. Bishop, "Pattern recognition and machine learning (information science and statistics)," 2007.

[12] J. Caballero, S. Venkataraman, P. Poosankam, M. G. Kang, D. Song, and A. Blum, "Fig: Automatic fingerprint generation," in *Proceedings of NDSS*, 2007.

[13] M. Allman, V. Paxson, and J. Terrell, "A brief history of scanning," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 77–82.

[14] G. Bartlett, J. Heidemann, and C. Papadopoulos, "Understanding passive and active service discovery," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 57–70.

[15] numpy: Randomly permute a sequence. [Online]. Available: http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.permutation.html

[16] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications." in *Proceedings of Usenix Security*, 2013, pp. 605–620.

[17] Maxmind geoip2. [Online]. Available: https://www.maxmind.com/en/geoip2-services-and-databases

[18] K. P. Murphy, "Naive bayes classifiers," *University of British Columbia*, 2006.

[19] Plcscan plc devices detection on the network. [Online]. Available: https://code.google.com/p/plcscan/

[20] Assigned numbers authority (iana). [Online]. Available: http://www.iana.org/

[21] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz, "Analysis of the clustering properties of the hilbert space-filling curve," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 1, pp. 124–141, 2001.

[22] International monetary funds world economic outlook (weo),april 2015. [Online]. Available: http://www.imf.org/external/pubs/ft/weo/2015/01/

[23] Nmap, network security scanner tool. [Online]. Available: https://nmap.org/

[24] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[25] D. Stuttard and M. Pinto, *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. John Wiley & Sons, 2011.

[26] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber, "How dynamic are ip addresses?" in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 301–312.

[27] J. Heidemann, Y. Pradkin, R. Govindan, C. Papadopoulos, G. Bartlett, and J. Bannister, "Census and survey of the visible internet," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*. ACM, 2008, pp. 169–182.

[28] D. Leonard and D. Loguinov, "Demystifying service discovery: implementing an internet-wide scanner," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 109–122.

[29] C.-Y. Hong, F. Yu, and Y. Xie, "Populated ip addresses: classification and applications," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 329–340.

[30] E. Bou-Harb, M. Debbabi, and C. Assi, "Cyber scanning: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1496–1519, 2013.

[31] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer *et al.*, "The matter of heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 475–488.

[32] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in scada networks," *Computers & Security*, vol. 25, no. 7, pp. 498–506, 2006.

[33] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah, "Whos in control of your control system? device fingerprinting for cyber-physical systems," in *Proceedings of NDSS*, 2016.