Automatically Discovering Surveillance Devices in the Cyberspace

Qiang Li
School of Computer and Information Technology
Beijing Jiaotong University
liqiang@bjtu.edu.cn

Xuan Feng
Institute of Information Engineering, CAS
School of Cyber Security, University of Chinese Academy of Sciences
fengxuan@iie.ac.cn

Haining Wang
Department of Electrical and Computer Engineering
University of Delaware
hnw@udel.edu

Limin Sun *
Institute of Information Engineering, CAS
School of Cyber Security, University of Chinese Academy of Sciences
sunlimin@iie.ac.cn

ABSTRACT

Surveillance devices with IP addresses are accessible on the Internet and play a crucial role in monitoring physical worlds. Discovering surveillance devices is a prerequisite for ensuring high availability, reliability, and security of these devices. However, today's device search depends on keywords of packet head fields, and keyword collection is done manually, which requires enormous human efforts and induces inevitable human errors. The difficulty of keeping keywords complete and updated has severely impeded an accurate and large-scale device discovery. To address this problem, we propose to automatically generate device fingerprints based on webpages embedded in surveillance devices. We use natural language processing to extract the content of webpages and machine learning to build a classification model. We achieve real-time and non-intrusive web crawling by leveraging network scanning technology. We implement a prototype of our proposed discovery system and evaluate its effectiveness through real-world experiments. The experimental results show that those automatically generated fingerprints yield very high accuracy of 99% precision and 96% recall. We also deploy the prototype system on Amazon EC2 and search surveillance devices in the whole IPv4 space (nearly 4 billion). The number of devices we found is almost 1.6 million, about twice as many as those using commercial search engines.

*This author is the corresponding author.

MMSys'17, June 20-23, 2017, Taipei, Taiwan

DOI: http://dx.doi.org/10.1145/3083187.3084020

CCS Concepts

•**Networks** → **Network management;** *Network experimentation;* •**Information systems** → Data provenance;

Keywords

Surveillance device; Automatic device discovery; Network measurement

1. INTRODUCTION

A surveillance device is one typical Internet of Things (IoT) device, playing a vital role in bridging between the cyberspace and the physical world. Nowadays, for the sake of remote access and control, more surveillance devices are visible and accessible via IP addresses. However, this would also pose security and privacy concerns. For instance, people can directly know those surrounding scenes by watching the video streams while you are working or falling asleep at home [1]. Even worse, many surveillance devices, being exploited as parts of a "botnet," could attack critical national infrastructures. On October 21, 2016, the "botnet" that consists of a vast number of surveillance devices [2], such as IP-camera, Digital or Network Video Recorder (DVR/NVR), and Closed-Circuit TV (CCTV), attacked the servers of Dyn services, in which much of the Internet's domain name systems (DNS) are located, causing Internet service disruption across Europe and the United States. Such a security incident forces us to rethink the assumptions of how we use and maintain these surveillance devices.

From a security perspective, discovering online surveillance devices in the cyberspace is a prerequisite of preventing them from being compromised and exploited. It can help system administrators with security auditing, detecting new kinds of vulnerabilities and intrusion, and preserving device integrity on the Internet. Meanwhile, discovering surveillance devices can also shed light on availability, reliability, and the distribution of these devices. Device users or manufacturers would plan a wiser decision based on the online

activities of surveillance devices.

However, it is quite challenging to automatically and accurately discover surveillance devices in the cyberspace. Existing search engines, Shodan [3] and Censys [4], find these Internet-connected devices by using manually marked keywords. They send requests of application-layer protocols to online devices and recognize them by comparing the field values of reply messages with pre-defined keywords. Unfortunately, such a manual-based device identification is an arduous and error-prone process, and it is especially hard to achieve completeness. This is because it's hard to keep the discovery updated with the addition of numerous new devices and version upgrades.

In this paper, we aim to discover these online surveillance devices in an automatic and accurate manner. We take the first step in automatically generating fingerprints of surveillance devices and identifying them in the cyberspace. An intuitive insight behind our work is that every surveillance device typically uses a web-view interface for remote configuration and permission login. These webpages have two characteristics: *invariant* and *distinct*. webpages are written into the firmware of surveillance devices and remain intact for a long time. Thus, the webpage's appearance is capable of acting as a signature of a device. Moreover, there are a variety of web appearances for different surveillance devices. Those embedded webpages can be represented as the distinct features to recognize surveillance devices among Internet-connected devices.

The key feature of our approach is to automatically generate fingerprints based on webpages and use them to discover surveillance devices on the Internet. However, in practice, we have to address two challenges to enable this feature. (1) There are a large number of web appearances. Many other embedded devices also have webpages, such as routers, network bridges, printers, and industrial control devices. Additionally, most surveillance webpages have user login interfaces with password protection. (2) Traditional web crawler technology is time-consuming and does not meet our requirement. For a commonly seen URL of a website, there are many redundant pages within it. Commercial websites selling cameras would also affect the identification of surveillance devices. Moreover, many online surveillance devices do not have their domain names in their URLs, and they only have IP addresses.

To overcome these challenges, we propose to use natural language processing to extract the content of webpages. Visible and invisible information are extracted from the login or configuration pages of surveillance devices. We propose an iterative approach to find out common features of fingerprint generation for surveillance devices. Initially, we use a short vector related to the web content of surveillance devices. During each iterative process, statistical metrics are used to select new features to expand the initial vector. Machine learning algorithms are used for training a classification model based on those generated features. We use the classification models as the fingerprints of surveillance devices. Furthermore, we propose a real-time and non-intrusive web crawling scheme based on network scanning technology. We send a packet with stateless connection to an IP address to determine whether its host is alive. If the device is alive, an HTTP connection is established to obtain its page according to its HTTP status codes [5]. Note that our approach avoids crawling commercial websites with

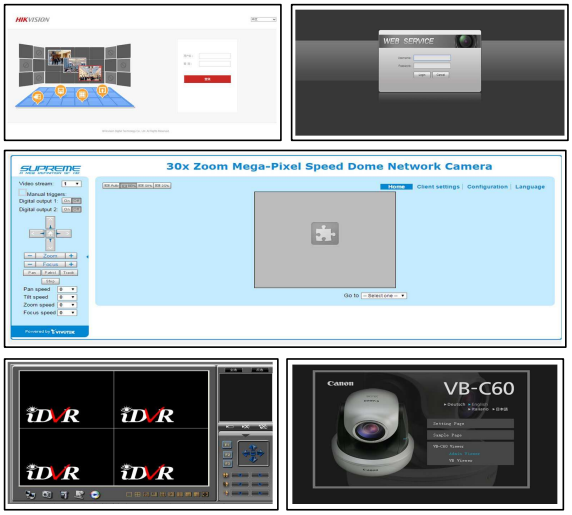

Figure 1: The various and distinct appearances of surveillance webpages.

Table 1: Services in surveillance devices.

Services	Function
RTSP [8]	Control & Transfer live streams
Onvif [9] / PSIA [10]	Interoperability Standards
HTTP [5]	User configuration

information about selling surveillance devices.

We implement a prototype of our proposed system using python and go [6] as a self-contained piece of software based on open-source libraries. We run our system in real-world experiments to evaluate its performance. The experiments show that our approach can achieve 99% precision and 96% recall in device classification. We also deploy the prototype system in Amazon EC2 [7] and perform device discovery four times from September 2015 to January 2016. Two of the four search attempts are the Internet-scale searches, in which we search the whole IPv4 space (nearly 4 billion) within 24 hours and identify 1.6 million surveillance devices. The number of surveillance devices we found is nearly twice as many as those using existing device search engines.

The major contributions of our work are summarized as follows.

- We have proposed an automatic fingerprint generation approach for surveillance devices. It is the first work to use web appearance to identify online devices in the cyberspace.
- We have implemented a prototype system and have verified it in real-world experiments. The experimental results show that our approach achieves very high accuracy in device identification.
- We have deployed the prototype system in Amazon EC2 and have performed surveillance device discovery four times over a five-month period. The number of devices we found is nearly 1.6 million, much more than those using commercial search engines.

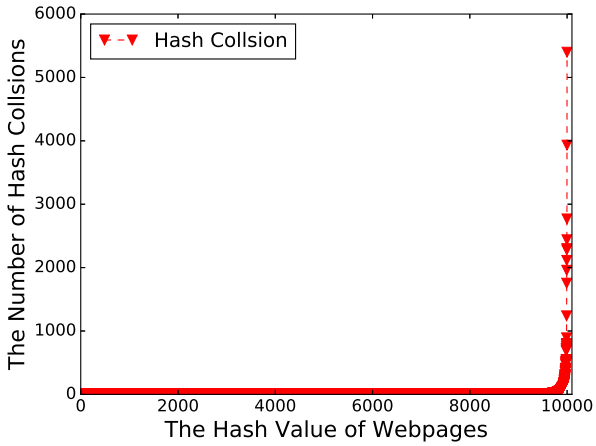

Figure 2: The diversity of webpages in the cyberspace.

The remainder of the paper is structured as follows. Section 2 highlights system design considerations. Section 3 presents the automatic fingerprint generation based on webpages. Section 4 describes our real-time web crawling. Sections 5 and 6 detail our system implementation and real-world experiments. Section 7 discusses future improvements. Section 8 surveys related work, and finally, Section 9 concludes the paper.

2. SYSTEM DESIGN CONSIDERATIONS

In this section, we first describe surveillance devices and their webpages, especially our design rationale behind the webpage-based fingerprint generation. We then introduce our design considerations of performing Internet-wide measurements for data collection and device discovery.

2.1 Surveillance Devices

A surveillance device is a type of digital video device typically deployed for monitoring the surrounding environment, which can send and receive data via a local computer network or the Internet. Typically, commercial surveillance devices run over four different application services, including PSIA [10] ONVIF [9], RTSP [8] and HTTP [5], as listed in Table 1. PSIA and ONVIF are standardized profiles for IP-based physical products, supporting interoperability of various devices and systems. However, due to compatibility issues, many surveillance devices do not support those two standards. RTSP is used to establish and control an application media session between a client and a server. It requires two phases: user authentication and data transmission. HTTP is used to set the configuration of surveillance devices.

Today, device manufacturers usually provide a user-friendly web interface to configure, access and manage a surveillance device conveniently. Users can change the configuration setting and access the streaming video from those webpages. As shown in Figure 1, there are five different web appearances from five different types of surveillance devices. The top webpages are used for login; the middle webpage is used for configuration, and the bottom webpages are used for device description. We observe that different devices typically have unique webpage properties (layout, structure, and content), which could serve as signatures of surveillance devices to discover them in the cyberspace.

In order to generate the fingerprint of a surveillance device, two fundamental properties must be held for those webpages. First, those webpages should be relatively stable over time; in other words, their appearances should rarely change. Second, there should be a significant variability in different webpages, so that it is reasonable to use a webpage as the signature of a surveillance device.

We first explore the stability of these devices' webpages. Manufacturers produce surveillance devices by re-using existing techniques, such as an embedded system on a chip (SOC) designs based on ARM or MIPS CPUs and network connectivity via Ethernet or WiFi. Most of them are controlled by vendor-specific and chipset-specific firmware [11]. The webpages of surveillance devices are also fixed in the firmware, and many manufacturers set those files to "read-only" [12]. The only way to change webpages is to update the firmware to a new version. Many devices often cannot be remotely updated and require the user's physical access to do updates. Even though a surveillance device needs to update its firmware, its webpages are likely to remain intact because firmware updates often just patch existing vulnerabilities.

Then, we investigate the diversity of webpages on the Internet. We have randomly crawled down 72,656 webpages from the whole IPv4 space. We extract their contents and use a hash algorithm to calculate their MD5 checksums for every webpage. There are 10,000 different hash values from those webpages. Figure 2 shows the diversity of those webpages in the cyberspace. We can see that nearly 8,686 pages have a distinct MD5 checksum value and no hash collision occurs among them. For those pages that have hash collisions, we further confirm that their hosts are indeed the same type of devices from the same manufacturer. There is a long-tail effect for the hash collisions of the webpages on the Internet, implying that there are a few very popular surveillance devices being widely deployed. The diversity of webpages is significant enough to distinguish devices from one another. Thus, we will use the characteristics of webpages as the signature for surveillance device discovery in the cyberspace.

As far as we know, this is the first work to automatically generate device fingerprints based on their web appearances. We will use natural language processing to extract features from the web and use machine learning techniques to generate their fingerprints.

2.2 Internet-wide Measurement

Our work will use the generated fingerprints to discover online surveillance devices on the Internet-wide scale. We have carefully designed our measurements to collect webpages across the Internet and reduce the side effects of our measurements upon the remote networks.

At first, we clarify the purpose of our measurement by adding reverse DNS entries for our measurement server [13] and running a simple webpage on port 80 that describes the goals of this research. The webpage describes what data we are collecting, and how to contact us to be excluded from our research by being added to our blacklist. Second, we do not apply traditional web scraping to collect webpages on the Internet. Instead, we use network scanning technology to collect webpages, which can reduce our

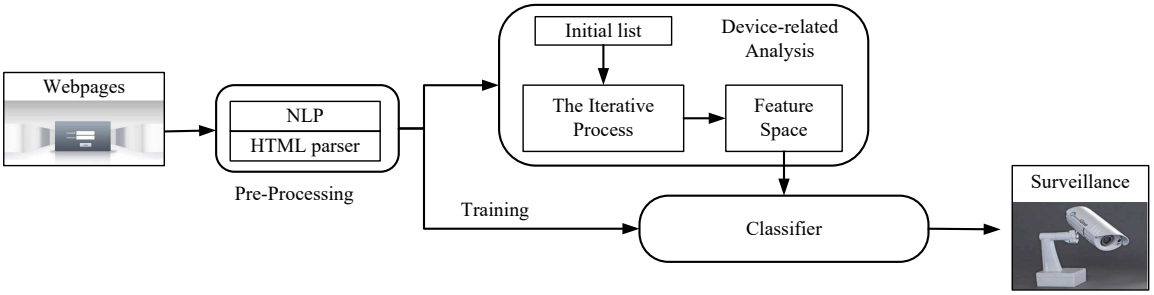

Figure 3: Automatically generating fingerprints of surveillance devices based on webpages.

measurement latency. Note that our web crawling approach performs standards-compliant handshakes without any malformed payloads involved. If a host runs an HTTP service, we will attempt to obtain its webpages; otherwise, we will just skip it. Third, our web scraping approach will not collect the information of selling surveillance devices from the commercial websites. This will improve the performance of our generated fingerprints.

Many devices are deployed in a local or enterprise system network, which cannot be accessed directly. In this work, we only focus on those public devices, rather than those inaccessible devices behind VPNs or firewalls, which are better protected.

3. FINGERPRINT GENERATION

In this section, we present the details of fingerprint generation and describe the key techniques applied in our identification approach.

3.1 Overall Architecture.

Figure 3 shows the overall architecture of the fingerprint generation based on webpages for surveillance devices. It consists of three components to generate fingerprints of surveillance devices: the pre-processing module, the device-related analysis module, and the classifier module. In the pre-processing module, we extract the content of webpages through an HTML (XML) documents parser and process them through natural language processing (NLP). The detailed steps include pulling data from web files, word splitting, stemming, and redundant content removal. In the device-related analysis module, it is an iterative process to find common features from webpages of surveillance devices. We first select a small set of original features associated with the surveillance devices. Then, the feature selection algorithm would iterate the related features automatically from the data set of the webpages. The extensive features serve as the feature space for every webpage of surveillance devices during the classification phase. In the classifier module, we use a supervised machine learning approach to train a classifier based on a set of training data. Each webpage is transformed into a feature vector in the pre-processing module, and the training set is represented as a training matrix, where columns are feature spaces and rows are the number of training data. Given any webpage, our approach can identify whether it is a surveillance device.

We propose to generate fingerprints of surveillance devices based on web appearance. In particular, our approach has two advantages. First, fingerprints generated by our method are not limited to user login webpages of devices in the cyberspace. Because many webpages have access control, we can only reach their user login pages. From a user's point view, the login interface has little information, as it merely displays the login information. However, much information is invisible and hidden behind the appearance of webpages, including comments, quotes, CSS, and scripts. Even for the login web interface, the hidden information is capable of identifying a surveillance device. We will use natural language processing to extract visible and invisible information for generating fingerprints of surveillance devices. Second, our approach can handle webpages of other embedded devices, such as routers, net-printers, and so on. The number of those devices is much larger than the number of surveillance devices. It is impossible and impractical for us to use the training webpages to cover all of the embedded devices. Thus, we propose an iterative approach that selects common features for the webpages of surveillance devices (Section 3.3).

Although our approach is used to generate fingerprints of surveillance devices, it could be easily expanded to identify other physical devices with minor changes.

3.2 Pre-Processing

Webpages are typically created by markup languages (HTML) along with embedded objects (scripts, images, videos, and so on). Browsers coordinate those various resource elements to render a webpage as the graphical user interface. On a lower level, we can use HTTP to make such requests for obtaining those webpages. Note that both visible and invisible information can be obtained via the HTTP protocol. Here we put them together and parse them in the following two forms:

- **Text.** There are various text contents in webpages, such as headings, paragraphs, lists, hyperlinks, quotes, and comments. They are delineated by tags, written using angle brackets, like ... , to provide document text.
- **Non-Text.** Non-textual information consists of images, audios, and videos, which are element attributes and used to present the webpage.

We use the HTML parser [14] to extract raw data from a webpage. For textual information, markup language symbols are directly removed because they have no meaningful information for classification. We extract all textual contents from the webpage. For non-text information, although computer version technologies can cope with them, it is very

Figure 4: An example of what the pre-processing module extracts from the content list of a webpage.

difficult to meet our requirements in real-time and accuracy. Thus, we transfer non-text content into text. For instance, we parse tags $<img\ directory/name>$ and record them as the composite text "image name". We extract them as the form of textual information from every webpage.

After parsing, we use a natural language processing package [15] to handle these textual contents. There are usually conjunctive words, delimiter-separated words, and lettercase separated words. For instance, words are divided by the symbol '/' in the hyperlinks. Regression expressions are used to split them into individual words. Considering that various languages and surveillance devices are deployed all over the world, we use a translation language package to convert into standard English.

We use stemming to transfer words to their original or root forms. For example, "Services" is replaced by the word "Service." Those words have a similar usage to differentiate cameras from other devices, and the stemming process reduces the amount of textual information. After stemming, we remove the redundant text content. Webpages written by markup languages have many words through which to organize the structure of webpages. Additionally, we also remove numbers, punctuations, and stopwords. Stopwords are some of the most common words, like "the" and "is." These words carry little meaningful information about the representative of the webpages. If we were to feed them directly to a classifier, it would shadow real interesting terms. Hence, we exclude them.

Figure 4 shows a case that the pre-processing module extracts the content list from a webpage. The left part of the figure is the raw data of the webpage with many HTML semantics and irregular words coded by developers. The right part is the content list after processing by the pre-processing module. After the HTML parser and natural language processing, each webpage becomes a concise item as a feature vector to process in the next stage.

3.3 Device-related Analysis

We need to extract common features used for identifying surveillance devices. There are many webpage appearances on the Internet. Surveillance devices from different manufacturers have different appearances. For instance, "Hikvision" device webpages are different from "Sony" webcam appearances. Additionally, other embedded devices also have webpages, such as routers, network bridges, printers, and industrial control devices. In this module, we aim to seek

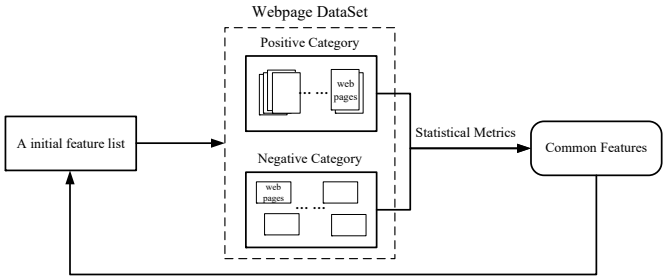

Figure 5: The iterative process for finding out common features for webpages of surveillance devices.

common features of surveillance devices that are different from other devices.

We propose an iterative approach to extract common features for generating fingerprints of surveillance devices. First, we manually choose a small set of features as the initial features and use them to find derivative features from the dataset. During each iterative process, we employ statistical measurement methods to calculate the correlation degree of each feature. We remove redundant and irrelevant features from the dataset during each iterative process, because they do not contribute to the accuracy of a predictive model. We use several statistical metrics for calculating the importance of each feature, such as the Chi-Square (Chi^2) test, information gain, and correlation coefficient scores. According to the scores calculated by statistical algorithms, we choose the top K features as common features. Those K features will be the initial features for the next iterative process.

Here we take the Chi^2 test [16] as the statistical metric to illustrate our iterative process. A chi-squared test is a statistical test that investigates whether the distributions of categorical variables differ from one another. In particular, we use the set of surveillance webpages as the expected distributions and the other webpages as another category. The Chi^2 test can identify those features that are the most related to the surveillance devices. Figure 5 shows our proposed selection based on the Chi^2 test. First, a device-related textual list is given to describe a surveillance device. It is impractical to emulate all possible contents related to surveillance devices by human efforts. We use a limited-length textual list, in which these texts are commonly seen in the webpage of surveillance devices. In this paper, our list is only six words (see Section 6), such as "camera" and "surveillance." We use a training data mixed with various webpages. Note that the training data is randomly chosen from the Internet. Then we divide the training data into positive and negative categories. The rule is simple:

- If the textual list appears within a webpage, the sample is labeled as positive; otherwise, it is negative.

Based on the training data, the Chi^2 test can distinguish the deviations between the positive and the negative. We manage to obtain the textual content that causes differences from what is expected. If there is no significant difference between categories, the Chi^2 test will stop and rank the device-related words in descending order. Based on the Chi^2 test, we can get a much larger list than the initial device-related textual list. We use the list as the feature space to represent the webpage to process in the next stage. Note

that, except for manual labor over the initial textual list (its length is very short), the feature selection process is completely automatic.

Finally, generated common features are used as the feature space for webpages of surveillance devices. The feature space refers to the n-dimensions, which is crucial in pattern recognition and machine learning.

3.4 Classifier

Based on the training data and feature space, we build a classification model for determining whether a webpage is a surveillance device. Each webpage is transferred into a textual list in the pre-processing module and is represented as a feature vector in the device-related analysis module. The training data is a set of mixed webpages and is represented as a matrix. Each row is a feature vector as per page instance, and each column is a value of the fields in the feature space. We have investigated various machine learning algorithms [17] for building the model, including logistic regression, linear discriminant analysis, support vector machine (SVM), decision trees, boosting, and neural networks. We utilize the standard support vector machine as our classifier. SVM is widely used for classification and regression analysis. Given a set of training examples with two different categories, the algorithm tries to find a hyperplane separating the examples. As a result, it determines the side of the hyperplane to which new test examples belong. In particular, for an unknown webpage from online devices, the classifier can determine whether it is a surveillance device.

4. REAL-TIME WEB CRAWLING

Web crawling is a technique of extracting information from websites and has been widely used by most search engines. To identify a surveillance device, we need to obtain webpages on the Internet and determine whether they belong to surveillance devices based on the generated fingerprints as proposed above. Directly using traditional web crawler is infeasible for discovering surveillance devices, due to the following three reasons. First, there are many redundant webpages on the Internet, and they are useless in discovering surveillance devices. Furthermore, commercial websites with information selling surveillance devices would affect the fingerprint generation of surveillance devices' webpages. Second, many online devices have not yet registered domain names and only have their IP addresses for forming their URLs.

To meet these issues, we propose to obtain webpages through the network scanning technology. Network scanning is a procedure for identifying active hosts on a network. Scanning procedures return the information about which IP addresses map to live hosts on the Internet and what services they offer. Then we use the Hypertext Transfer Protocol (HTTP) to send a "*GET /* " request to live host IP addresses. The responses include the root webpages of remote host. Figure 6 shows the online web crawling based on scanning technology.

Horizontal Scanner. There are 4 billion IPv4 addresses in the cyberspace, and we send one packet to every IP address to determine whether there is a live host. Recent research ZMap [18] suggests that one-packet probing can cover nearly 96% of the detection space and can speed up the measurement. We adopt this approach by sending out only one packet to an IP address each time to discover live hosts. For each IP address, unlike TCP, we do not maintain

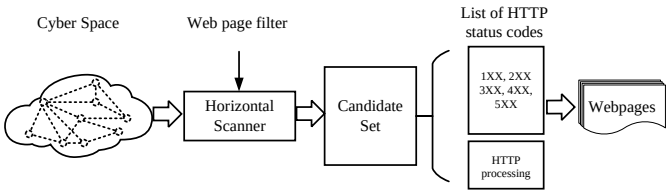

Figure 6: Real-time webpage crawling.

```
<META HTTP-EQUIV="refresh" CONTENT="0;
      URL=/cgi-sys/defaultwebpage.cgi">
<script>
self.location = "webclient.html";
</script>
```

Figure 7: A redirect case in web crawling.

a state connection. Instead, we use stateless connections to send probing packets without waiting to synchronize the connection times, which can significantly speed up the measurement. Furthermore, our detection range order is randomized. If we probed every address in numerical order (e.g., 10.0.0.1, 10.0.0.2, 10.0.0.3.), it would cause a burst of consecutive packets, thus disturbing the remote network. They are used in many previous works [19]. To avoid this problem, we use an alternative approach to IP-sequential, a random permutation of the address space [20]. We target uniformly and randomly from the full range and intermingle probes to many subnets, which reduces the instantaneous load on individual networks and produces an unbiased random sampling.

Web Crawling. After horizontal scanning in the cyberspace, we can obtain the candidate set. Compared with the initial 4 billion IPv4 addresses, the number of live hosts is just tens of millions. Then we build standards-compliant TCP handshakes with each live host. An application-layer HTTP *GET /* request is sent to each live host. Their responses include the data of root webpages. However, there are several cases in which we cannot directly use the response packets to identify surveillance devices.

We use the process to handle HTTP status codes according to the list [5], as shown in Figure 6. If we are successful in obtaining the response data, we can directly store them as files. Otherwise, we would further process them based on the following three rules:

- If the request is failed, we remove it from the live candidate. The "4xx " class of status code indicates that we do not receive the correct data. For instance, "400" means a bad request; "401" means an unauthorized request, "403" means that the server refuses to answer, and so on. For ethical considerations, we never attempt to guess login credentials for any online webpage. We only obtain their login interface webpages if they are protected by password authorization.

- If the server fails to fulfill an apparently valid request, we would try to re-send the request after a period. The "5xx" indicates that the server is encountering an error or is otherwise incapable of performing the request. We will divide the response code to determine if the

live host is in a temporary or permanent condition. For instance, "500 " means Internal Server Error, and "503" means Service Unavailable.

- If the redirect appears, we would try to re-send the "*GET*" request to the new address. Redirection is commonly seen when we obtain webpages via network scanning technology. When the status code is "302 redirect," we should make the HTTP connection again and get the redirect webpage. Furthermore, even the status code is "200 Ok," we also finish the redirection in several cases. The redirect process works as follows: the regular expression extracts the hyperlink position and combines the IP address and category of the webpages, and then we send a new request with the IP/new-index. For instance, the status code is "200 Ok" in Figure 7, and we send the HTTP "GET" request with the "IP/webclient.html" to the live host.

Exploiting network scanning, we can collect webpages on the Internet in a real-time and non intrusive manner. Our web crawling approach has three advantages. First, we can avoid data collection from the commercial websites related to camera information. In fact, many commercial websites with information selling devices, use the hierarchical directory structure to represent their URL addresses. For instance, the manufacturer "SONY" sells its devices on a URL ("https://pro.sony.com/bbsc/ssr/cat-securitycameras/"), in which the digital camera information is hidden in its subfolder ("bbsc/ssr/catsecuritycameras/"). Our web crawling approach does not access those webpages. Second, the horizontal scanner filters out a large number of unqualified hosts, and our Internet-wide measurement latency is reduced. Third, we attempt to limit the side effects of our measurement upon remote networks as much as possible. For every candidate, there are only three TCP handshakes and one HTTP request on average.

5. IMPLEMENTATION

We have implemented a prototype of our proposed approach as a self-contained piece of software based on open-source libraries. The key components of our approach, web crawling, and device classification are mainly written in Python and Go language [6].

Figure 8 shows the component diagram of our prototype. We send a single TCP packet that has "SYN" filed in its header and a "NULL" data payload to determine whether a host is alive. By reusing IP address randomization algorithms [21], we send this probing packet to every IP address with a certain communication port, such as "80" and "8080," where web services are offered. For each IP address, no state is maintained like the network scanner tool, ZMap [18]. Our probing TCP packets are stateless, and we utilize network bandwidth as much as possible to accelerate the horizontal scanning speed. The response packets are stored in buffer queues. If a host responds with "SYN-ACK" in the TCP header or responds to the pre-generated packet, we put this host into the candidate set; otherwise, we discard it. The candidates are stored as a JSON file to process in the next phase.

The HTTP probe sends the request encapsulated "GET" to the candidate date set in buffer queues or the JSON file. For each candidate, we establish TCP connections in parallel and then communicate at the application layer to obtain the webpages. Based on response status codes, we process the candidate webpages. In the redirect case, the sender would re-send the request with a new address. In the re-connect case, the sender leaves it in the last position of the request list. We store the HTTP response data on the disk.

We use the open-source Python library "BeautifulSoup" [14] to remove the markup languages syntaxes. It is a toolkit to retrieve data out of HTML and XML files, dissect a document, and extract what we need. We use the Natural Language Toolkit (NLTK) [15] to obtain the content list from every webpage. Each webpage is transformed into a feature vector for classification. We use the open-source skit-learn with SVM classifier [22] to train a classification model (see Section 6.2). The classifier determines whether the webpage belongs to a surveillance device. If an online surveillance device is identified, we store it in the database.

Table 2: The collected dataset of webpages on the Internet.

Webpage description	The amount
Login webpages	35,558
Configuration interface webpages	6,761
Commercial website	0

Table 3: The initial seed list for feature selection.

the initial list					
camera	dvr	surveillance	webcam	web-service	zoom

6. REAL-WORLD EXPERIMENTS

In this section, we conduct real-world experiments to evaluate the performance of our prototype system. Also, we deploy the prototype system on Amazon EC2 to discover online surveillance devices across the entire IPv4 space.

6.1 Data Collection

To collect datasets of the ground truth, we randomly select IP addresses with the permutation algorithm [21] from the whole IPv4 space. We use the horizontal scanner and HTTP "GET /" requests to individual IP addresses. In total, we have found 42,319 webpages, as shown in Table 2. There are 35,558 webpages with login interfaces, 6,761 webpages with configuration interfaces, and zero commercial websites. This is because our web crawling method is based on network scanning technology, rather than a traditional web crawler. Many redundant webpages are excluded. Additionally, we exclude the webpages from non-embedded devices, including those of popular HTTP servers, such as Apache and IIS, as well as common Telnet authentication prompts. Then we search the messages of the webpages in Google. If search results are related to surveillance devices, we mark them with a "Surveillance" tag, otherwise a "Non-surveillance" tag. This manual effort of creating the ground-truth dataset costs us nearly one month, and we find 8,202 webpages of surveillance devices in total. Approximately 95% of these webpages of surveillance devices are user login interfaces with password protection, and less than 5% of the webpages contain detailed information, like the configuration interface of Webcams. We divide the ground-truth dataset into two parts: a 20,000 size part for training (3,847 surveillance device web-

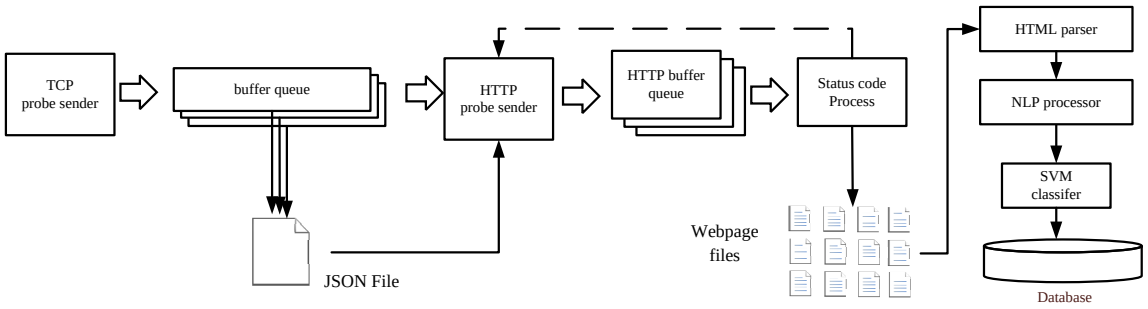

Figure 8: The component diagram of our prototype system.

pages) and a 22,319 size part for testing (4,355 surveillance device webpages).

6.2 Classification Performance

As mentioned before, we choose a small set of features as the initial list to find device-related features for surveillance devices. Table 3 presents the initial seed list. They are commonly seen in the webpage content of surveillance devices, such as "camera," "surveillance," and "DVR" (digital video recorder). We use precision and recall to present the classification performance.

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN},$$

where TP is the true positive number, FP is the false positive number, and FN is the false negative number. Figure 9 shows the classification performance along with the parameters of the feature selection process. The X-axis is the precision of identifying a surveillance device, and the Y-axis is the recall. In our experiments, the number of iterative processes would be empirically set to 3. The common features remain stable with the increasing number of iterations when $\tau = 3$. For each iterative process, there are two parameters: the feature selection algorithm and top N number. In our experiments, we choose four statistical feature selection algorithms (Chi^2 test, information gain, correlation coefficient scores, and TF-IDF [23]). We choose the parameter N in the range of [10,200]. As shown in Figure 9, the black circles contain those points that have high recall and precision under one parameter setting in the feature selection process, indicating that when the feature selection algorithm is the Chi^2 test and the top number N is set to 100, the classification model achieves the most promising results. Thus, we adopt this particular parameter setting in the following stages.

Second, we evaluate the impact of the training size upon the classification performance. We use the F1 score to represent the classifier performance. The F1 score can be interpreted as a weighted average of precision and recall:

$$F1\ score = 2 \cdot \frac{precision \cdot recall}{precision + recall}.$$

It reaches its best value at 1 and its worst score at 0. Figure 10 shows the classification performance along with the number of the training set. When the training data size is small (fewer than 500 webpages), the F1 score is only 80%. This is because the bias of the training samples causes the performance degradation. However, as the training

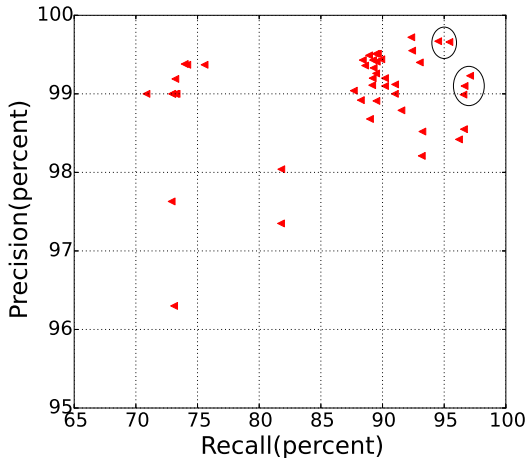

Figure 9: Precision-recall graph of different parameter settings.

set size increases, the classification performance significantly improves. When the training size is more than 5,000 webpages, the F1 score is close to 96.5%, and our classification can achieve a promising result.

Table 4: The overall accuracy for four classification models.

Classifier	TPs	FPs	GT	PR	RC	F1
SVM	4,147	42	4,355	99.00	95.22	96.98
KNN	4,209	40	4,355	99.06	96.65	97.84
Naive Bayes	3,865	374	4,355	91.18	88.75	89.95
Decision Tree	3,830	44	4,355	98.86	87.94	93.09

Furthermore, we evaluate the impact of the classification choice upon the performance. We choose four standard classification approaches: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Multinomial Naive Bayes, and Decision Tree. They are typical, supervised learning algorithms in pattern recognition [17]. We also use the Chi^2 as the feature selection and select the top 100 as features. The training data size is 20,000, and the test data size is 22,319; they do not have any overlap. Table 4 shows their performance among these four classification algorithms. The col-

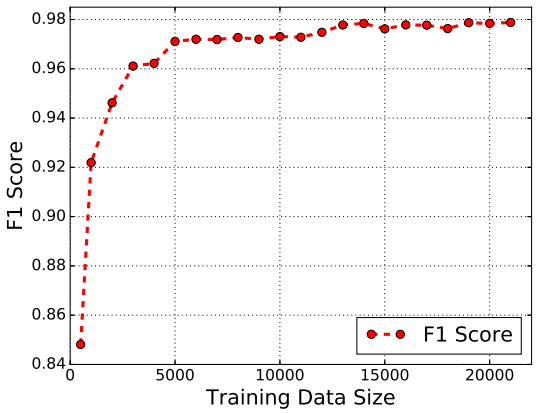

Figure 10: Training Data Size and F1 Score.

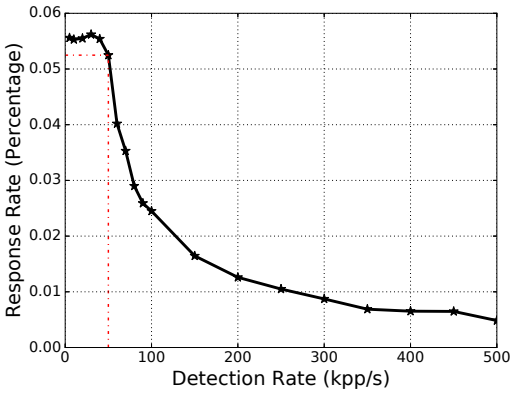

Figure 11: The hit rate and detection rate for webpage crawling.

umn "TPs" is the number of true positives; "FPs" is the number of false positives; "GT" is the number of ground truths; "PR" is precision and "RC" is recall. The results show that SVM and KNN have the best performance, and Multinomial Naive Bayes performs the worst. Note that there are no substantial differences among four classification performances. Here, we choose SVM because it is capable of generating a maximum margin classifier with robustness. We use SVM in our prototype system and run it to determine whether a webpage belongs to a surveillance device on the Internet.

6.3 Web Crawling Performance

Here we present the performance of crawling webpages on the Internet by exploiting network scanning technology. First, we evaluate the impact of the network bandwidth upon our proposed real-time webpage crawling method. As we mentioned in Section 4, we use stateless TCP to discover live hosts from a range of IP addresses. The higher the network bandwidth, the less time we spend on detection. However, overwhelming detection probing packets would cause network congestions and packet losses. We conduct the experiments under the different hit rates and detection rates. As Figure 11 shows, there is a tradeoff between the hit rate and detection rate. The hit rate is equal to $N_{candidate}/N_{total}$, where N_{total} is the total number of IP addresses and $N_{candidate}$ is the number of responding hosts. The detection rate is the speed of discovering physical devices. When the detection rate is 50,000 packets per second, we can achieve a stable hit rate for crawling webpages. A practical issue is that network congestion reduces the hit rate. We suggest adopting the most stable detection rate while keeping the hit rate high.

Second, we evaluate the performance of webpage crawling via HTTP "GET" requests. We use the dataset that we have collected all webpages (Section 6.5 describes our measurement) from ports 80 and 8080, which are most popular ports offering web services. Figure 12 shows the percentages of HTTP response status codes in the webpage crawling stage. There are 51 million (51,516,332) hosts giving their HTTP response code. There are 186 different kinds of status code. We only list the top 10 HTTP response status codes, but they occupy nearly 99.46% of all responses. There is a typical long-tail distribution of the HTTP

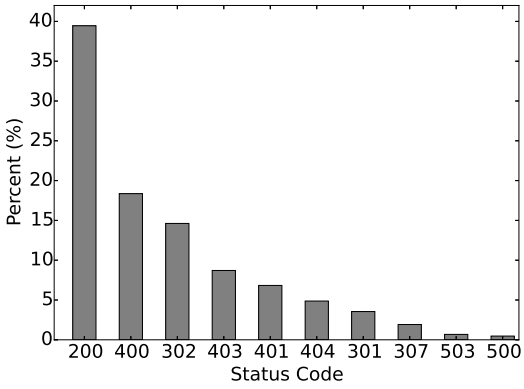

Figure 12: The percentage of HTTP response status codes in the webpage crawling stage.

response code throughout the world. The status code "200" ranks the highest, whose pages occupy 39.45% of the total pages, followed by the status codes "400" and "302." There is a further process handling redirect status code "302." For the other status codes like "4xx" and "5xx," they contribute little to our classification, and we just exclude those webpages.

Furthermore, we measure the time latency of different stages. The detection time of our approach is determined by three parts: (1) horizontal scanner, (2) HTTP "GET" request, and (3) classification, denoted as $T = T1 + T2 + T3$. We randomly choose 65,536 IP addresses to test their time latency. The experimental environment is that we send packets at the speed of 0.5 MB/s (about 500 packets per second). As shown in Figure 13, the horizontal scanner ($T1$) stage costs 135 seconds, and the HTTP "GET" request ($T2$) stage costs 10.18 seconds, and the last stage costs 7.88 seconds. The most time-consuming part is the horizontal scanner. The reason is that the target number of the horizontal scanner is 65,536 IP addresses, which is much more than the other two stages. The classification stage costs the least amount of time. Note that the time cost of the classification stage is mainly due to the pre-processing. Overall, the detection latency of our approach is acceptable.

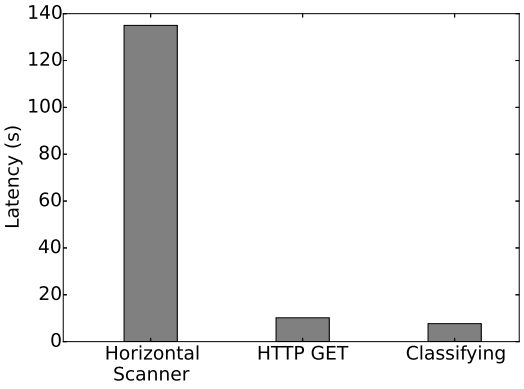

Figure 13: Time latency of online surveillance device discovery.

Table 5: CPU, memory, and bandwidth usage of prototype software.

	CPU usage	Memory	Bandwidth
Training process	10%	232.4MB	-
Online discovery	53%	208.9MB	50Mbps

6.4 System Overhead

We have measured the CPU, memory, and bandwidth usage of the prototype system. Table 5 presents the CPU, memory, and network bandwidth overheads of the training stage and the online discovery stage, respectively. The training process is running on a commercial laptop (Windows 10, 4vCPU, 8GB of memory). It costs only 10% of the CPU usage and 232MB of the memory usage. The online discovery process is running on a server inside Amazon EC2 [7], running Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-48-generic x86_64) with 2 vCPU, 8GB of memory and 450Mbps of bandwidth. The discovery stage consists of online webpage crawling, pre-processing, and classification. The average CPU usage is 53%, the average memory usage is 208Mbps, and the network bandwidth usage (out) is 50Mbps, about 10% of Amazon server network bandwidth. It is affordable in practice, and we can use the prototype system to detect surveillance devices on the Internet.

6.5 Surveillance Measurement Campaigns

We have deployed the prototype system to identify surveillance devices on a cloud server inside Amazon EC2 [7]. Table 6 presents the surveillance device discovery at the Internet scale. We have exhaustively searched the entire IPv4 address space (close to 3.7 billion addresses). Every time, we have excluded both reserved/unallocated IP space from IANA [24] and IP addresses that send emails to complain about our discovery activity. Altogether, we have added about 610 million IP addresses to our blacklist to exclude them from the Internet-wide search. For each data collection, we send 50,000 packets per second, and each experiment lasts for about 20 hours.

In the (3.7 billion IP addresses) experiment, we retrieve some raw data, including 51,516,332 webpages. Because surveillance devices are embedded with limited processing capabilities, they cannot run the "Apache" or "Microsoft IIS" web servers. Therefore, we exclude these webpages and some with bad status codes. After these filtering processes, we retrieve the webpages that likely belong to surveillance devices, and the number is more than 8 million (8,785,185). Our system extracts the content of webpages and determines whether a webpage belongs to a surveillance device or not. Table 7 shows the comparison between our approach and Shodan [3]/Censys [4]. We use our initial words to search surveillance devices in these two search engines and then retrieve the results. Note that the numbers of surveillance devices found in Shodan and Censys have a strong bias, because they include all of the pages that have matched keywords but are not surveillance into their search results. It is evident that our approach can discover more than 1.6 million (1,602,142) surveillance devices, nearly twice as many as existing search results.

Table 6: Surveillance device discovery at the Internet scale.

Begin Time	IP Space	Protocols	Ports
2015-09-25	3.7 billion	HTTP	80, 8080
2015-12-22	0.36 billion	HTTP	80, 8080
2016-01-07	0.36 billion	HTTP	80, 8080
2016-01-05	3.7 billion	HTTP	80, 8080

Table 7: Comparison with Shodan and Censys.

	The number of surveillance devices
Shodan [3]	155,999
Censys [4]	780,953
Our system	1,602,142

7. DISCUSSION AND FUTURE WORK

Our focus is on the discovery of visible surveillance devices across the Internet. Two aspects can be improved in our future work.

Invisible surveillance devices. For security concerns, all devices should have been access-controlled by network/application firewalls or network address translators (NATs). There are more than 245 million video surveillance cameras installed around the world since 2014 [25]. Although less than 1% of devices are exposed to the Internet, the total number is still very large, with nearly 1.6 million devices with IP addresses available for remote access. In this work, we only focus on those public devices, rather than the devices behind a VPN or firewall. We believe that invisible devices are in better security circumstances than visible ones.

As mentioned before, our approach cannot find invisible devices. They are located behind firewalls or NATs, which block our detection packets. Even if some devices are not behind firewalls or NATs, system administers might manually block our probing packets. In order to obtain an approximate "bird's-eye view" of surveillance devices (visible and invisible) in the cyberspace, we will continue the follow-up work in the future. First, we will deploy a monitoring sensor on multiple campus networks and use passive probing to listen to the outgoing/incoming data traffic at the network entrance. Then, we apply statistical methods to estimate the number of invisible devices.

General framework. Our work is a binary classification about whether a webpage is related to a surveillance device or not. Besides, different manufacturers would like to use different webpages to manage their devices. Intuitively, the webpage of a surveillance device can identify the manufacturer brand and the product version. In our future work, we will calculate the degree of similarity between the webpages of surveillance devices and use network graphs to distinguish the different types of surveillance devices.

Many other embedded devices also have webpages, such as routers, network bridges, printers, and industrial control devices. Our approach can be extended to discover other types of online devices, with minor modifications. In our future work, we will propose a general framework for automatically generating device fingerprints based on the webpages embedded in those devices.

8. RELATED WORK

Fingerprint generation. Fingerprinting is a technique for identifying operating systems (OS), applications, or network services. More than two decades ago, Comer et.al. [26] proposed to use differences between TCP implementations to generate operating system fingerprints. Since then, several well-known OS fingerprinting tools have been built, including Nmap [27], Xprobe2 [28], p0f [29], and RING [30]. These tools send TCP packets to remote IP addresses and then use the fingerprints stored in the database to identify whether a live host runs on Windows or Linux. These tools are open source, and many developers have contributed to their development as well as manually labeled OS fingerprints. Different from these works, our approach automatically generates the fingerprints of surveillance devices. Other fingerprinting research proposed to use timestamps to identify physical devices. Kohno et.al. [31] suggested using clock skew deviations to fingerprint devices. Its design principle is that different device implementations have a slightly different deviation according to Network Time Protocol. However, it faces network topology changes, time variances, and network noise in practice.

Network scanning. There have been many research studies on Internet-wide scanning over the past few years. The network scanning research [18, 32–35] mainly focuses on increasing the scanning speed over the Internet, from 4 months down to 30 days, one day, and more recently, 45 minutes. Xie et al. [32] proposed the UDmap algorithm to identify and analyze hosts across the entire IP address space. Heidemann et al. [33] explored the visible Internet to characterize the edge hosts and evaluate their usages via an active scanner within 30 days. Hong et al. [35] searched the entire Internet and then classified IP addresses as popular or unpopular. Leonard et al. [34] implemented IRLscanner as an Internet-wide detection mechanism within 24 hours, which focuses on typical application layer protocols, such as HTTP and SMTP. Durumeric et al. [18] proposed ZMap for Internet-wide host detection within 45 minutes. This method is a theoretical upper bound and cannot be easily achieved in practice due to network congestion. These network scanning works demonstrate the feasibility of discovering live hosts through Internet-wide scanning, called horizontal scanning, which is fundamental for finding surveillance devices in our work. Compared to these previous studies, our goal is to identify surveillance devices on the Internet. We extend the network scanning techniques to crawling webpages and use them as the signature of surveillance devices.

Internet-connected device searching. In the new era of the development of Internet of Things, massive physical devices are becoming pervasive and "invade" our daily life. Searching online physical devices has garnered great interest in the industry and academia. Shodan [3] is the world's first search engine for discovering Internet-connected devices. Censys [4] is an open-source search engine that identifies the devices and networks that compose the Internet. However, they both use manually marked keywords to discover physical devices. They send requests in the application-layer protocols to devices and recognize them by comparing the field values of replies with pre-defined keywords. Xuan et al. [36] detected ICS (Industrial Control System) devices on the Internet by analyzing 17 industrial control protocols. However, their device identification still depends on manual analysis. As a manual process, such an approach is arduous and incomplete, making it difficult to keep updated with the addition of numerous new devices and versions. By contrast, our work is the first to use webpages as device signatures. We propose an automatic fingerprint generation method, which can accurately discover surveillance devices on the Internet.

9. CONCLUSION

Online surveillance devices play a crucial role in monitoring the physical world. In this paper, we proposed a novel approach for automatic and accurate surveillance device searches. The core of our approach is to use a webpage embedded in a surveillance device as its fingerprint. We used natural language processing to extract the content of a webpage and generate a device fingerprint automatically. Based on device fingerprints, we utilized machine learning for device classification. Moreover, we proposed a new web crawling scheme to obtain webpages of surveillance devices in a real-time and nonintrusive manner. We implemented a prototype of our approach and evaluated its performance through real-world experiments. The experimental results show that our automatically generated fingerprints can achieve 96% recall and 99% precision in surveillance device classification. We further deployed our prototype on a cloud server in Amazon EC2 and conducted a device search over the entire IPv4 address space. The number of identified surveillance devices by our search is twice as many as those using existing device search engines.

10. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful and detailed feedback. This work was supported by the National Natural Science Foundation of China (Grant 61602029), National Key Research and Development Program (Grant 2016YFB0801303-1), the Major R&D Plan of Beijing Municipal Science & Technology Commission (Grant Z161100002616032), and National Natural Science Foundation of China (Grant U1536107).

11. REFERENCES

[1] Google Unsecured IP Cameras. [Online]. Available: http://www.inurl-view-index-shtml.net/google-unsecured-ip-cameras

[2] Major cyber attack disrupts Internet service across Europe and US. [Online]. Available: https://www.theguardian.com/technology/2016/oct/21/ddos-attack-dyn-internet-denial-service

[3] Shodan. The search engine for Internet-connected devices. [Online]. Available: https://www.shodan.io/

[4] Censys. A search engine for devices and networks based on Internet-wide scanning. [Online]. Available: https://censys.io/

[5] RFC 2616. List of http status codes. [Online]. Available: https://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

[6] The Go Programming Language. Open Source Project. [Online]. Available: https://golang.org/

[7] Amazon EC2. Amazon elastic compute cloud. [Online]. Available: https://aws.amazon.com/ec2/

[8] RFC 2326: Real Time Streaming Protocol. [Online]. Available: http://www.ietf.org/rfc/rfc2326.txt

[9] ONVIF. Open Network Video Interface Forum website. [Online]. Available: http://www.onvif.org

[10] PSIA. Physical Security Interoperability Alliance website. [Online]. Available: http://www.psialliance.org

[11] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A Large-Scale Analysis of the Security of Embedded Firmwares," in *23rd USENIX Security Symposium (USENIX Security 14)*, San Diego, CA, Aug. 2014, pp. 95–110.

[12] D. D. Chen, M. Egele, M. Woo, and D. Brumley, "Towards automated dynamic analysis for linux-based embedded firmware," in *The Network and Distributed System Security Symposium (NDSS)*, 2016.

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

[14] BeautifulSoup. A python library designed for quick turnaround projects like screen-scraping. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/

[15] NLTK. A leading platform for building Python programs for Natural Language Toolkit. [Online]. Available: http://www.nltk.org/

[16] The Chi Square Statistic. [Online]. Available: http://math.hws.edu/javamath/ryan/ChiSquare.html

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

[18] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications." in *Proceedings of the 22nd USENIX Security Symposium*, 2013.

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

[20] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," in *Proceedings of the 11th USENIX Security Symposium*, 2002, pp. 149–167.

[21] Numpy: Randomly Permute A Sequence. [Online]. Available: http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.permutation.html

[22] Scikit-Learn Machine Learning in Python. [Online]. Available: http://scikit-learn.org/stable/index.html

[23] G. Salton, A. Wong, and C.-S. Yang, "A vector space model for automatic indexing," *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[24] Assigned Numbers Authority (IANA). [Online]. Available: http://www.iana.org/

[25] Information Handling Services Markit Ltd. 245 million video surveillance cameras installed globally in 2014. [Online]. Available: https://technology.ihs.com/532501/245-million-video-surveillance-cameras-installed-globally-in-2014

[26] D. E. Comer and J. C. Lin, "Probing TCP Implementations," in *Proceedings of the USENIX Summer 1994 Technical Conference - Volume 1*, ser. USTC'94, 1994, pp. 17–17.

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

[28] Xprobe official home, 2004. [Online]. Available: http://www.sys-security.com/index.php?page=xprobe/

[29] Project details for P0F, 2004. [Online]. Available: http://freshmeat.net/projects/p0f/

[30] F. Veysset, O. Courtay, O. Heen, I. Team *et al.*, "New tool and technique for remote operating system fingerprinting," *Intranode Software Technologies*, vol. 4, 2002.

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

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

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

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

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

[36] X. Feng, Q. Li, H. Wang, and L. Sun, "Characterizing industrial control system devices on the Internet," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, Nov 2016, pp. 1–10.