Release the Hounds! Automated Inference and Empirical Security Evaluation of Field-Deployed PLCs Using Active Network Data

Ryan Pickren
Georgia Institute of Technology
Atlanta, GA, USA
rpickren3@gatech.edu

Animesh Chhotaray
Georgia Institute of Technology
Atlanta, GA, USA
achhotaray3@gatech.edu

Frank Li
Georgia Institute of Technology
Atlanta, GA, USA
frankli@gatech.edu

Saman Zonouz
Georgia Institute of Technology
Atlanta, GA, USA
szonouz6@gatech.edu

Raheem Beyah
Georgia Institute of Technology
Atlanta, GA, USA
rbeyah@coe.gatech.edu

Abstract

Surveying field-deployed Industrial Control System (ICS) equipment has numerous security applications, including attack-surface management and measuring the adoption of vulnerability patches. However, discovering real-world devices using massive Internet-scale scan datasets is tedious and error-prone. We introduce *PLCHound*, a novel ICS asset discovery solution designed to automatically reveal elusive ICS devices hiding in network data collected by Internet-scale scanners such as Censys or Shodan. Our solution systematically uncovers indirect evidence of controllers using subtle network-based indicators and temporally-resistant signatures that are often overlooked in prior work. We present *PLCHound*'s architecture, experimentally verify its accuracy, and explore the security advantages of enhanced device discovery. We also use *PLCHound* to perform the largest comprehensive examination of the publicly-reachable population of ICS devices by popular vendors. Our results reveal that the industry-accepted estimations and latest published papers undercount the true number of public devices by up to 37x. We also find that 95.88% of devices expose protocols that cause them to be remotely vulnerable to recent critical CVEs.

CCS Concepts

• **Networks** → *Network protocols*; • **Security and privacy** → **Network security**; • **General and reference** → *Measurement*.

Keywords

Network Security, Industrial Control Systems, Programmable Logic Controllers

ACM Reference Format:
Ryan Pickren, Animesh Chhotaray, Frank Li, Saman Zonouz, and Raheem Beyah. 2024. Release the Hounds! Automated Inference and Empirical Security Evaluation of Field-Deployed PLCs Using Active Network Data. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3658644.3690195

CCS '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-0636-3/24/10
https://doi.org/10.1145/3658644.3690195

1 Introduction

Industrial Control Systems (ICSs) refer to the broad category of software and hardware systems designed to automate industrial processes, like those commonly found in critical infrastructure such as water treatment plants, nuclear facilities, and electrical substations. At the heart of these systems, ruggedized single-purpose computers called Programmable Logic Controllers (PLCs) control physical equipment by reading sensors and manipulating actuators. PLCs communicate with peripheral ICS devices such as Human-Machine Interfaces (HMIs), Engineering Workstations, and Data Historians via a plethora of IP-based networking protocols. Historically, PLCs exclusively used domain-specific protocols such as ISO-TSAP (UDP/102), EIP (UDP/44818), and Modbus (TCP/502) to transmit information within closed ICS environments. Recently, PLCs also began to incorporate standard IT protocols such as HTTPS (TCP/443), SSH (TCP/22), and FTP (TCP/21) for administrative configuration and maintenance.

Unfortunately, the interconnectivity of these systems have made them a prime target for cyberattacks in recent years. Notable real-world attacks on ICSs include the infamous Stuxnet [36] worm, which targeted Iran's nuclear program, Triton [24], which disrupted Saudi petrochemical plants, and the recent Ukraine power grid attacks [23]. These incidents highlight that cybercriminals are actively targeting and successfully compromising these systems.

To further compound the issue, many ICS networks are misconfigured to allow inbound traffic from the public Internet to reach their controllers. The existence of Internet-facing ICS devices, especially PLCs, is an extensively studied phenomenon [6, 27, 39, 42, 70] and has directly resulted in recent critical infrastructure breaches [20]. This topic gained attention shortly after the proliferation of Internet-scale scanning services such as Shodan [2] and Censys [1] due to the urgent need to study and protect exposed critical infrastructure. These scanning services probe public IP addresses using common network protocols and record the decoded responses into large searchable databases, thus creating a vast catalog of Internet-facing network services. These databases provide an extensive, although oftentimes overwhelmingly large, source of data to search for exposed ICS devices.

Interestingly, the industry standard approach to querying these massive datasets for ICS devices is somewhat naive and one-dimensional. Most prior work uses a simple heuristic approach, where a manually chosen indicator (e.g., stringified model number) is

searched against a commonly used port for ICS protocols (e.g., 502). This approach is widely used in academic papers [6, 17, 27, 39, 70] and industry-leading reports [34, 51]. In fact, Shodan itself recommends using this approach to query its dataset [60].

While this approach may yield some results, it unfortunately only captures the small subset of devices that are grossly misconfigured to allow their primary, unredacted, ICS protocol to be exposed. In practice, most ICSs enforce strict network segregation, where critical traffic is isolated to secure enclaves and blocked by firewalls [16]. Furthermore, the responses from network probes shift over time depending on the firmware version and customer configuration, meaning that a static, manually-chosen, indicator may only apply to a fraction of exposed devices. Thus, the complex state of ICS environments results in incomplete and transitory network data, which causes accurate large-scale security analyses to be exceedingly challenging. Refining a query to include multiple protocols and capture shifting probe responses requires substantial manual effort, as demonstrated in Section 2.2.

In this paper, we address this challenge by developing a custom optimization algorithm that performs robust entity resolution using subtle network-based indicators and temporally-resistant signatures. This ICS asset discovery solution, which we call *PLCHound*, was created using observations about how ICSs manifest in practice. Specifically, we capitalize on the observation that once devices become field-deployed, customers often enable additional protocols and incorporate modular hardware add-ons to increase functionality (e.g., telecom gateways, local HMIs, etc.). We also observe that ICSs tend to simultaneously use multiple PLCs behind a single NAT, meaning that clues about different firmware versions and configuration settings are often present in search results even if the query did not directly discover those devices. We use this auxiliary information to infer how the field-deployed population has changed over time, and through inductive reasoning, build complex signatures that capture the true field-deployed population. Unlike prior work that attempts to directly read obvious static indicators in ICS protocols, our approach uncovers evolving network-based signatures that allow us to indirectly infer the existence of devices.

Beyond merely identifying more hosts than prior work, *PLCHound* discovers a more diverse and realistic population of devices. These devices speak multiple protocols, use a variety of firmware versions, and are coupled with an assortment of hardware add-ons. The rich nature of this dataset enables us to answer important security questions and reveal interesting security-related trends. For example, we use *PLCHound* to study the real field-deployed attack surface, gain insight into industrial firewall configurations, explore non-ICS methods for attacking ICSs, and survey customer-enabled security settings. Our results show that vulnerable co-located devices actively provide attackers with potential paths to circumvent industrial firewalls, and even more troubling, that 95.88% of devices directly expose protocols that cause them to be remotely vulnerable to recent critical CVEs. Using these results, we conducted an extensive disclosure campaign to notify vulnerable network administrators and co-authored a public security advisory with the ICS vendor. In summary, our main contributions are as follows:

(1) We introduce *PLCHound*, a holistic, cross-protocol, ICS asset discovery solution that infers the existence of devices using subtle indicators and temporally-resistant signatures (with a verified 98.67% true positive rate);
(2) We perform the largest comprehensive survey of Internet-facing PLC devices by popular vendors (up to 37x more devices than prior work);
(3) We provide the first rigorous examination of the field-deployed ICS attack surface (revealing that 95.88% of devices are vulnerable to recent CVEs).

2 Background and Motivation

This section discusses prior ICS asset discovery papers and provides the reader with the real-world case study that motivated our work.

2.1 Related Work

Hunting for Internet-exposed ICS devices, especially PLCs, is an extensively explored area of research. This topic was largely pioneered by Leverett et al. in their 2011 paper, where they detailed hand-curated Shodan queries and their corresponding ICS devices [39]. This academic work was quickly followed up by an industry report, co-authored by the United States Department of Homeland Security, called "Project Shine" that sought to quantify to what extent critical systems were exposed to the Internet [51].

These works laid the foundation for numerous subsequent studies [5, 27, 35, 69, 70] that researched different aspects of the Internet-accessible ICS population. Unfortunately, accurately querying the large datasets produced by Shodan and Censys proved to be challenging, which forced most prior work to limit their search to a single specific protocol and use a best-effort static string query. Recently, Ashley et al. explicitly discussed the painstaking process of manual query formation and proposed a query optimization model using flowcharts [6].

To the best of our knowledge, our paper is the first and only work that automates the ICS asset discovery process by programmatically generating queries that span multiple protocols, user settings, and device configurations. The most similar work to our paper is the recent study by Sasaki et al., which used fuzzy hashing to group remote management device WebUIs together, although their work still required manual signature extraction and query formation [55].

2.2 Motivating Example

For our motivating example, we aim to identify the ICSs that are remotely vulnerable to CVE-2022-45140, a recent critical (CVSS 9.8) unauthenticated remote code execution vulnerability affecting multiple families of WAGO devices and present in every firmware version since inception. Specifically, this vulnerability impacts PLCs by abusing an HTTP-based Application Program Interface (API) exposed by the on-board embedded webserver. This analysis is useful for many parties, including 1) government agencies performing mandatory audits (e.g., NERC CIP [18]) 2) large corporations with many disperse facilities (e.g., telecoms), and 3) independent security researchers. Note that while this specific example only impacts WAGO PLCs, a similar analysis can be performed using other vulnerabilities impacting other ICS vendors (e.g., Siemens, Allen-Bradely, Mitsubishi, etc.).

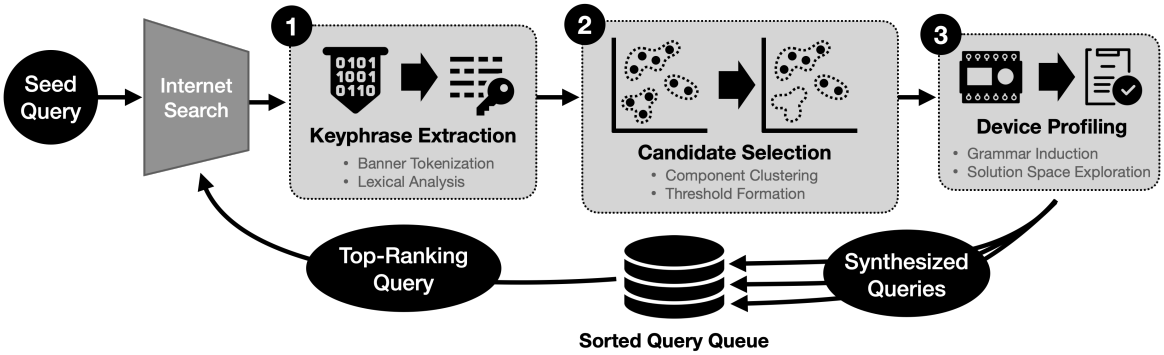

Figure 1: High-Level Overview of the *PLCHound* Algorithm

Searching for these devices using traditional approaches (i.e., manually chosen keyword against a commonly used ICS port) would entail using a Censys query like below:

same_service(port=44818 AND banner:"750-881")

This query uses the aforementioned methodology since *44818* is the port commonly used for the EIP ICS protocol and *750-881* is the model number of one of the affected devices. As of the time of writing, this query discovers 7 hosts, 6 of which expose one or more HTTP servers. Since these hosts oftentimes contain multiple devices behind NAT, we must manually inspect every HTTP server to confirm they belong to the target device. In total, these 6 hosts exposed 12 unique HTTP servers and a manual inspection (using an HTTP client such as *curl* [21]) reveals that 5 of them do indeed route to the PLC. In summary, the above querying methodology shows that 5 PLCs are remotely vulnerable to CVE-2022-45140.

Many prior works [9, 25] stop at this point, however a keen observer may note that we can use those 5 results to form a new query to directly find more HTTP-enabled PLCs. Manual inspection of the HTTP responses and a cross-reference to the Censys Data Definitions Search Syntax [12] allows us to form the below query:

same_service(port=80 AND http.response.html_title=" WAGO Ethernet Web-Based Management ")

As of the time of writing, this query discovers 303 hosts, meaning that this new query resulted in a 5,960% increase in discovered ICSs. These results suggests that the HTTP protocol is less scrutinized by industrial firewalls and network operators than typical ICS protocols such as EIP. Unfortunately, for the reasons mentioned in Section 1, this query is still only capturing a small subset of the total population.

To expand this query even further, the user must manually inspect all hosts to look for additional PLC-originating protocols not picked up by this query but still present in the search results nonetheless due to customers either owning multiple PLCs behind a single NAT or using multiple different protocols to speak to the same device. These supplementary datapoints may reveal clues about different firmware versions and configuration settings, which may be used to expand the search. The user must manually compare different non-captured protocol responses and isolate common, yet still distinct enough to not introduce false positives, substrings to build syntactically valid queries. At the end of this tedious and error-prone process, the user may eventually end up with the below query:

*same_service(extended_service_name:{"HTTP", "HTTPS"} AND (http.response.headers:(key:Server AND value.headers:"WAGO_Webs") OR http.response.html_title=/[ ]?WAGO [A-Za-z]*Web-(B|b)ased Management[ ]?/ OR tls.certificates.leaf_data.subject.email_address=/[a-z]{3}[a-z]?o[a-z\S\s]*\@wago\.com/))*

This query encompasses multiple device configurations, firmware versions, and user settings, thus capturing a much larger percentage of the field-deployed population. As of the time of writing, this query resulted in 6,616 hosts (+2,084%). Key components of the above query include the following:

(1) **Temporally-Resistant Signatures** e.g., regular expression for the HTML title that allows firmware updates to evolve the string from " *WAGO Ethernet Web-based Management* " to *"WAGO Web-based Management"* to *"WAGO Web-Based Management"*;
(2) **Device Configurations** e.g., including both the encrypted and plaintext variations of the HTTP protocol;
(3) **User Settings** e.g., localization language preference changing the TLS certificate contact from *info@wago.com* to *support.de@wago.com*;

These results reveal that the true number of Internet-facing devices vulnerable to CVE-2022-45140 is over 1,323X that discovered with the naive EIP-based query and 21X that discovered with the direct HTTP query.

Unfortunately, developing an advanced query like this requires substantial manual effort and multiple iterations of intermediate queries to collect enough datapoints to generalize patterns. In this paper, we present a fully automated approach that begins with a simple seed query and automatically expands the search without any user interaction or effort.

3 Solution Overview

The *PLCHound* algorithm has three major stages, ❶ **Keyphrase Extraction**, ❷ **Candidate Selection**, and ❸ **Device Profiling**, as shown in Figure 1. This section provides a high-level overview of each stage to give the reader insight into *PLCHound*'s core functionality. See Section 4 for in-depth design details. While this section uses the PLC from the motivating example as a case study to build

intuition, the algorithm is generic enough to work for all ICS vendors, as demonstrated in Section 5.2. Thus, our solution is suitable for any ICS cybersecurity effort that seeks to examine the complex field-deployed attack surface.

Search Engine & Seed Query. Our contribution builds on top of a queryable dataset of active network scan data. This dataset of protocol probe responses (called "banners") can be independently generated using publically available scanners (e.g., Zmap [26] or Nmap [44]) or accessed via a third-party commercial service (e.g., Censys [1], Shodan [2], or ZoomEye [71]). Implementation details regarding the search engine are outside the scope of this paper since our approach is fully agnostic to the underlying scanner.

PLCHound takes a simple seed query as input, which is used to guide the initial trajectory of the algorithm. Any query developed using the industry-standard technique from prior work (i.e., keyword against a commonly used ICS port) is sufficient. Ideally, this query should have a low false-positive rate and return exemplary results. We discuss the algorithm's sensitivity to different seed queries in Section 5. As an example, we can use the naive EIP query from Section 2.

Stage 1: Keyphrase Extraction. Once an initial database of devices has been constructed using the search results from the seed query, *PLCHound* enters the *Keyphrase Extraction* stage. In this stage, the algorithm is tasked with automatically extracting meaningful textual phrases that can be used as positive indicators that the PLC is present. See Section 4.1 for details regarding the custom tokenization process, domain-specific lexical analysis [43, 49], and NLP-based conditionals used to assign relevance scores to phrases.

Running the banners found by our example seed query through this process identifies the following phrases as relevant, in decreasing order: {"*wago*", "*pfc200*", "*kontakttechnik*", [...] "*ssi*"}. In this example, the first phrase ("*wago*") is the manufacturer name, the second phrase ("*pfc200*") is the device family name, and the third phrase ("*kontakttechnik*") is a German expression meaning *Communication Technology*. This list continues up to the phrase "*ssi*," which is the obscure file extension used by the server-side scripting language of the embedded webserver. Note that none of these phrases were included in the seed query, thus demonstrating the algorithm's ability to extract keyphrases that the user may be unaware of and possibly not even understand due to language barriers. These phrases influence how the algorithm selects candidates in the next stage and ultimately synthesizes queries in the final stage.

Stage 2: Candidate Selection. As mentioned in Section 1, real-world ICSs often run multiple devices behind a single NAT, causing them to share the same public IP address. When those other devices are differently configured PLCs, we use them to garner clues about different firmware versions and user settings. However, if those other devices are unrelated to the target PLC, they can potentially misguide the algorithm and cause unfounded correlations. To avoid this situation, *PLCHound* automatically identifies and removes the irrelevant co-located banners from the dataset, as shown in Figure 2. For example, running the banners found by our seed query through this stage automatically identifies the PLC's Web-Based Management (WBM) web portal as a valid banner type while simultaneously discarding the irrelevant co-located employee login

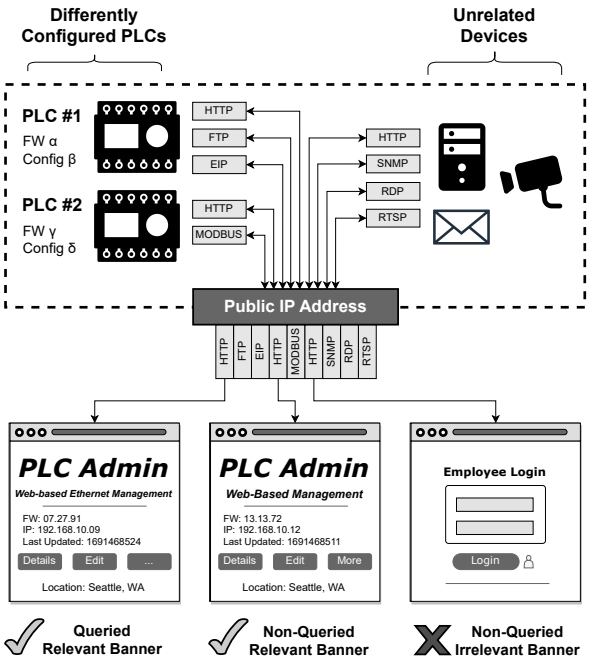

Figure 2: Example of Candidate Selection Process

portals. Details regarding our banner clustering process, cluster ranking methodology, and dynamic threshold formation are included in Section 4.2.

Stage 3: Device Profiling. In the *Device Profiling* stage, *PLCHound* builds regular expression (regex) based signatures that account for the variability within the remaining valid banners. This variability is caused by the field-deployed customizations discussed in Section 1 such as firmware version differences, customer language preferences, and usage statistics. These signatures intentionally use flexible character sets in areas of high entropy to potentially match unseen valid banners. *PLCHound* then scrutinizes each signature to ensure it is descriptive enough to not match false positive banners. Lastly, these signatures are used to synthesize new queries that can be safely applied to the Internet-wide dataset. See Section 4.3 for details regarding the fault-tolerant grammar induction and programmatic solution space exploration used to synthesize queries. The valid banners within the example seed dataset ultimately synthesized the following query to discover PLCs using the FTPes protocol.

```
same_service(extended_service_name="FTPes" AND
tls.certificates.leaf_data.subject_dn=/C\=DE, ST\=Germany,
[A-Z][A-Z]?\=[A-Z][a-zA-Z\S\s0-9]+, O\=Wago Kontakttechnik
GmbH \& Co\. KG, [A-Z][A-Z]?\=[A-Z][A-Z\S\s0-9a-z]+,
emailAddress\=info\@wago\.com/)
```

The entire cycle then repeats, each time finding new devices, honing the device profile, and synthesizing more robust queries as it learns more about the total population. This process creates a diverse multi-protocol dataset of field-deployed devices, which accurately depicts the complex and fragmented state of real-world ICS environments.

4 Algorithm Design Details

This section discusses the in-depth design details for the three stages of the *PLCHound* algorithm, namely ❶ **Keyphrase Extraction**, ❷ **Candidate Selection**, and ❸ **Device Profiling**. In our dataset, "hosts" refer to Internet-facing middleboxes with public IP addresses and "banners" refer to the decoded protocol responses from probing hosts. As described in this section, *PLCHound* is designed to automatically minimize False Positive Hosts (FPHs) by only using True Positive Banners (TPBs) to synthesize robust, yet descriptive, new queries.

4.1 Stage 1: Keyphrase Extraction

Since *PLCHound* only utilizes TPBs to synthesize new queries, it must somehow identify and prune away the False Positive Banners (FPBs). ICS subject-matter experts are able to manually perform this task by inspecting all banners and making educated guesses based off of substrings found in their ASCII representations. For example, a human operator can deduce that an SNMP banner containing the product model in the system description (e.g., "*6ES7 215-1BG40-0XB0*") will likely belong to the PLC. Unfortunately, this strategy is often tedious and requires deep domain-specific knowledge.

PLCHound replaces the need for domain-specific knowledge by automatically assigning "importance scores" to banner substrings in the *Keyphrase Extraction* stage. These scores are a prerequisite to FPB pruning, since they determine which phrases (and by extension, which banners) likely originated from the target device. In practice, high-scoring phrases are often model numbers, vendor names, and chipset identifiers. The goal of this stage is to automatically identify these strings. To accomplish this task, we first split each banner into discrete phrases (*Banner Tokenization*), determine the importance of each phrase (*Lexical Analysis*), and remove the nondescript phrases (*NLP Pruning*), as discussed below.

Banner Tokenization. The objective of the *Banner Tokenization* step is to intelligently parse raw banners into meaningful substrings called phrases. A key challenge with this step is keeping semantically equivalent, yet syntactically different, phrases consistent across different protocols. For example, the phrase *"s7 1200"* could be represented as *"s71200"* inside the JavaScript code located within the HTTP response, represented as *"S7-1200"* inside the FTP login banner, and represented as *"s7\uFFFD1200"* in the raw EIP bytestream.

We account for these differences by first tokenizing banners using common delimiters found in human-readable language corpuses [57] (e.g., whitespaces, punctuation, and newlines) then applying our own custom tokenization rules. These rules accommodate raw network streams, high-level source code, and other common data types found in banners. In particular, we unescape control sequences (e.g., null bytes), split apart common naming conventions in source code (snake case and camel case), and delimit all non-printable byte sequences.

Next, we use a variable-length sliding window to capture N-Grams consisting of one-to-five individual tokens. These N-Grams are then joined with a whitespace (" ") to create protocol agnostic multi-token phrases (e.g., *"s7 1200"*).

Lexical Analysis. Since the above tokenization process creates a large number of mostly-overlapping phrases, we must systematically determine which of them can aid in the targeted search process and which can be discarded as irrelevant. Irrelevant phrases can either originate from FPBs (e.g., employee login pages that share the same public IP addresses as PLCs) or nondescript portions of TPBs (e.g., RFC-defined FTP status strings). The goal of this step is to automatically identify which phrases are significant enough to guide the trajectory of the algorithm.

To numerically quantify our confidence that a given phrase originated from a TPB, we measure its occurrence in explicitly queried banners (called "matched banners"). This value is then normalized so a 1 indicates that the phrase is present in every matched banner and a 0 indicates that it is not present in any matched banners. We refer to this number as the phrase's *match score*, $S_{Match}(x)$. In practice, phrases with high match scores tend to be relevant strings such as *"siemens simatic s7"* as well as generic snippets of boilerplate protocol messages such as *"html head meta charset utf."*

Hence, the match score alone is not sufficient in finding noteworthy phrases. To help identify device-specific strings, we also measure how many different protocols use that phrase. This step exploits a key property of PLCs - they are deeply interconnected to a multitude of ICS devices such as engineering workstations, HMIs, and data historians. This interconnectivity often requires PLCs to simultaneously speak multiple protocols, many of which contain common phrases that allow peripheral equipment to identify the device. The number of phrase-containing-protocols is then normalized, so a 1 indicates the token is present in every protocol and a 0 indicates it is only present in a single protocol (i.e., the protocol in which the phrase was discovered). We refer to this number as the phrase's *cross-protocol score*, $S_{XProt}(x)$. In practice, phrases with high cross-protocol scores tend to be relevant strings such as *"cpu 1200"* as well as common network verbiage such as *"user login."*

While neither metric (match score nor cross-protocol score) are perfect indicators in isolation, we observe that many phrases with high values for both metrics tend to be exceptionally relevant strings. This observation leads us to the third and final lexical metric, which we call the *super score*, $S_{Super}(x)$. This value is calculated on the pruned intersection of phrases with non-zero match scores and phrases with non-zero cross-protocol scores, and the numerical value is simply their average. The pruning process (explained next) ensures that only highly scrutinized phrases get a non-zero super score.

NLP Pruning. Crucially, not every relevant phrase is descriptive enough to accurately guide the trajectory of the algorithm. For example, even though the phrase *"nuremberg"*, the city in Germany where Siemens develops PLCs, has a high match score (since it is present in many matched banners) and has a high cross-protocol score (since it is present in many protocols), using it alone to guide the algorithm will likely introduce FPH since other companies in that city also manufacture devices.

Therefore, we must apply additional "descriptiveness" checks before awarding a given phrase a non-zero super score. These checks are inspired by commonly-used NLP conditions [54] designed to weed out insignificant tokens. Firstly, we require all tokens in each phrase to be above a certain length to avoid coincidentally-identical

short random sequences. Next, we require tokens to not be present in a domain-specific stopword list to avoid generic filler strings. We also require all tokens to meet a minimum total host agreement threshold to prevent a small minority of hosts from misguiding the algorithm. And finally, to avoid common English words, we require tokens to not exceed a large language corpus popularity threshold.

Lastly, we compile all metrics into a three-dimensional vector, $S(x) = [S_{Match}(x), S_{XProt}(x), S_{Super}(x)]$, that captures the "importance" of a given phrase, x. This vector is used in the *Candidate Selection* stage to identify TPBs and the *Device Profiling* stage to identify descriptive queries.

4.2 Stage 2: Candidate Selection

Now that *PLCHound* is aware of important phrases, it uses them to automatically isolate the TPBs from the FPBs. This is a crucial step of the algorithm because only TPBs are used to synthesize new queries.

To achieve this goal, we first break apart each banner into functionally distinct components (*Banner Decomposition*), group similar values together (*Component Clustering*), assign each group a score (*Cluster Scoring*), and dynamically infer a cutoff point (*Threshold Formation*). These steps ensure that only highly relevant and descriptive banners get treated as true positives. While we intentionally ignore the FPBs during the core algorithm, Section 5.3 explores how vulnerable services behind these FPBs can be a potential entry point for attackers into the privilaged ICS network.

Banner Decomposition. The first step in removing FPBs is to decompose each banner into functionally distinct components according to its protocol specification. For example, an HTTP banner can be parsed according to RFC7230 [28] to extract various, functionally distinct, components, such as metadata located in response headers (e.g., "Set-Cookie") and the message body. Additionally, the body content can be further parsed according to its declared content type (e.g., HTML/RFC1866 [8], JSON/RFC8259 [10], CSV/RFC4180 [58]), as shown in Figure 3. Each leaf node in the decomposition tree contains a functionally distinct string that can be individually scrutinized. For our implementation (discussed in-depth in Section 5.1), we rely on Censys's Data Definition parsing engine [12]; however similar decompositions can be produced on any raw probe response.

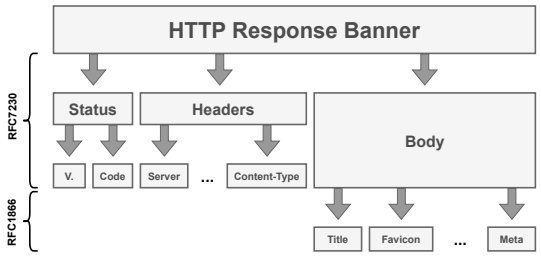

Figure 3: Banner Decomposition of HTML over HTTP

Component Clustering. Since each leaf node in the decomposition tree (called a "banner component") is functionally distinct, we can individually scrutinize it in isolation from the rest of the banner. This process allows the algorithm to easily notice similarities in certain portions of the banners (e.g., *HTML title*), while ignoring high entropy sections (e.g., *Last-Modified* HTTP header).

We use these similarities in banner component values to automatically identify outliers, which may indicate the presence of a FPB. A key challenge with this task is that banner components have an unknown number of unique "types," and more than one of them are potentially true positives. This variability is due to the high-degree of programmability of field-deployed PLCs, meaning that customers can configure their devices to inadvertently produce wildly different types of banners. For example, customers can configure the Schneider Modicon M241 PLC HTTP homepage to either be the WBM administrative page or a CODESYS WebVisu HMI screen. While these pages should not be grouped together, they are both still valid indicators of the PLC, as shown in Figure 4.

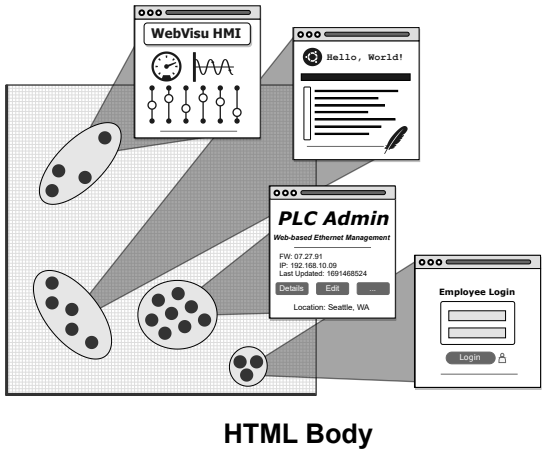

Figure 4: 2D Projection of Banner Component Clusters

We address this challenge by first transforming each component value into a feature vector using Term-Frequency Inverse-Document-Frequency [53] (TF-IDF) then use an unsupervised machine learning-based approach, where the KMeans algorithm [41] is used in conjunction with the Silhouette Method [52] to cluster neighboring feature vectors together into a variable number of clusters.

Cluster Scoring. Since each banner component has an unknown number of valid clusters, we must develop a system for measuring each cluster's validity likelihood on a 0-1 scale. Towards this goal, we created a validity metric using our domain expertise to quantitatively evaluate each cluster. This algorithm incorporates 5 different features, one of which corresponds to the "trustworthiness" of the datapoints, another corresponds to the entropy within the cluster, and the final three incorporate the lexical analysis scores from Section 4.1. Details concerning our cluster scoring algorithm are presented in Section A of the Appendix.

Threshold Formation. At this point, every banner has been decomposed into individual components and every component has an N-dimensional matrix, where similar values are clustered together and scored, as shown in Figure 5. Next, we use these scored clusters to automatically identify the FPBs.

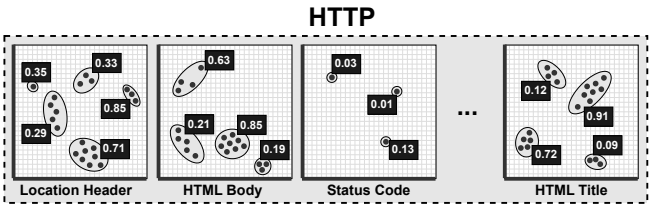

Figure 5: Scored Banner Component Clusters

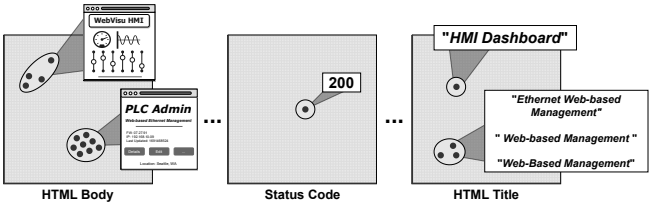

Figure 7: Example of Remaining Clusters post FPB Removal

To achieve this goal, we first create an overall banner score by averaging the scores from these independently-evaluated components. This process allows all relevant banners to be ranked highly, regardless of which part of the banner was deemed meaningful. For example, the CODESYS WebVisu HMI page may be ranked lower in the HTML body and ranked higher within the "Set-Cookie" response header, but vice versa may be true for the WBM homepage. Since all components values are averaged together, both types of pages will be considered relevant.

In practice, our clustering method often causes the subtle field-deployed difference between devices (e.g., firmware differences and customer settings) to not induce major deviations in the overall banner scores, so long as the devices are generating the same "type" of banner (e.g., HMI vs WBM). We can use a histogram of banner scores to visualize this tendency, as shown in Figure 6.

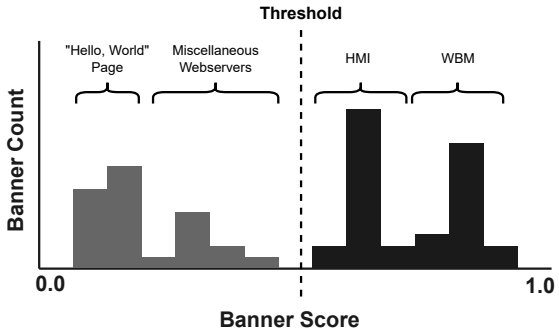

Figure 6: Histogram Illustration of Banner Scores

The final step in this stage is to methodically remove the FPBs. To accomplish this task, we incorporate a binary classification of the Jenks natural breaks optimization algorithm [33] (K=2) to automatically determine a per-protocol threshold. We consider banners above this threshold to be a likely TPB, although Section 4.3's fault-tolerant design allows for occasional misclassifications without issue. In practice, this removal procedure oftentimes eliminates many clusters entirely, since none of the enclosed datapoints were above the threshold. Therefore, the remaining clusters represent functionally distinct "types" of valid banner components, and the datapoints within the remaining clusters are real field-deployed examples, as shown in Figure 7.

4.3 Stage 3: Device Profiling

In the *Device Profiling* stage, we use inductive reasoning to meld the true positive banner component examples together (*Grammar Induction*) to create a comprehensive representation of how this device manifests in practice. This compact representation, which we call the "device profile," contains all signatures needed to discover additional hosts (*Query Synthesis*). A key challenge with this stage is that it is tasked with creating robust and widely-applicable queries, to find previously unknown devices, without being overly broad, which could introduce FPH.

Grammar Induction. While the previous stage attempted to remove all FPBs, we cannot be certain that some banners were not accidentally misclassified. This inherent risk forces us to conservatively combine the "true positive examples" in such a way where any left-in FPBs do not poison our profile. To accomplish this challenging task, we developed a highly fault-tolerant grammar induction algorithm that discovers patterns that match most, but not necessarily all, of the strings within each cluster.

During this process, we generalize common patterns so that the resulting regular expressions can potentially match valid unseen future strings. For example, if the algorithm observes that two strings differ due to a "4" being replaced by a "7", we assign this sequence a character set of all 0-9 digits, in hopes that other valid strings may include different numbers in this position. In practice, this strategy allows the algorithm to account for variable sequences inside banners such as timestamps, version numbers, and customized display names. We call these templatized banners *temporally-resistant signatures* since they account for shifting strings across firmware versions and user settings. Thus, they are able to capture devices throughout multiple stages of their lifecycle.

Query Synthesis. Unfortunately, not all signatures can produce accurate queries. Since our goal is to *intelligently hone a targeted search*, not *aggressively widen the search scope*, we must only synthesize descriptive queries.

Towards this goal, we heavily scrutinize each regular expression to ensure it will not introduce FPH if applied to the Internet-wide dataset. We do this by performing programmatic exploration of the regular expression solution space (i.e., brute forcing example matches) to essentially "fuzz" the allowed grammar. We then tokenize each example match using the process described in Section 4.1 and compare the yielded phrases to known phrases with non-zero super scores. If the example match does not contain any phrases with non-zero super scores, we conclude that the signature is too vague and therefore fails the test.

If a signature passes the test, we use it to synthesize a new query. During the synthesis process, the field name is built by traversing the decomposition tree from Section 4.2 and the value is the stringified regex signature. The specific syntax needed to build a valid query is dependent on the underlying database. Our implementation (described in detail in Section 5.1) adheres to Elastic Cache syntax using Censys Data Definitions [12].

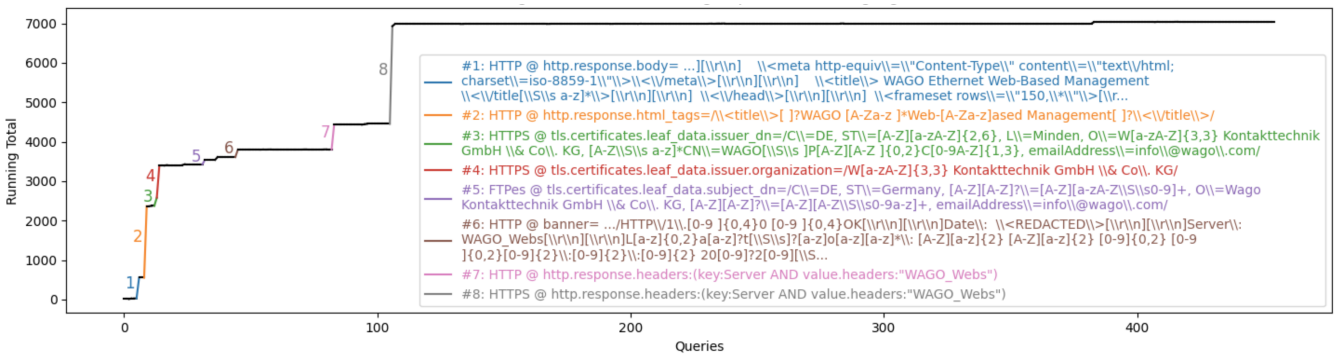

Figure 8: Count of Devices over Iterations of *PLCHound* w/ Large Uptick Queries Highlighted

Finally, we end with a list of robust, yet highly descriptive, new queries. These queries are then added to a running queue that holds all unused queries discovered during prior iterations. The best query is then applied against the Internet-wide dataset and the algorithm repeats. Details regarding the cyclical behavior can be found in Section B of the Appendix.

5 Evaluation

This section provides an overview of our software implementation of *PLCHound* as well as a rigorous evaluation of its performance using experimentally collected results. We also evaluate *PLCHound*'s ability to aid in various security-oriented tasks.

5.1 Implementation

PLCHound is implemented using 2,339 lines of Python code and utilizes both disk and in-memory caching layers. This section provides a brief description of the software implementation and highlights any notable optimizations and utilization of third-party components.

Scanner Interface. We utilize the popular Censys [1] service as our Internet-wide scanner. Specifically, we use their publically-available Python library on top of a custom disk-cache to prevent unnecessary network requests and credit exhaustion. Note that our asset discovery algorithm is fully agnostic to the underlying scanner, meaning that future researchers can implement variations of *PLCHound* using other readily available scanners such as Shodan or ZoomEye.

Core Algorithm. In the *Keyphrase Extraction* stage, we use the third party word frequency library, *wordfreq* [61] and a domain-specific stopword list that we developed in-house using the top 1,000 responses of the Censys Report Tool [14]. We also use *sklearn* [46] for banner content tokenization and *functools* [29] for caching to avoid repeated tokenizations. In the *Candidate Selection* stage, *sklearn* [46] aids in KMeans clustering and Silhouette Score, while *jenkspy* [38] determines the dynamic score threshold. The *Device Profiling* stage leverages a modified *RegExTractor* [48] for fault-tolerant grammar induction and *rstr* [45] for regex solution space exploration.

Testbed Configuration. The output of *PLCHound* is two-fold - 1) a vast snapshot of diverse hosts that currently contain an online PLC and 2) a list of robust, yet descriptive, re-usable queries. These queries can continue to be used going forward to study the field-deployed population. In fact, since these queries were built using temporally-resistant signatures with highly flexible regexes in areas of high-entropy, they will likely continue to function against future, unseen, firmware versions and user configurations. Thus, the operator will only need to seldom, if ever, re-run the full *PLCHound* algorithm. For our experiments, we ran *PLCHound* on an Apple M1 MacBook Pro (3.2 GHz with 32 GB memory), on which each run took roughly 10 hours to complete.

5.2 Experimental Results

In this section, we experimentally evaluate *PLCHound*'s performance against real-world devices and discuss the accuracy of its output. Specifically, we present *PLCHound*'s results using the naive EIP query from Section 2 as a seed, rigorously determine the FPH rate, and analyze how different seed queries influence the outcome. We also evaluate how *PLCHound* performs on other vendors to show its generalizability across the ICS industry.

Case Study: Automating the Motivating Example. We evaluated *PLCHound* by addressing the motivating question from Section 2.2. In this scenario, a third-party group (e.g., Government Agency or independent researcher) seeks to perform a security analysis to determine how many PLCs are remotely susceptible to a certain recent vulnerability. Prior to our solution, accurately performing this task required deep subject matter expertise and significant manual effort.

Recall that the seed query searches for a PLC model number on a port commonly used for the EIP protocol. After the first iteration of the algorithm, *PLCHound* synthesized 27 new queries, spanning 7 different protocols (FTP, HTTP, HTTPS, SNMP, EIP, CODESYS, and MODBUS), as listed in Table 4 in the Appendix. Note that the robustness of these queries increases over time, as the algorithm is exposed to more examples from which it can generalize patterns and refine the device profile. An example query synthesized from this first iteration is included below.

same_service(extended_service_name="SNMP" AND snmp.oid_system.contact="support@wago.com")

After the second iteration, *PLCHound* synthesized 26 additional queries, several of which were a combination/generalization of prior queries. An example query synthesized from this second iteration is below.

*same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.issuer_dn=/C\=DE, ST\=[A-Z][a-zA-Z]{2,6}, L\=Minden, O\=W[a-zA-Z]{3,3} Kontakttechnik GmbH & Co\. KG, [A-Z\S\s a-z]*CN\=WAGO[\S\s]P[A-Z][A-Z]0,2C[0-9A-Z]{1,3}, emailAddress\=info\@wago\.com/)*

This process continued until *PLCHound* fully mapped out the entire lineage of firmware versions and hundreds of different user settings. In total, the algorithm yielded 456 unique queries and discovered 7,042, online at the time of scan, devices (777%-3,748% more devices than discovered in prior work [35, 70]). Of these 7,042 hosts, 5,258 (74.67%) did not contain the "*SCADA*" Censys label [15], despite belonging to the most quintessential SCADA device (i.e., PLCs). These results further illustrate that traditional approaches to ICS asset discovery are ill-equipped to adequately characterize the complex and fragmented state of the modern field-deployed population. Figure 8 shows a graph of discovered hosts over time. Since many of these queries are refined versions of prior queries, the running total count of devices will not necessarily increase after every iteration (hence the diminishing returns observed in the graph). These relatively flat periods can intuitively be thought about as times when the algorithm was honing the device profile and refining queries to most effectively capture the existing population. Figure 8 is annotated to explicitly list the most effective queries during the life of the algorithm (i.e., queries that added the most previously unknown devices at that time).

To distill the large list of overlapping and potentially redundant queries into a minimal collection for the task at hand (finding HTTP(s)-exposed PLCs), we first filter out all non-HTTP(s) protocol queries. Next, we use Microsoft's Theorem Proving tool, Z3 [22], to perform minimal set coverage analysis. This process isolates the minimal number of queries needed to rebuild the HTTP(s) dataset, thus finding the most robust queries from our list and solving the motiving example without any domain expertise or human interaction.

False Positive Analysis. Ensuring a low false positive rate is essential for the utility of our solution. To confirm this property, we rigorously scrutinize every query and host from our dataset through a series of manual inspections.

First, we use Z3 [22] to remove redundant queries from our synthesized list. Next, we manually inspect all 18 remaining queries to identify potentially ambiguous signatures (i.e., signatures that do not explicitly require the vendor or model number). This step resulted in 10 suspicious queries, listed in Table 5 in the Appendix, that warrant a deeper investigation. The other 8 queries are incapable of matching FPBs, since they require the full vendor name and/or model number to be present in specific fields of the TLS certificate, HTTP response headers, HTML title, or message body.

Next, we gather all hosts that were *only* discovered by suspicious queries (i.e., no high-confidence query was able to "vouch" for it). This process resulted in 127 hosts out of the 7,042 total results. We manually inspected every suspicious host and concluded that 33 were indeed true positives, while the remaining 94 were inconclusive but likely still PLC-related. Therefore, the lower bound for *PLCHound*'s true positive rate was experimentally verified to be 98.67%. Full details regarding the manual inspection process are presented in Section C of the Appendix.

Sensitivity to Seed Queries. The only aspect of *PLCHound* that requires manual effort is the one-time formation of the seed query to guide the initial trajectory of the algorithm. As we demonstrated with the EIP seed query from Section 5.2, this query can be easily constructed via the methodology commonly-used in prior work [6, 27, 39, 70]. In this section, we experimentally validate that the overall outcome of *PLCHound* is largely insensitive to this seed by running the algorithm from scratch using three different initial queries. All three of which include a simple variation of the device's model number and/or device family identifier against a port commonly used for a popular network protocol.

We first examine the seed query from Section 5.2, as written below. This query attempts to find devices that advertise the model number within the main body of the EIP banner. At the time of writing, this simplistic query returns 8 hosts. *PLCHound* used this seed to synthesize 456 additional queries (18 minimally as calculated by Z3) and discovered 7,042 currently online devices.

same_service(port=44818 AND banner:"750-881")

For the second seed query, we attempted to find devices that include the vendor name and model family within the main body of the SNMP banner. This query, included below, discovered 39 devices at the time of writing. *PLCHound* used this seed to synthesize 563 additional queries (20 minimally) and discovered 7,216 currently online devices.

same_service(port=161 AND banner:"WAGO 750")

The final seed query attempted to discover devices with the formal controller name inside the main body of the CODESYS banner, as listed below. At the time of writing, this query found 20 hosts. *PLCHound* used this seed to synthesize 525 additional queries (21 minimally) and discovered 7,227 currently online devices.

same_service(port=2455 AND banner:"pfc200")

All three runs of the algorithm followed a similar path during their journey to find these ~7,200 devices, as shown in Figure 9. Even more compelling, all three resulting datasets share 95.58% of devices, thus confirming that the algorithm reliably converges to the same approximate ground truth, regardless of starting point. Therefore, the operator of *PLCHound* does not need to concern herself with finding an optimal or advanced seed query.

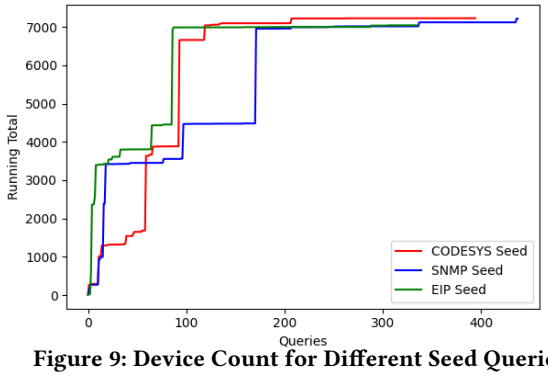

Figure 9: Device Count for Different Seed Queries

Generalizability Across Different Vendors. While WAGO is a prime example of a *modern* PLC vendor (official partner of Amazon Web Service IoT Greengrass [67], Docker compatible runtime [64], Linux-based apps [65], etc.), established ICS vendors such as Allen-Bradley (Rockwell Automation) and OMRON still hold a large market share world-wide [4]. Traditionally, devices by these vendors only offered modest network connectivity (i.e., few protocols and limited features), however, their newer PLCs are starting to rival WAGO in terms of Internet-capable feature sets [63].

Since *PLCHound* leverages the complex network footprint of modern PLCs to automatically map out the fragmented field-deployed population, it inherently works best against vendors that produce highly programmable devices that speak many protocols. That said, *PLCHound* is still able to discover previously unstudied pockets of devices made by all vendors using its automated approach. Similar to the WAGO case study from Section 5.2, we created an EIP-based seed query to capture Allen-Bradley (Rockwell Automation) PLCs.

Without any human-intervention, *PLCHound* used this EIP seed to automatically synthesize highly accurate queries for discovering Allen-Bradley devices speaking HTTP, SNMP, and MODBUS. Notable queries are listed below:

```
same_service(extended_service_name="HTTP" AND
http.response.html_title=/1766-L32[A-Z]2[A-Z]?A
[A-Z]\/[0-9]?1[0-9]?\.0[0-9] /)
```

Note how the above synthesized HTTP-based query has been automatically honed to include flexible regexes in substrings responsible for indicating the firmware version and specific chipset.

Similar to the HTTP-query, the below synthesized SNMP-based query was refined to allow variable firmware versions and chipset identifiers.

```
same_service(extended_service_name="SNMP" AND
snmp.oid_system.desc=/Allen-Bradley 1766-L32[A-Z]2[A-Z]?A
[A-Z]\/[0-9]?1[0-9]?\.0[0-9] MicroLogix1400 Series [A-Z] Revision
[0-9]?1[0-9]?\.[0-9] /)
```

In total, *PLCHound*, was able to find several thousand Allen-Bradley PLC devices that have been left out of prior work (1,228 more than EIP-based prior work [25] and 6,748 more than Modbus-based prior work [6]). We also conducted a similar experiment with OMRON PLCs, again successfully discovering unstudied pockets of the field-deployed population. The full results are presented in Figure 10. We randomly sampled 50 hosts from each vendor and manually confirmed that they are all true positives, thus demonstrating that the low false positive rate calculated in Section 5.2 likely holds true across vendors [1]. Note that it is difficult to meaningfully compare the total host count between different studies for several reasons. The first reason being that most prior work only attempted to find devices that speak a specific protocol, which severely undercounts the true population. Another reason why comparisons are difficult is because snapshots of currently-online devices vary drastically over time and can be heavily skewed by external factors such as the search engine's caching tolerance and scan frequency [11]. In fact, many prior works actually combine the results from multiple search engines, which in effect merges multiple cached records together to produce an aggregate of hosts over time, not an instantaneous snapshot (and may even double-count hosts due to IP-churn). Lastly, many studies did not perform any false-positive analysis and used overly broad queries. Despite these challenges, this comparison, while imperfect, is still a useful metric for gauging how our solution could enhance other research projects.

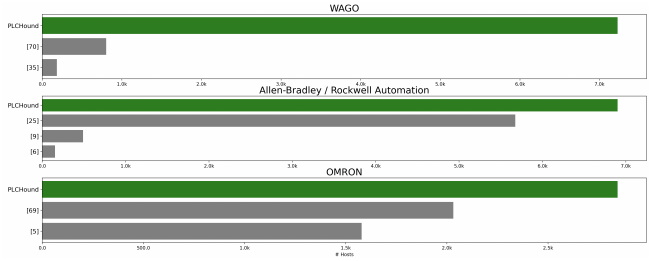

Figure 10: PLCHound with three different vendors

These results demonstrate that the traditional ICS asset discovery method of using a simple keyword against a single protocol fails to capture the complex and fragmented field-deployed population. The effectiveness of *PLCHound* to find elusive devices using nuanced network signatures will only increase over time as established vendors continue to embrace the emerging industrial IoT connectivity trend.

5.3 Security Applications

We can use the rich dataset of devices generated by *PLCHound* to answer interesting security questions that prior work struggles to accurately assess. Since most prior work uses simplistic manually-developed queries to find devices that speak a single specific protocol, they inherently contain a bias that may skew their analysis. As demonstrated in Section 5.2, our approach converges on a ground truth snapshot of devices, regardless of the protocol used in the seed query. Thus, our dataset represents a more objective truth that provides insight into how these devices actually manifest in practice. In this section, we use the results from the motivating example to address four unique security applications.

Unstudied Vulnerable Populations. The Purdue Enterprise Reference Architecture (PERA) dictates that level 1 controllers (e.g., PLCs) be separated from non-critical zones such as insecure business networks [68]. Under this model, network isolation prevents untrusted entities from communicating with PLCs. Unfortunately, as we have seen in this paper, real-world ICSs oftentimes violate this standard and accidentally expose their PLCs to attackers via the public Internet.

Prior work has used datasets of PERA-violating PLCs to perform remote vulnerability analysis [70], ransomware susceptibility models [25], and proactive outreach campaigns [55]. As we demonstrated in Section 5.2, the traditional approach to ICS asset discovery (i.e., manually chosen static string against a commonly used ICS port) can drastically undercount the true field-deployed population. The sheer magnitude of devices unintentionally left-out of these studies suggests that prior work may have been less comprehensive than originally thought. Our results with *PLCHound* indicate that there is a vast population of unstudied vulnerable devices. This

[1] Supported by a 95% confidence interval, as calculated in Section D of the Appendix.

data serves as a reminder that ICSs should always change default passwords and keep firmware up-to-date regardless of assumed network protections, since true airgaps prove challenging to deploy and maintain in practice.

Field-Deployed PLC Attack Surface. The diverse dataset produced by *PLCHound* allows us to accurately characterize the true field-deployed PLC attack surface. Towards this goal, we review the collection of hosts discovered in Section 5.2 and discuss insights provided by studying their PLC-originating banners. Specifically, we analyze which protocols and ports are most commonly exposed and use them to postulate upstream industrial firewall rules. The top five most common PLC-originating protocols, in decreasing order, are: HTTP (73.98%), CODESYS (29.79%), HTTPS (28.78%), MODBUS (12.32%), and SSH (11.35%). The majority of hosts only expose a single PLC-originating protocol (56.61%), followed by 26.00% exposing two, 9.17% exposing three, 6.84% exposing four, and less than 2% exposing five or more. This tendency can be visually observed using the top five PLC protocols in Figure 11.

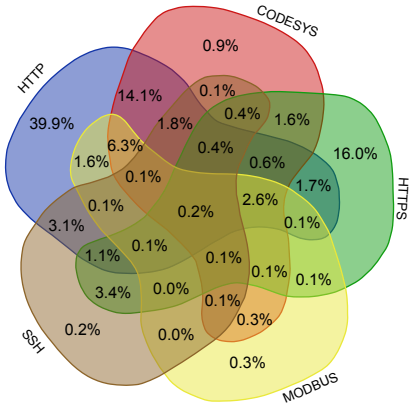

Figure 11: Field-Deployed PLC Attack Surface

This data offers an intuitive explanation for why *PLCHound* is able to find so many more devices than prior work - a small number of devices are extremely exposed, causing them to be picked up by trivial queries, while the vast majority of devices are only partially exposed, thus can only be picked up by advanced queries. A deeper investigation into these exposed protocols reveal that many services are using a non-standard port. This practice appears to vary substantially by protocol, with 76.47% of HTTP not using port 80, 61.15% of SSH not using port 22, 31.68% of HTTPS not using port 443, 31.07% of FTP not using port 21, and 0% of EIP, MODBUS, or CODESYS using a non-standard port, as shown in Table 1. This observation strongly suggests that networking devices within ICS plants (e.g., industrial routers, switches, gateways) often perform port-forwarding on the standard IT protocols, however tend to leave the ICS-specific protocols untouched.

We can also use this data to speculate about real-world industrial firewall configurations. With the exception of CODESYS (a closed-source proprietary protocol), the majority of devices are only exposing protocols common in a standard IT office environment (HTTP, HTTPS, SSH). This disproportional representation of IT

Table 1: Non-Standard Port usage of PLC Protocols

Protocol	Standard Port	Percent Compliant
HTTP	80	23.53%
SSH	22	38.85%
HTTPS	443	68.32%
FTP	21	68.93%
EIP	44818	100%
MODBUS	502	100%
CODESYS	2455	100%

protocols in our dataset, combined with the lack of port-forwarding on ICS protocols, suggests that many industrial firewalls are likely blocking ingress traffic on ports commonly used for ICS (e.g., 502 and 44818), however fail to block traffic on other ports. Unfortunately, this is an inadequate protection for modern PLC devices, since they also speak (and can be attacked using) a plethora of non-ICS protocols.

Unfortunately, this inadequate network isolation results in 95.88% of identified WAGO PLCs to expose either HTTP or HTTPS, causing them to be vulnerable to CVE-2022-45140. Additionally, the latest firmware as of the time of writing is also vulnerable to CVE-2022-45137, CVE-2022-45138, and CVE-2022-45139 via the HTTP(s) protocols. As ICS vendors continue to add webserver functionality to their devices (an emerging trend revealed by prior work [47]), the exploitability via non-ICS protocols will only continue to rise. This analysis reveals the urgent need for ICS facilities to adapt a more modern approach to network isolation to protect all PLC-originating protocols.

Non-PLC Attacker Entry Points. Interestingly, studying the inverse of Section 5.3 (i.e. the non-PLC attack surface within these hosts) also provides fascinating insight into the real-world manifestation of ICS plants. We can again review the collection of hosts discovered in Section 5.2, except this time review the banners that *did not* originate from the PLC. This analysis reveals co-located devices that may be potential entry points into the privileged ICS network. Once behind the industrial firewall, an adversary could potentially pivot (or simply send malicious instructions) to the PLC using an unauthenticated real-time protocol to attack the ICS [2].

The most common PLC co-located protocols, in decreasing order, are: HTTP (65.07%), HTTPS (26.15%), IKE (10.33%), FTP (8.95%), SSH (6.05%), and PPTP (3.82%). A best-effort manual inspection reveals that the most common shared components that engender these banners are the PowerLogic Power Meter [56] (present in 8.40% of all hosts), Four-Faith Industrial Router [30] (6.60%), Pure-FTPd [50] (4.75%), OpenSSH [50] (3.90%), and Fritz Box Router [7] (3.45%).

Using this data, we perform a brief security analysis to determine which non-PLC known vulnerabilities impact the largest percentage of PLC-owning hosts. We specifically focus on "high" (CVSS v3.0 > 7.0) or greater Remote Code Execution (RCE) vulnerabilities, since this is the type of issue best suited for infiltrating the network. To extract version numbers, we either rely on the data already present in the collected banner (passive) or send a single unauthenticated HTTP(s) GET request (active). We identified that 7.30% of hosts are vulnerable to CVE-2021-22713 (PowerLogic ION7650 FW<V416), 3.41% are vulnerable to CVE-2023-38408 (OpenSSH V5.5-9.3), and

[2]Additional pivoting steps inside the network may be required, depending on LAN topology and trust relationships.

0.26% are vulnerable to CVE-2019-12168 (Four-Faith Wireless Mobile Router F3x24 v1.0), as summarized in Table 2. Collectively, these three vulnerabilities cover 10.15% of all hosts in our dataset.

Table 2: PLC and Co-located CVEs

CVE ID	CVSS	Impacted Device	Percent Vuln.
CVE-2022-45137	6.1	PLC	95.88%
CVE-2022-45138	9.8	PLC	95.88%
CVE-2022-45139	5.3	PLC	95.88%
CVE-2022-45140	9.8	PLC	95.88%
CVE-2021-22713	7.5	Power Meter	7.30%
CVE-2023-38408	9.8	SSH Server	3.41%
CVE-2019-12168	7.2	Industrial Router	0.26%

This data shows that it is feasible for adversaries to use known vulnerabilities impacting other, non-PLC, services to potentially circumvent network-based defenses and gain privileged access to the industrial plant, and perhaps even the PLC itself.

Industrial End-User Security Choices. The holistic multi-protocol dataset produced by *PLCHound* allows us to gain insight into how PLC customers actually configure their devices in practice. By analyzing these configuration settings, we can ascertain the level of security consciousness demonstrated by customers. To evaluate this property, we analyze three different types of PLC communication, namely *web*, *file transfer*, and *remote shell*, each of which offer a plaintext and an encrypted option. While the encrypted option provides security benefits, it can also be more difficult to configure and manage, especially in ICS environments where third party Certificate Authorities (CA) are oftentimes unreachable. Therefore, we consider any intentional effort to support encryption to be a conscious security-minded choice.

The first method of communication is *web*, which, in the context of PLCs, enables customers to access onboard administrative webpages and web-based HMIs. Of these customers, 74.45% chose HTTP and 25.54% chose HTTPS. The second method of PLC communication is *file transfer*, which is typically used to retrieve operational log files and perform other administrative tasks. Of these customers, 55.19% chose FTP and 44.81% chose FTPes. The final method of communication is *remote shell*, which allows customers to execute arbitrary OS-level commands within the controller for maintenance and debugging purposes. Of these customers, 89.15% chose SSH and 10.85% chose Telnet. Table 3 summarizes the customer choices for each communication type.

Table 3: PLC User Security Choices

Communication	Percent Plaintext	Percent Encrypted
Web	74.45%	25.54%
File Transfer	55.19%	44.81%
Remote Shell	10.85%	89.15%

These results reveal that most customers prefer to use the unencrypted variations of protocols to access *web* and *file transfer*, however, prefer to use the encrypted method for accessing *remote shell*. After examining the PLC user experience closely in our lab, we offer a potential explanation for this behavior. Both HTTPS and FTPes utilize the SSL/TLS Handshake, where a digital certificate is presented to the client (typically a web browser and file explorer). In a normal IT environment, this certificate is verified by a third-party CA, so the client can confirm the connection is safe. However, in an ICS environment, these embedded devices use self-signed certificates that, by-default, invoke an error message when the client attempts to connect. Depending on the client application, dismissing these errors can be non-trivial and may require disabling security settings. SSH encryption, on the other hand, does not rely on a third party CA, so no error or warning message is ever displayed to the client. This hypothesis is in-line with prior work studying the impact of customer friction on the adoption of security mechanisms [32].

6 Discussion

In this section, we discuss the scope of *PLCHound*'s scans, inherent limitations with its approach, and other possible uses for our solution. We also address how we handled ethical considerations when building and verifying *PLCHound*.

6.1 Scope, Limitations, & Future Work

Same as prior works [6], *PLCHound* attempts to find the entire family of devices made by a particular vendor. In practice, these devices are primarily PLCs and PLC-adjacent devices, such as hardware add-ons and local HMIs. All of these devices are typically relevant to security analyses, since they tend to share firmware code and can be impacted by the same vulnerabilities. Note that while our solution does not automatically exclude honeypots, prior work provided methods to accurately identify common honeypot frameworks if desired [25].

It is important to acknowledge that *PLCHound*'s conservative approach to query synthesis (i.e., requiring highly-descriptive signatures) may result in hosts that only expose nondescript banners to be left out of the dataset. These FNHs are indistinguishable from TNHs and therefore are unable to be discovered using our queries. For example, some PLCs use the popular embedded-system-compatible third party component, Dropbear [13], as their SSH server software. Unfortunately, since all Dropbear banners are identical and many irrelevant embedded devices also use Dropbear, *PLCHound* will be unable to craft a query to identify PLCs that *only* expose SSH. This limitation is not unique to *PLCHound* and also impacts manual asset discovery.

While *PLCHound* was built using observations about how ICSs manifest in practice, it is possible that our solution could be adapted to discover non-ICS devices, so long as they speak multiple protocols and have a large fragmented network footprint. Possible candidates for future work include household IoT devices (e.g., smart doorbells) and Internet-connected vehicles (e.g., automobiles, marine vessels). We hope that future researchers can utilize our robust entity resolution methodology, perhaps with slight modifications to our stopword list and cluster scoring algorithim, to uncover previously unstudied populations of devices in other domains.

6.2 Ethics & Responsible Disclosure

We approached this project with a strong belief that visibility into exposed ICS devices is crucial to securing public infrastructure. With this mission in mind, we created *PLCHound* to enhance sanctioned and ethical security research. Given that publicly available scan datasets, offered by services such as Shodan and Censys, are already widely used in academic works [6, 27, 39, 70], we hope that

our solution can directly aid in ongoing projects to improve the security posture of industrial systems. That being said, we understand that identifying exposed and vulnerable ICS devices raises ethical concerns that need to be addressed.

The first point we would like to emphasize is that *PLCHound* is fully passive and offline. Since our solution relies on an existing database of scan data, there is no risk that running our tool could cause any direct harm to real-world systems. However, as part of the manual inspection process from Section 5.2, we performed one-time active probing using the best-practice safeguards recommended by prior work [62]. To ensure that our probing process did not interfere with any live systems, we limited our testing to only sending a single non-mutating request to a non-critical protocol (i.e. HTTP GET) and did not perform any authentication attempts. Full details regarding the caution exercised during active probing can be found in Appendix C.

We are also taking steps to reduce the possibility of *PLCHound* misuse by restricting access to our software implementation. Like all Internet measurement tools, such as ZMap [26] and custom web-crawlers [37], there is an inherent risk that bad actors could use the solution to identify potential victims for exploitation. While the risk of malicious utilization of Shodan and Censys existed prior to our tool, we acknowledge that *PLCHound* removes certain barriers to entry, such as domain-expertise and engineering effort, that may have prevented wide-spread abuse. Therefore, we decided to not release *PLCHound*'s code, and to instead, only provide output queries to vetted academic researchers upon request.

Finally, we performed an extensive outreach campaign to notify the ICSs discovered in Section 5.3 that their PLCs were publicly-facing with known vulnerabilities. We utilized the third-party service, *IPInfo* [3], to collect contact information and conducted a large-scale email disclosure campaign through our local Office of Information Technology (OIT). Although we cannot definitively attribute end-user actions to our notification efforts, a follow-up scan, performed one month after the outreach, showed that 34.3% of the contacted IP addresses no longer exposed a PLC device. These results are in-line with prior work on the effectiveness of outreach campaigns [40]. Lastly, we co-authored a security advisory with the vendor of the targeted PLC to inform the community about the underestimated dangers of network misconfigurations [66].

7 Conclusion

In this paper, we present the first method for automatic ICS asset discovery using active scan data. We experimentally evaluated that our solution, *PLCHound*, is able to accurately infer the existence of ICS devices using subtle indicators and temporally-resistant signatures with a 98.67% true positive rate. The fully-automated query synthesis approach allows *PLCHound* to dig through large datasets of scan data collected by scanners such as Censys and Shodan without any domain expertise or human interaction to find up to 37x more devices than prior work. We use this data to perform a holistic field-deployed security analysis and reveal that ICS facilities are indeed highly vulnerable and readily-accessible. As ICS vendors continue to embrace the industrial IoT connectivity trend, our automated approach becomes even more invaluable [3].

[3] An extended version of this paper is available at https://ryanpickren.com/plchound.

References

[1] 2023. Censys Search. https://search.censys.io/. Accessed: 2023-08-17.

[2] 2023. Shodan. https://www.shodan.io/. Accessed: 2023-08-17.

[3] 2024. ipinfo.io. https://ipinfo.io/.

[4] Arizton Advisory and Intelligence. [n. d.]. PLC Market - Global Outlook and Forecast 2020-2025. https://www.arizton.com/market-reports/plc-market-analysis.

[5] Simon Daniel Duque Anton, Daniel Fraunholz, Daniel Krohmer, Daniel Reti, Daniel Schneider, and Hans Dieter Schotten. 2021. The global state of security in industrial control systems: An empirical analysis of vulnerabilities around the world. *IEEE Internet of Things Journal* 8, 24 (2021), 17525–17540.

[6] Travis Ashley, Sri Nikhil Gupta Gourisetti, Newton Brown, and Christopher Bonebrake. 2022. Aggregate attack surface management for network discovery of operational technology. *Computers & Security* 123 (2022), 102939.

[7] AVM. [n. d.]. FritzBox | Our top models. https://en.avm.de/products/fritzbox/.

[8] Tim Berners-Lee and Daniel W. Connolly. 1995. Hypertext Markup Language - 2.0. https://www.rfc-editor.org/info/rfc1866. https://doi.org/10.17487/RFC1866

[9] Roland Bodenheim, Jonathan Butts, Stephen Dunlap, and Barry Mullins. 2014. Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices. *International Journal of Critical Infrastructure Protection* 7, 2 (2014), 114–123.

[10] Tim Bray. 2017. The JavaScript Object Notation (JSON) Data Interchange Format. https://www.rfc-editor.org/info/rfc8259. https://doi.org/10.17487/RFC8259

[11] Censys. [n. d.]. Censys Internet Scanning Intro. https://support.censys.io/hc/en-us/articles/360059603231-Censys-Internet-Scanning-Intro.

[12] Censys. [n. d.]. Data Definitions. https://search.censys.io/search/definitions?resource=hosts.

[13] Censys. [n. d.]. Dropbear SSH. https://matt.ucc.asn.au/dropbear/dropbear.html.

[14] Censys. [n. d.]. Report on Hosts. https://search.censys.io/search/report?resource=hosts.

[15] Censys. [n. d.]. Search with Labels. https://support.censys.io/hc/en-us/articles/13446586006292-Search-with-Labels.

[16] Centre for the Protection of National Infrastructure. [n. d.]. Firewall Deployment for SCADA and Process Control Systems. https://www.energy.gov/sites/prod/files/Good%20Practices%20Guide%20for%20Firewall%20Deployment.pdf.

[17] Joao M Ceron, Justyna J Chromik, Jair Santanna, and Aiko Pras. 2020. Online discoverability and vulnerabilities of ICS/SCADA devices in the Netherlands. *arXiv preprint arXiv:2011.02019* (2020).

[18] CISA. [n. d.]. NERC Critical Infrastructure Protection (NERC CIP). https://niccs.cisa.gov/education-training/catalog/captiva-solutions-llc/nerc-critical-infrastructure-protection-nerc-cip.

[19] CODESYS. [n. d.]. CODESYS WEBVISU. https://www.codesys.com/products/codesys-visualization/webvisu.html.

[20] Cybersecurity Infrastructure Security Agency. 2023. Exploitation of Unitronics PLCs used in Water and Wastewater Systems. https://www.cisa.gov/news-events/alerts/2023/11/28/exploitation-unitronics-plcs-used-water-and-wastewater-systems.

[21] Daniel Stenberg. 2022. curl: Command Line Tool and Library for Transferring Data with URLs. https://curl.se/.

[22] Leonardo de Moura and Nikolaj Bjørner. 2023. Z3 Theorem Prover. https://github.com/Z3Prover/z3.

[23] Defense Use Case. 2016. Analysis of the cyber attack on the Ukrainian power grid. (2016), 1–29.

[24] Alessandro Di Pinto, Younes Dragoni, and Andrea Carcano. 2018. TRITON: The first ICS cyber attack on safety instrument systems. In *Proc. Black Hat USA*. 1–26.

[25] Michael Dodson, Alastair R Beresford, and Daniel R Thomas. 2020. When will my PLC support Mirai? The security economics of large-scale attacks against Internet-connected ICS devices. In *2020 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 1–14.

[26] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2013. {ZMap}: fast internet-wide scanning and its security applications. In *22nd USENIX Security Symposium (USENIX Security 13)*. 605–620.

[27] Ismail Erkek and Erdal Irmak. 2021. Cyber security of internet connected ics/scada devices and services. In *2021 International Conference on Information Security and Cryptology (ISCTURKEY)*. IEEE, 75–80.

[28] Roy T. Fielding and Julian Reschke. 2014. Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. https://www.rfc-editor.org/info/rfc7230. https://doi.org/10.17487/RFC7230

[29] Python Software Foundation. 2023. functools — Higher-order functions and operations on callable objects. https://docs.python.org/3/library/functools.html.

[30] Four-Faith. [n. d.]. Four-Faith. https://www.fourfaith.com/.

[31] Thomas Hanka, Matthias Niedermaier, Florian Fischer, Susanne Kießling, Peter Knauer, and Dominik Merli. 2021. Impact of active scanning tools for device discovery in industrial networks. In *Security, Privacy, and Anonymity in Computation, Communication, and Storage: SpaCCS 2020 International Workshops, Nanjing, China, December 18-20, 2020, Proceedings 13*. Springer, 557–572.

[32] Cormac Herley. 2009. So long, and no thanks for the externalities: the rational rejection of security advice by users. In *Proceedings of the 2009 workshop on New security paradigms workshop*. 133–144.

[33] George F Jenks. 1967. The data model concept in statistical mapping. *International yearbook of cartography* 7 (1967), 186–190.

[34] Kaspersky. [n. d.]. Industrial Control Systems and Their Online Availability. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/07/07190427/KL_REPORT_ICS_Availability_Statistics.pdf.

[35] Anastasis Keliris and Michail Maniatakos. 2016. Remote field device fingerprinting using device-specific modbus information. In *2016 IEEE 59th international Midwest symposium on circuits and systems (MWSCAS)*. IEEE, 1–4.

[36] Ralph Langner. 2011. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy* 9, 3 (2011), 49–51.

[37] Tobias Lauinger, Abdelberi Chaabane, Sajjad Arshad, William Robertson, Christo Wilson, and Engin Kirda. 2018. Thou shalt not depend on me: Analysing the use of outdated javascript libraries on the web. *arXiv preprint arXiv:1811.00918* (2018).

[38] Mathieu Leplatre. 2023. jenkspy: Optimal Jenks-Caspall Natural Breaks classification in pure Python. https://github.com/mthh/jenkspy.

[39] Eireann P Leverett. 2011. Quantitatively assessing and visualising industrial system attack surfaces. *University of Cambridge, Darwin College* 7 (2011), 21.

[40] Frank Li, Zakir Durumeric, Jakub Czyz, Mohammad Karami, Michael Bailey, Damon McCoy, Stefan Savage, and Vern Paxson. 2016. You've got vulnerability: Exploring effective vulnerability notifications. In *25th USENIX Security Symposium (USENIX Security 16)*. 1033–1050.

[41] James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Vol. 1. Oakland, CA, USA, 281–297.

[42] Ariana Mirian, Zane Ma, David Adrian, Matthew Tischer, Thasphon Chuenchujit, Tim Yardley, Robin Berthier, Joshua Mason, Zakir Durumeric, J Alex Halderman, et al. 2016. An internet-wide view of ics devices. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 96–103.

[43] Tin Q Nguyen, Sage E Pickren, Neena M Saha, and Laurie E Cutting. 2020. Executive functions and components of oral reading fluency through the lens of text complexity. *Reading and writing* 33 (2020), 1037–1073.

[44] nmap.org. [n. d.]. NMAP. https://nmap.org/.

[45] Leapfrog Online. 2023. rstr: Random string module for Python. https://github.com/leapfrogonline/rstr.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[47] Ryan Pickren, Tohid Shekari, Zonouz Saman, and Raheem Beyah. 2024. Compromising Industrial Processes using Web-Based Programmable Logic Controller Malware. In *NDSS*. 1–18.

[48] Iuliu Popovici. 2023. RegExTractor: A simple tool to extract regular expressions from text. https://github.com/iuliux/RegExTractor.

[49] Davide Pozza, Riccardo Sisto, Luca Durante, and Adriano Valenzano. 2006. Comparing lexical analysis tools for buffer overflow detection in network software. In *2006 1st International Conference on Communication Systems Software & Middleware*. IEEE, 1–7.

[50] Pure-FTPD. [n. d.]. Pure-FTPD. https://www.pureftpd.org/project/pure-ftpd/.

[51] Bob Radvanovsky. [n. d.]. Project SHINE: 1,000,000 Internet-Connected SCADA and ICS Systems and Counting. https://www.tofinosecurity.com/blog/project-shine-1000000-internet-connected-scada-and-ics-systems-and-counting.

[52] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[53] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[54] Serhad Sarica and Jianxi Luo. 2021. Stopwords in technical language processing. *Plos one* 16, 8 (2021), e0254937.

[55] Takayuki Sasaki, Akira Fujita, Carlos H Ganán, Michel van Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. 2022. Exposed infrastructures: Discovery, attacks and remediation of insecure ics remote management devices. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2379–2396.

[56] Schneider. [n. d.]. PowerLogic Energy Meters. https://www.se.com/us/en/product-range/1717-powerlogic-energy-meters/.

[57] scikit. [n. d.]. Tokenizing text with scikit-learn. https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html#tokenizing-text-with-scikit-learn.

[58] Yakov Shafranovich. 2005. Common Format and MIME Type for Comma-Separated Values (CSV) Files. https://www.rfc-editor.org/info/rfc4180. https://doi.org/10.17487/RFC4180

[59] Zain Shamsi, Ankur Nandwani, Derek Leonard, and Dmitri Loguinov. 2014. Hershel: single-packet os fingerprinting. *ACM SIGMETRICS Performance Evaluation Review* 42, 1 (2014), 195–206.

[60] Shodan. [n. d.]. Shodan Explore | Industrial Control Systems. https://www.shodan.io/explore/category/industrial-control-systems.

[61] Robyn Speer. 2022. *rspeer/wordfreq: v3.0.* https://doi.org/10.5281/zenodo.7199437

[62] Mike Thelwall and David Stuart. 2006. Web crawling ethics revisited: Cost, privacy, and denial of service. *Journal of the American Society for Information Science and Technology* 57, 13 (2006), 1771–1779.
[63] PLC Technician Training. [n. d.]. Latest Advancement in PLC Technology. https://www.plctechnician.com/news-blog/latest-advancement-plc-technology.
[64] WAGO. [n. d.]. Container Virtualization with Docker. https://www.wago.com/us/open-automation/modular-software/linux/docker.
[65] WAGO. [n. d.]. Control Included: Embedded Linux. https://www.wago.com/us/embedded-linux.
[66] WAGO. [n. d.]. High Number of Unreported Errors in Controllers Accessible via the Internet. https://www.wago.com/global/open-automation/cybersecurity/georgia-institute-warns-about-underestimated-risks.
[67] WAGO. [n. d.]. WAGO Products Join AWS Partner Device Catalog and AWS IoT Greengrass. https://www.wago.com/us/aws-partner.
[68] Timothy Williams. 1998. The Purdue enterprise reference architecture and methodology (PERA). *Handbook of life cycle engineering: concepts, models, and technologies* 289 (1998).
[69] Yixiong Wu, Jianwei Zhuge, Tingting Yin, Tianyi Li, Junmin Zhu, Guannan Guo, Yue Liu, and Jianju Hu. 2021. From Exposed to Exploited: Drawing the Picture of Industrial Control Systems Security Status in the Internet Age.. In *ICISSP*. 237–248.
[70] Binbin Zhao, Shouling Ji, Wei-Han Lee, Changting Lin, Haiqin Weng, Jingzheng Wu, Pan Zhou, Liming Fang, and Raheem Beyah. 2020. A large-scale empirical study on the vulnerability of deployed IoT devices. *IEEE Transactions on Dependable and Secure Computing* 19, 3 (2020), 1826–1840.
[71] ZoomEye. [n. d.]. ZoomEye. https://www.zoomeye.org/.

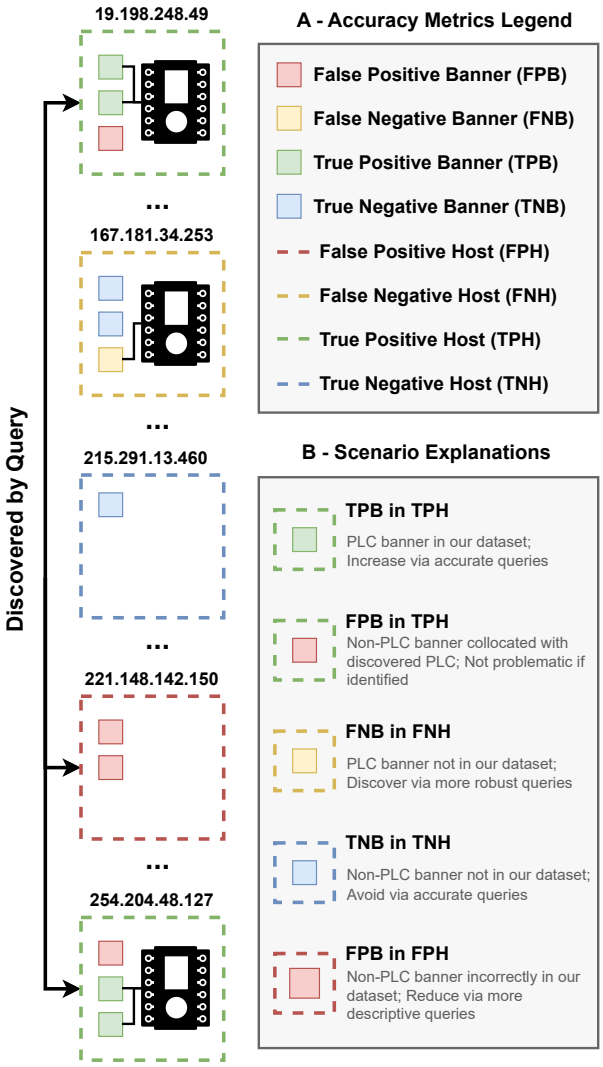

Figure 12: Accuracy Terminology

A Cluster Scoring Details

The first feature ("trustworthiness") was built from the intuition that datapoints discovered by high-confidence queries are more trustworthy than datapoints discovered by medium-confidence queries. In essence, we want a method for rewarding non-matched banner components that have been clustered together with highly-matched banner components. To achieve this goal, we first assign queries a numeric confidence value, with the seed query being assigned 1.0 and future queries assigned via a process described in Section 3. To incorporate this query score into the cluster, we simply take an average across all queries that found the enclosed datapoints.

The second feature ("entropy") was built from the observation that banners that originated from the PLC tend to only contain subtle differences (within the same "type"), where banners from miscellaneous other devices tend to vary drastically. We utilize the cluster's standard deviation value to incorporate this feature.

These two features are then linearly combined with the averages of the Match, Cross-Protocol, and Super lexical analysis scores. The coefficients for this linear combination were experimentally discovered via manual ranking and linear regression. The final result is then normalized so that a 1 indicates a high likelihood of belonging to the PLC and a 0 indicates a low likelihood.

B Cyclical Behavior

PLCHound's intentionally conservative approach to query synthesis often causes new queries to only capture a slightly larger population than previous queries. As a result of this design, the running total collection of hosts tends to grow slowly over time, especially as *PLCHound* refines areas of the device profile with high entropy (e.g., the public IP address within the HTTP request URI banner component).

Since several of *PLCHound*'s calculations are computationally expensive (e.g., component clustering and grammar induction), we choose to only repeat the core algorithm once the running total has significantly changed and the results are likely to be meaningfully different. Thus, we sort the query queue by score to ensure the best queries are near to the top of the list, and pop off as many queries as needed (which may or may not have been synthesized this iteration) to increase the running total host count by at least 10%.

The core algorithm then repeats, with new keywords being extracted and new candidates being selected. This process exposes the algorithm to previously unseen examples, from which it can generalize additional patterns while honing the device profile. As this cycle continues, *PLCHound* learns more and more about the field-deployed population, enabling it to create advanced queries that reveal previously unstudied pockets of field-deployed ICS devices.

During this iterative process, *PLCHound* will automatically refine its queries to include the entire lineage of firmware versions, various user settings, and ultimately, highly-correlated peripheral devices such as hardware add-ons. Eventually, *PLCHound* will uncover all available signatures from the real-world population, at which point the algorithm halts, and the user can examine the comprehensive collection of devices.

C Manual True Positive Analysis

After we isolated the 127 hosts that warranted a deeper investigation, we used manual inspection to verify their identity. Figure 12.A visually illustrates how accuracy metrics relate to banners and hosts in our dataset. The various banner/host combinations that occur during the algorithm are explained in Figure 12.B.

First, we then leveraged Censys's History API to check if these hosts recently contained an unambiguous signature that is no longer online due to external factors (e.g., customer reconfiguration, updated firewalls rules, etc.). Specifically, we look for the vendor name and/or model number in any banner within the past 30 days. If this is found, then we can say with high confidence that these hosts are indeed true positives. This step identified 22 of the 127 suspicious hosts as confirmed true positives. The remaining 105 hosts (1.49%) without such a signature are considered potential false positives that require additional scrutiny.

Table 4: Queries Synthesized after First Iteration

Query (Truncated)
same_service(extended_service_name="FTP" AND transport_fingerprint.raw="16000,64,true,M,1452,false,false")
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.issuer.email_address="info@wago.com")
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.subject_dn="C=DE, ST=NRW, L=Minden, O=WAGO Kontakttechnik GmbH & Co. KG, OU=WAGO Product Lifecycle Management Group, CN=750-88x Server Certificate, emailAddress=info@wago.com")
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.subject.organization="WAGO Kontakttechnik GmbH & Co. KG")
same_service(extended_service_name="HTTP" AND http.response.headers:(key:Server AND value.headers:"WAGO_Webs"))
same_service(extended_service_name="HTTPS" AND http.response.body="... <title>WAGO Ethernet Web-Based Management </title>\r\n </head>\r\n <frameset rows=\"150,*\">\r\n <noframes>\r\n <body>\r\n Your browser does not support frames. Fallback to simple ...")
same_service(extended_service_name="HTTPS" AND banner="HTTP/1.1 200 OK\r\nLast-Modified: Sat Jan 1 01:00:00 2000\r\nServer: WAGO_Webs\r\nContent-Type: text/html\r\nContent-Length: 661\r\n")
same_service(extended_service_name="SNMP" AND snmp.oid_system.contact="support@wago.com")
same_service(extended_service_name="SNMP" AND snmp.oid_system.desc="WAGO 750-881 PFC ETHERNET")
same_service(extended_service_name="HTTP" AND http.response.html_title=" WAGO Ethernet Web-Based Management ")
same_service(extended_service_name="HTTPS" AND http.response.html_tags="<title>WAGO Ethernet Web-Based Management </title>")
same_service(extended_service_name="EIP" AND banner=/cB\<[\S\s][\S\sA-Z]\(q[\S\s]4[A-Za-z][\S\s]WAGO 750-881 PFC ETHERNET/)
same_service(extended_service_name="EIP" AND banner=/.*750-881.*/)
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.subject.email_address="info@wago.com")
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.issuer.organization="WAGO Kontakttechnik GmbH & Co. KG")
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.issuer.common_name="WAGO PLM CA")
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.subject.organizational_unit="WAGO Product Lifecycle Management Group")
same_service(extended_service_name="HTTP" AND http.response.html_tags="<title>WAGO Ethernet Web-Based Management </title>")
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.issuer.organizational_unit="WAGO Product Lifecycle Management Group")
same_service(extended_service_name="EIP" AND banner="c...\u0001\t4\u0000\\\ufffd!\ufffd\u001aWAGO 750-881 PFC ETHERNET\u0000\ufffd")
same_service(extended_service_name="CODESYS" AND transport_fingerprint.raw="16000,64,true,M,1452,false,false")
same_service(extended_service_name="HTTPS" AND http.response.html_title=" WAGO Ethernet Web-Based Management ")
same_service(extended_service_name="EIP" AND transport_fingerprint.raw="16000,64,true,M,1452,false,false")
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.issuer_dn="C=DE, ST=NRW, L=Minden, O=WAGO Kontakttechnik GmbH & Co. KG, OU=WAGO Product Lifecycle Management Group, CN=WAGO PLM CA, emailAddress=info@wago.com")
same_service(extended_service_name="MODBUS" AND transport_fingerprint.raw="16000,64,true,M,1452,false,false")
same_service(extended_service_name="HTTP" AND banner="HTTP/1.1 200 OK\r\nLast-Modified: Sat Jan 1 01:00:00 2000\r\nServer: WAGO_Webs\r\nContent-Type: text/html\r\nContent-Length: 661\r\n")
same_service(extended_service_name="HTTPS" AND http.response.headers:(key:Server AND value.headers:"WAGO_Webs"))

Unfortunately, the validity of the remaining hosts cannot be verified using the passive techniques thus far (i.e., relying on an existing third-party network scan dataset). Thus, we must actively probe these hosts ourselves to induce a more descriptive signature. We approach this task with extreme caution, since these devices are likely controlling real-world equipment in live industrial plants.

Of the 105 potential false positive hosts remaining, the top 5 most commonly-used protocols, in decreasing order, were CODESYS, HTTP, FTP, MODBUS, and HTTPS. We conservatively considered the banners behind these services to be nondescript because they did not explicitly list the vendor name or model number, however they still contained reasonable indicators of the target device. Below is a breakdown of what was present in each banner type.

- **CODESYS**: A cryptographic signature (Hershel+ [59]) indicating a specific operating system is running a CODESYS Server
- **HTTP**: A URI path indicative of the third-party WebVisu [19] web-based HMI software (used by several different PLC vendors)
- **FTP**: A cryptographic signature (Hershel+ [59]) indicating a specific operating system is running a Nucleus RTOS FTP Server
- **MODBUS**: A cryptographic signature (Hershel+ [59]) indicating a specific operating system is running a MODBUS Server
- **HTTPS**: A URI path indicative of the third-party WebVisu [19] web-based HMI software (used by several different PLC vendors)

While we can confidently deduce that all 105 ambiguous hosts do indeed belong to some sort of ICS embedded device, we cannot say with 100% certainty that they fall in the scope of our original scan (i.e. the family of devices made by the target vendor). Thus, we must somehow induce a more descriptive signature from these hosts. For ethical reasons, we limited ourselves to only testing the protocols that are not used for real-time critical operations (to avoid potentially interrupting any essential tasks [31]). We also chose to not attempt to authenticate against any of these devices, even though doing so may confirm their identity (e.g., using the default WAGO FTP password). After much deliberation, we decided that sending a single unauthenticated HTTP(s) GET request to a known-path URI was a solid compromise between inducing a descriptive signature and being relatively unintrusive (since these devices are undoubtably already receiving similar traffic from common web-crawlers). We also included contact information for our project in the HTTP User-Agent header, however we did not receive any inquisitions, likely due to the innocuous nature of the probe.

Specifically, we took the 45 out of the remaining 105 hosts that exposed either HTTP or HTTPS and sent a single GET request to the "*/wbm*" URI path. This path typically contains the homepage for the default "Web Based Management" portal website, assuming it was not disabled by the customer or otherwise blocked by external factors (e.g., webapp firewall). Of these 45 hosts, 11 responded with

Table 5: 18 Minimal Queries Synthesized during WAGO PLC Case Study w/ EIP-based Seed Query

Query (truncated)	Suspicious (Y/N)
same_service(extended_service_name="FTPes" AND tls.certificates.leaf_data.issuer.organization="Wago Kontakttechnik GmbH & Co. KG")	N
same_service(extended_service_name="HTTPS" AND tls.certificates.leaf_data.issuer_dn=/C\\=DE, ST\\=[A-Z][a-zA-Z]{2,6}, [A-Z][A-Z]?\\=[A-Z][a-zA-Z\\S\\s0-9]+, O\\=W[a-zA-Z]{3,3} Kontakttechnik GmbH \& Co\\. KG, [A-Z][A-Z]?\\=[A-Z][A-Z\\S\\s0-9 a-z]+, emailAddress\\=info\\@wago\\.com/)	N
same_service(extended_service_name="HTTP" AND http.response.headers:(key:Server AND value.headers:"WAGO_Webs"))	N
same_service(extended_service_name="HTTP" AND http.response.html_title=/[ ]?WAGO Ethernet Web-[a-zA-Z]ased Management[ ]?/)	N
same_service(extended_service_name="FTP" AND transport_fingerprint.raw="16000,64,true,M,1452,false,false")	Y
same_service(extended_service_name="MODBUS" AND transport_fingerprint.raw="16000,64,true,M,1452,false,false")	Y
same_service(extended_service_name="HTTPS" AND http.response.headers:(key:Server AND value.headers:"WAGO_Webs"))	N
same_service(extended_service_name="CODESYS" AND transport_fingerprint.raw="16000,64,true,M,1452,false,false")	Y
same_service(extended_service_name="HTTP" AND banner="HTTP/1.1 307 Temporary Redirect\\r\\nLocation: /webvisu/webvisu.htm\\r\\nContent-Length: 0\\r\\nDate: <REDACTED>\\r\\nServer: lighttpd\\r\\n")	Y
same_service(extended_service_name="HTTP" AND banner=/HTTP\\/1\\.1 200 [A-Z\\S\\s]{0,4}D[a-z]{2}[a-z]{0,2}e[a-z\\S\\s]{2,2} [\\S\\s]{0,3}[A-Z][a-zA-Z\\S\\s]{6,9}[\\r\\n][\\r\\n]Server\\: WAGO_Webs[\\r\\n][\\r\\n][A-Z][a-z]{2}[a-z]{0,3}t-[A-Z]?[a-z]{2}[a-z]{0,4}e[a-z]?\\: text\\/html[\\r\\n][\\r\\n]Pragma\\: no-cache[\\r\\n][\\r\\n]Expires\\: 0[0-9 \\S\\s]*[\\r\\n][\\r\\n]Cache-Control\\: [a-z]{2}	N
same_service(extended_service_name="HTTP" AND http.response.body=/\\<HTML\\>[\\r\\n][\\r\\n]\\<HEAD\\>[\\r\\n][\\r\\n] \\<TITLE\\>CoDeSys WebVisualization\\<\\/TITLE\\>[\\r\\n][\\r\\n][\\t]\\<style type\\=\\"text\\/css\\"\\>[\\r\\n][\\r\\n][\\t]\\/********* basic tags ********\\/[\\r\\n][\\r\\n][\\t]body[\\r\\n][\\r\\n][\\t]\{[\\r\\n][\\r\\n] [\\t][\\t]margin\\: 0;[\\r	Y
same_service(extended_service_name="SNMP" AND snmp.oid_system.desc=/WAGO [0-9A-Z]{2,3}-[0-9 A-Z]*[A-Z]{3}[A-Z\\S\\s]*/)	N
same_service(extended_service_name="HTTPS" AND http.request.uri=/https\\:\\/\\/[0-9]{2}[0-9]?\\.[0-9]{2}[0-9]?\\.[0-9]{2}[0-9]?\\.[0-9]{2}[\\S\\s0-9]*\\/webvisu\\/webvisu\\.htm/)	Y
same_service(extended_service_name="HTTP" AND http.response.body=/\\<!DOCTYPE html\\>[\\r\\n][\\r\\n]\\<html\\>[\\r\\n]\\<head[\\S\\sa-z A-Z0-9]*\\>[\\r\\n]\\<title\\>WAGO Ethernet Web-based Management\\<\\/title\\>[\\r\\n]\\<meta http-equiv\\=\\"Content-Type\\" content\\=\\"text\\/html; charset\\=iso-8859-1\\"\\>[\\r\\n]\\<meta name\\=\\"author\\" content\\=\\"WAGO Kontakttechnik GmbH \& Co\\. K	N
same_service(extended_service_name="HTTP" AND http.response.body="<!DOCTYPE html>\\r\\n<html>\\r\\n<head>\\r\\n\\t<meta http-equiv=\\"content-type\\" content=\\"text/html;charset=UTF-8\\" />\\r\\n\\t<script language=\\"javascript\\" src=\\"webvisu.js\\"></script>\\r\\n\\t<script language=\\"javascript\\" src=\\"browsercontrol_ext0.js\\"></script>\\r\\n\\r\\n</head>\\r\\n<body onload=\\"new Webvisu	Y
same_service(extended_service_name="HTTP" AND http.response.body=/\\<!DOCTYPE html\\>[\\r\\n][\\r\\n]\\<html\\>[\\r\\n][\\r\\n]\\<head\\>[\\r\\n][\\r\\n][\\t]\\<meta http-equiv\\=\\"content-type\\" content\\=\\"text\\/html;charset\\=UTF-8\\" \\/\\>[\\r\\n][\\r\\n][\\t]\\<script language\\=\\"javascript\\" src\\=\\"webvisu\\.js\\"\\>\\<\\/script\\>[\\r\\n][\\r\\n][\\t]\\<script language\\=\\"javasc	Y
same_service(extended_service_name="HTTP" AND http.request.uri=/http\\:\\/\\/[0-9]{2}[0-9]?\\.[0-9]{2}[0-9]?\\.[0-9]{2}[0-9]?\\.[0-9\\S\\s]{0,3}[0-9]{2}[0-9\\S\\s]*\\/webvisu\\/webvisu\\.htm/)	Y
same_service(extended_service_name="HTTP" AND http.request.uri=/http\\:\\/\\/[0-9]{2}[0-9]?\\.11\\.[0-9]{3}\\.[0-9]{3}[\\S\\s0-9]{0,4}\\/webvisu\\/webvisu\\.htm/)	Y

the legitimate portal website HTML code, thus confirming their identity in an unintrusive manner. We consider the remaining 34 HTTP(s) hosts to be "potential false positives" since this test was inconclusive for them.

This leaves 94 (1.33%) total hosts in which we cannot say with absolute certainty belong to the target device (although many likely do since they run the correct operating system and speak the correct protocols). Regardless, our tests empirically confirm that the lower bound for *PLCHound*'s true positive rate is 98.67%.

D Generalizability of True Positive Rate

To support the claim that the low false positive rate calculated in Section 5.2 holds true across vendors, we apply various statistical significance tests using the industry-standard 95% confidence ($p = 0.05$). For this analysis, we make the conservative assumption that all datapoints are independent from each other, even though in practice, many hosts are discovered by similar, if not identical, queries and therefore share an underlying dependence. Since all 50 of our randomly sampled hosts from each vendor were manually confirmed to be true positives, it is highly likely that other hosts in those datasets can be indirectly confirmed as well, given they were found by similar queries. We intentionally do not account for this relationship in our calculations to determine a conservative lower bound extrapolation.

First, we apply the Wilson Score Interval, which is given by:

$$\text{Lower bound} = \frac{\hat{p} + \frac{z^2}{2n} - z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

$$\text{Upper bound} = \frac{\hat{p} + \frac{z^2}{2n} + z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}$$

where:

- $\hat{p}$ is the sample proportion of true positives (1 in this case),
- n is the sample size (50 in this case),
- z is the z-score corresponding to the desired confidence level (for a 95% confidence level, $z \approx 1.96$).

Plugging in the values with a 95% confidence level:

$$\text{Lower bound} = \frac{1 + \frac{1.96^2}{2\times 50} - 1.96\sqrt{\frac{1\times 0}{50} + \frac{1.96^2}{4\times 50^2}}}{1 + \frac{1.96^2}{50}} \approx 0.9286$$

$$\text{Upper bound} = \frac{1 + \frac{1.96^2}{2\times 50} + 1.96\sqrt{\frac{1\times 0}{50} + \frac{1.96^2}{4\times 50^2}}}{1 + \frac{1.96^2}{50}} \approx 1$$

Thus, the 95% Wilson score interval for the true positive rate is approximately [0.93, 1.00], indicating that with 95% confidence, the true positive rate is at least 93% and could be as high as 100%.

Additionally, we can perform a binomial test to determine the lowest possible true positive rate given our 95% confidence constraint.

Given:

- Number of true positives (x): 50

- Total number of samples (n): 50
- Significance level (α): 0.05

We want to find the minimal hypothesized true positive rate ($p_{\min}$) such that the p-value is less than α:

$$\text{Find } p_{\min} \text{ such that:} \quad \text{p-value} = 1 - \sum_{k=0}^{x-1} \binom{n}{k} p_{\min}^{k} (1-p_{\min})^{n-k} < \alpha$$

By numerical calculation, we find that:

$$p_{\min} \approx 0.9418$$

Thus, we can confidently deduce that the true positive rate is likely above 94%.

Since the both conservative statistical tests support the claim of a high (>93% and >94%) true positive rate and precisely zero false positives were identified in our samples, it is reasonable to conclude that our experimentally verified true positive rate of 98.67% for WAGO likely holds true across vendors.