WNV-Detector

automated and scalable detection of wireless network vulnerabilities

Huang, Yanxi; Zhu, Fangzhou; Liu, Liang; Meng, Wezhi; Hu, Simin; Ye, Renjun; Lv, Ting

Published in:
EURASIP Journal on Wireless Communications and Networking

Link to article, DOI:
10.1186/s13638-021-01978-4

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

Huang *et al. J Wireless Com Network* (2021) 2021:85
https://doi.org/10.1186/s13638-021-01978-4

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

WNV-Detector: automated and scalable detection of wireless network vulnerabilities

Yanxi Huang[1], Fangzhou Zhu[1*], Liang Liu[1], Wezhi Meng[2], Simin Hu[1], Renjun Ye[1] and Ting Lv[1]

*Correspondence: fangzhouzhu@nuaa.edu.cn
[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
Full list of author information is available at the end of the article

Abstract

The security of wireless routers receives much attention given by the increasing security threats. In the era of Internet of Things, many devices pose security vulnerabilities, and there is a significant need to analyze the current security status of devices. In this paper, we develop WNV-Detector, a universal and scalable framework for detecting wireless network vulnerabilities. Based on semantic analysis and named entities recognition, we design rules for automatic device identification of wireless access points and routers. The rules are automatically generated based on the information extracted from the admin webpages, and can be updated with a semi-automated method. To detect the security status of devices, WNV-Detector aims to extract the critical identity information and retrieve known vulnerabilities. In the evaluation, we collect information through web crawlers and build a comprehensive vulnerability database. We also build a prototype system based on WNV-Detector and evaluate it with routers from various vendors on the market. Our results indicate that the effectiveness of our WNV-Detector, i.e., the success rate of vulnerability detection could achieve 95.5%.

Keywords: Wireless network, Router, Access point, Vulnerability detection, Device identification

1 Introduction

In recent years, *Internet of Things (IoT)* technology develops in a rapid manner. The traditional industrial IoT is widely used in energy, manufacturing, transportation, etc. A large number of sensors are connected to each other to effectively manage and maintain information, and improve the efficiency of industrial production [1–3]. As IoT gradually enters every household, a variety of smart home products bring convenience to our lives, but also cause more security problems. In the year of 2016, the Mirai botnet infected a large number of embedded and IoT devices, launched several large-scale distributed denial of service attacks, and overwhelmed several high-profile targets [4]. In 2019, the Kr00k vulnerability [5] was widely found in Wi-Fi chips used in wireless access points, routers and terminal equipment, which affected more than one billion Wi-Fi devices.

Thousands of IoT devices on the Internet pose security risks. For example, traditional home appliance manufacturers may lack knowledge in the design of security systems; thus, defective software components are reused in multiple different models, and security vulnerabilities are not repaired in time [6]. Also, users may lack the knowledge and

willingness to update their system firmware. All these can make the devices vulnerable. To track and protect these IoT devices, it is necessary to obtain their detailed information via device identification. Hence we can learn about the current security status of devices and take timely protection measures.

In the literature, fingerprinting and rule-based identification are two main methods used to mark devices. The former uniquely identifies a device by extracting its features, but a large amount of training data makes it hard to meet the rapidly growing demand for IoT devices. Also, the classification based on fingerprints is abstract, and administrators need to manually mark the data set, which has poor scalability for new devices. By contrast, rule-based methods extract concrete information from application layer response data. However, the writing of rules is usually an arduous manual process, and the coverage is limited due to incomplete information. Moreover, the existing identification solutions have limited granularity and cannot provide detailed information about the firmware version. Therefore, they cannot perform an accurate identification, since the vulnerabilities are highly related to the specific model and software/hardware versions.

1.1 Examples of scenarios

Based on the requirements of precise and accurate device identification, and due to the wide variety of IoT devices and different protocols in use, this paper mainly focuses on the wireless router, which is an essential entrance of the IoT. To illustrate our motivations, we preset the following application scenario.

Figure 1 shows an example scenario, in which user terminals, such as personal computers, tablets, and smartphones, access the Internet through the router. The routers may come from various vendors (Router I&II), have different models (Router II&III) and firmware versions (Router III&IV). As security software providers such as Norton, 360, and Kaspersky, have to manage and protect these devices from various users in different

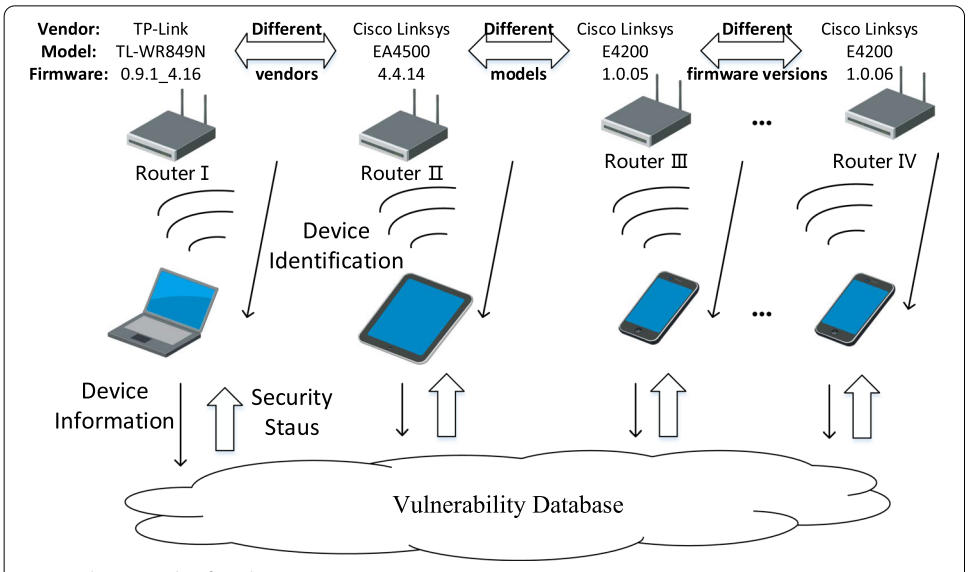

Fig. 1 The example of application scenarios

security states uniformly, we need an effective and accurate router identification scheme to perform a real-time assessment of security risks. Further, such function can be integrated into a WLAN management module of current mobile operating systems like Android, which directly manages connected network devices and provides users with security services. Another possible usage is in a campus or corporate network with a large number of wireless access points. Devices with different firmware and in distinct security states require better strategies for unified management. Network administrators need to take timely measures to apply security patches to vulnerable devices. Thus, there is a need to perform a precise and detailed device identification of the connected router locally, and to obtain the current security status of the device by requesting vulnerability data from the cloud database.

In summary, the premise of assessing the security status of routers is detailed and fine-grained device identification. Therefore, existing universal proposals [6–11] (analyzed in Sect. 4.3) for identifying IoT devices cannot meet the requirements from above application scenarios. Hence this work aims to establish a particular router version identification method to detect the device vulnerabilities accurately.

We attempt to obtain details that contain the vendor, device model, and firmware version, through automated device identification. However, due to the significant differences among the firmware of devices and the design of their system, it is extremely difficult to achieve such accurate identification and high versatility to ensure the correctness and information validity. Through actual observation and analysis of the product, we find that the admin websites of these wireless APs often contain detailed device information. To explore the web information is a promising solution, but there are two main challenges that hinder the automation. Firstly, the design is entirely different regarding the backstage management system of various manufacturers' devices. It is impossible to enumerate the corresponding access rules for each product manually. Secondly, the design style of webpage structure is also quite different, and general rules have to be established to extract devices' identification information accurately.

1.2 Contributions

The general idea of our approach is explained as follows. We simulate the login to the admin website and locate the webpages containing the device information according to the access rules established in advance. By grabbing the webpage and performing semantic analysis, we should obtain the identification information of the device. Then, we collect vulnerability information on the Internet through web crawlers and build a comprehensive and continuously updated vulnerability database. By querying the vulnerability database with the identifications, we can obtain the known vulnerabilities and related information of the devices. Hence we can determine the security status of wireless APs, and take proper measures to protect the device in time.

Based on the above idea, we design a framework called *WNV-Detector*, which can provide a universal, accurate, and scalable method of router version identification. Access rules are established to obtain device information pages based on semantic analysis, and *named entity recognition* (NER) would be leveraged to generate parsing rules to extract key data. In this way, the device identification can be automated. In addition, we provide a semi-automated method for rule updates with minimal user

involvement. When the device identification is completed, we utilize the information to perform a quick scan of known vulnerabilities by querying the previously built vulnerability database. Compared with traditional detection schemes, our method makes the granularity of device identification accurate to the firmware version level for the first time, and different from the abstract classification, it provides specific device tags. Moreover, the deployment difficulty of our solution is greatly reduced, without the need for a large number of data sets and training costs.

An earlier version of this paper was presented at [12]. In this journal paper, we make the following extensions. We redesign the framework of the detection of wireless network vulnerabilities, especially the rules for device identification. We introduce the implementation of the rule generator in detail, including access rules, parsing rules, entity checker and semi-automated rule updates, which greatly improved the accuracy of recognition and the scalability of the framework. We develop the WNV-Detector prototype using Python language, and select 22 common wireless routers on the market to conduct a comprehensive assessment of WNV-Detector, and achieve an overall vulnerability detection rate of 95.5%. Moreover, we evaluate the framework in terms of identification accuracy, rule effectiveness, and the scenario of visitors without administrator rights. The results show that our solution has advantages in terms of identification granularity, concreteness of device labels, and ease of deployment.

Our contributions of this work can be summarized as below.

- We propose a universal and extensible framework called WNV-Detector, which realizes automated and fine-grained device identification, and efficiently detects security vulnerabilities in wireless routers.
- We discover the availability of the device admin website, design access rules based on semantic analysis and utilize NER to generate parsing rules to extract valid information, and realize the identification accurate to the firmware version.
- We implement the prototype system of WNV-Detector, and design experiments to evaluate the effectiveness of our approach. Our results show that WNV-Detector can achieve high-precision and efficient vulnerability scanning, and provides more granular and specific device identifications than existing tools without the need of large-scale training data.

The remainder of this paper is organized as follows. Section 2 overviews the background of device identification, as well as IoT vulnerability life circle. Section 3 details the design of our framework and implements the prototype system. Section 4 evaluates our proposed WNV-Detector and discusses some open challenges. Finally, we conclude our work in Sect. 5.

2 Related works

2.1 Device identification

Device identification amis to identify the device by its characteristics or exclusive identifier uniquely. Currently, there are two major methods for identifying devices: rule-based identification and fingerprinting.

2.1.1 Rule-based identification

In the literature, the most commonly used technique for IoT device identification is banner grabbing. It is a technology for obtaining the system and running service on open ports of devices on the network. Netcat [13] and Nmap [14] are tools commonly used for interacting with the host system over the network. Shodan [15] and Censys [16] are two search engines that search online devices in cyberspace, and generating search results by auditing banners generated by ports on various devices.

Traditional banner grabbing requires the manual establishment of rules to extract effective information from a large amount of application-layer response data, which is difficult to meet the rapidly growing demand for IoT devices. Feng et al. [8] proposed automated acquisitional rules and realized the extraction of device identification through natural language processing. But the response layer data does not always contain useful information, resulting in limited coverage. Moreover, the granularity of device identification is also insufficient, which cannot meet the requirements of security assessment.

2.1.2 Fingerprinting

Device fingerprinting is the process of gathering information to generate device-specific signatures and identify individual devices [17]. For instance, Gao et al. [18] proposed a passive fingerprint identification method for wireless APs, by applying wavelet analysis to understand the sequence of export data packets, and using the heterogeneous characteristics from various vendors to distinguish different APs. GTID [7] used statistical techniques to capture the time-varying behavior of network traffic, analyzes the time arrival interval of data packets, and identifies the brand and model of the device. IoT Sentinel [6] uses the bursts of network traffic during the setup phase to identify the type of IoT devices. DÏoT [9] leverages untagged crowdsourced data captured in the client IoT to autonomously learn the device type identification models and anomaly detection models. PrEDeC [17] relies on the analysis of encrypted WiFi traffic, and identify device types via a random forest classifier that is trained by extracted features from Wi-Fi frames.

Fingerprinting is usually a classification problem, which requires establishing a function to classify each device correctly. Machine learning technique is often used for classification, which maps inputs to class labels based on training data. However, compared with a limited number of operating systems and specific application versions, the number of IoT devices is vast and numerous, so that it is quite difficult to collect a large amount of training data to achieve high-precision coverage. Moreover, the classification of device fingerprints is abstract (e.g. manufacturer #1, model #2), and the specific correspondence needs to be manually identified by the administrator in advance. Also, the addition of new devices requires re-training, resulting in poor scalability. Fine-grained identification of firmware and system version has become an impossible task.

In summary, traditional device identification methods are limited by the integrity of rules or training sets, and cannot meet the identification requirements of IoT devices with different models and firmware versions. And for the ever- increasing number, the cost of updating the identification rules is unbearable.

2.2 IoT vulnerabilities

Vulnerabilities are weaknesses in a system or its design that can be exploited by intruders to execute commands, access unauthorized data, and perform denial of service attacks. In IoT systems, both system hardware and software are often flawed in design, leading to vulnerabilities [19].

Generally, an IoT security vulnerability has the following life cycle [20]. At first, a vulnerability can be discovered by the vendor, third-party analysts, or hackers. If it is identified by a hacker, then the security risk is very high. In the next stage, the vulnerability would be publicly disclosed. The usual practice is to notify the manufacturer for reaction and recover before publishing it to the public. As the hacker community is active in developing and releasing zero-day attacks, it may further increase the level of security risks [21]. When all users installed the patch and fixed the bug, the life cycle of this vulnerability ends. During the whole period, every stage can be exploited by hackers.

When vulnerabilities are publicly posted on the Internet, they are often accompanied by specific exploitation steps. On the NVD [22], CVE [23], and many other enterprise vulnerability platforms, the vulnerability details are often organized. In the research community, Feng et al. [24] found that more than 80% vulnerability reports are published with detailed exploitation steps, which is extremely easy for hackers to exploit. While a large number of vulnerabilities are disclosed through various sources on the Internet, without maintaining confidentiality and notifying vendors. If users and vendors are not aware, the life cycle of a vulnerability can be active for up to five years.

3 WNV-Detector

3.1 Threat model

In this work, we consider that an adversary can use existing and publicly available vulnerability reports on the web to compromise wireless routers, especially devices that are connected to the Internet. Attackers can remotely exploit the security vulnerabilities of these devices, obtain privilege to manage wireless routers and execute malicious code. As a result, attackers can further invade the internal network, by violating user privacy and endangering system security.

3.2 Overview

Human analysts can query and obtain the device identification information (e.g., vendor, model, firmware version) of wireless routers, search the vulnerable platform via the Internet, analyze the security status of devices, and formulate the subsequent security measures. While this method is not scalable to the wireless router market due to hundreds of manufacturers and tens of thousands of product models.

To meet the requirements of large-scale and precise device identification, we design and implement an automated, scalable, high-precision and device-independent framework (called *WNV-Detector*) for detecting wireless network vulnerabilities. Fig. 2 depicts the infrastructure of WNV-Detector. In particular, our approach generates access rules and parsing rules for device identification based on the rule generator, and scans known vulnerabilities of corresponding versions of devices with identification tuples.

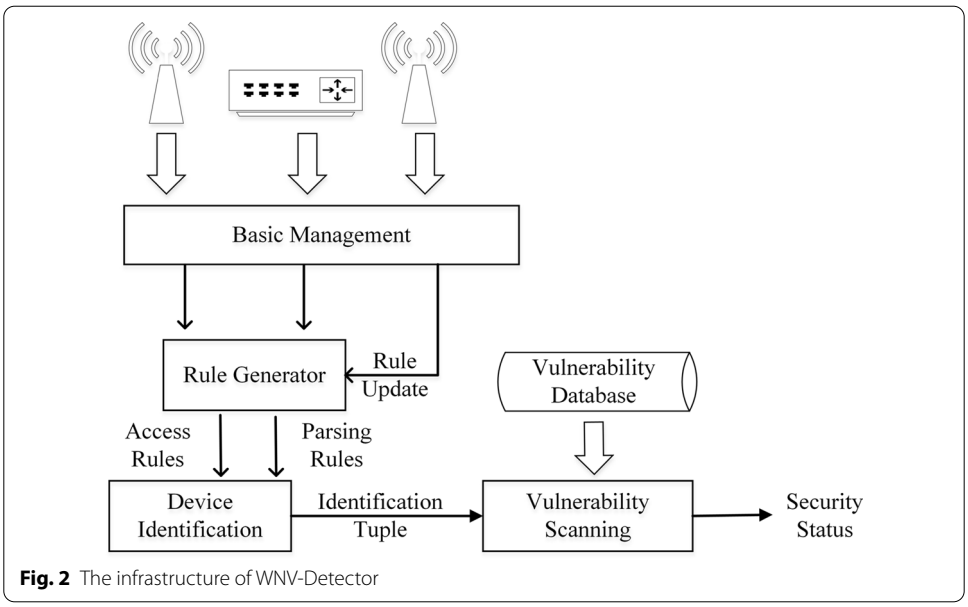

Fig. 2 The infrastructure of WNV-Detector

3.3 Device access

WNV-Detector manages the wireless AP/router of the access system unified. The basic information of the device is maintained here, while the admin account and password are utilized for the simulated login during the device identification phase. Note that with the aim to protect the security of the device, it is acceptable for the security platform to require the management password at the initial access. Regarding the device owner or network administrator, the information is clearly accessible. WNV-Detector does not store the management key locally, instead, it can be managed uniformly by the security software platform or the online password manager. This is acceptable for ensuring the security of devices and networks. Related application examples can be found in Google Play [25].

3.4 Rule generation

In WNV-Detector, we develop a rule generator that automatically generates rules to identify the device information of wireless routers with minimal human intervention. Figure 3 illustrates the workflow of our rule generator. The goal of access rule is to find the one containing device information from a series of admin pages of the router, while the use of parsing rule is to accurately identify the device identification tuple from the possible pages. The entity checker aims to guarantee the reliability of the device identification for rule extraction, and a semi-automated method is used to update rules with minimal user intervention.

3.4.1 Access rules

How to automate the process of accessing the device information page is a challenge. Intuitively, it seems to be a feasible way of using a web crawler to grab all pages on the

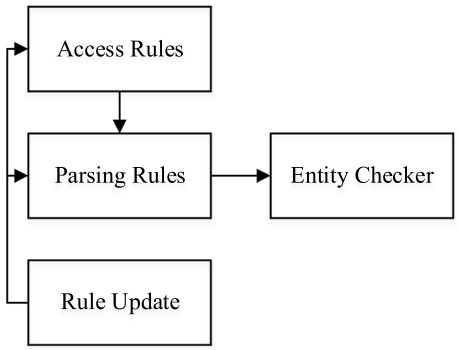

Fig. 3 The workflow of rule generator

admin website, but there would be a large amount of useless data. It is quite inefficient if we parse and filter them all. For the sake of the completely different design of webpages, it is difficult to find a common way to get the information fleetly.

For this issue, we analyze a large number of routers from the market and collect device information manually. By observing the results, we find that except for a few devices that can directly display the version information on the homepage, most of them need to reach the device information page by clicking on a series of menu items. For example, a model of TP-Link router needs to click on a set of labels *Maintenance-> System Tools-> Firmware*.

In practice, we find that the access path of most devices may contain similar keywords, such as *administration*, *status*, *upgrade*, *setting*, etc. To achieve a rapid acquisition of the information page, we first aim to build a keyword dictionary. By retrieving the tags with these keywords, clicking on these links sequentially, and getting all relevant pages via a breadth-first crawler, we can greatly improve the efficiency of information acquisition. This method is highly dependent on the completeness of the keyword library, which is simple but easy to perform, and offer high identification accuracy. However, the information has to be collected manually in advance, resulting in a scalability issue. That is, the method of manually establishing a rule base cannot meet the needs of large-scale device identification.

To overcome this issue, we propose an automated acquisition method based on semantic analysis and machine learning. We capture and analyze the access paths of these devices, and notice that high semantic similarities exist among these menu labels. In addition, there is a strong dependence, and the form is similar among different devices. If a model can be built to analyze these semantic similarities, automatically learn and generate rules for access paths, the efficiency of obtaining information can be significantly improved.

In this work, we use the Word2vec framework [26] to implement the analysis. Due to the shortcomings of unsupervised training, we grab 240 webpages from 22 routers, extract data and build a corpus. In addition, part of the training data comes from news websites. With the training data, a continuous bag-of-words model (CBoW) can be established. Each word is mapped to a vector to represent the relationship between words. In the actual acquisition, we crawl the pages, extract all the label texts, and

calculate the similarity with the keywords from the previously trained model. If a specified threshold is reached, then the page can be considered to contain device information. For example, if we set the keyword as *setting* and the threshold as 0.7, we can obtain the labels of related phrases such as *Network Settings*, *Wi-Fi Settings*, *Advanced* and *System* from the admin page of a certain router. The pages labeled by these tags may contain the required device information. After analyzing these semantic similarities, the rules of the access path can be generated according to the dependency relationship.

3.4.2 Parsing rules

After obtaining the relevant webpages, we can extract the information of devices. Due to the different interfaces of major manufacturers, the related terms are mostly inconsistent, which brings a great challenge to our analysis. Focused on this issue, we define the tuple of device identification as *(vendor, device model, firmware version)*. Table 1 provides some examples of device identification.

The main purpose is to quickly process the collected HTML documents and identify the required information from a large amount of irrelevant and useless data. In this work, the rule generator leverages Natural language processing (NLP) techniques to parse the identification tuple. For page parsing, we use *named entity recognition (NER)* to help extract entities from the webpages. However, NER is highly domain-specific; that is, even the most advanced NER system may only perform well in a particular domain. Hence there is a need to tune the NER system in our case.

To achieve high-precision identification of device information, we tune the NER system according to our actual requirements. The vendor information can be extracted from existing data, the device model can be obtained via NER based on regular rules, and the firmware version can be known via NER based on semantic rules. In the implementation, we use the python Beautiful Soup package [27] to parse the grabbed pages, and use the natural language toolkit (NLTK) [28] to perform NER on the processed webpages.

Next, we introduce the detailed generation process of parsing rules for extracting identification tuples.

(a) Vendor Our acquisition of vendor information depends on the establishment of a device vulnerability database. Since each piece of vulnerability information on the Internet usually contains the manufacturer and the model of affected devices, it is feasible for us to build the identification tuples. In the evaluation, we collected data information

Table 1 Examples of device identifications

Vendor	Model	Firmware version
Linksys	X3500	v1.0.01
D-Link	DIR-619L	B1_2.02
ZTE	E5501	V1.0.0_V1.0.2
Netgear	WGT624	V4.2.11_1.0.14
Xiaomi	Mi Router 3	2.22.15

from 305 vendors and built a vendor dictionary for identification. It is worth noting that the vendor dictionary can be easily expanded.

(b) Device model We can obtain the device model by leveraging the NER system based on regular rules. The model is usually alphanumeric and may contain some symbols. A specific example of device model on the webpage is shown in Fig. 4(a). It can appear directly on the webpage title, or after some keywords like *device name*. Due to the large number of device models from major manufacturers and the rapid production update, it is an open challenge to find a universal rule that can cover all the devices. In this work, after analyzing our purchased devices and relevant vendors, we use the regular expression *[A-Za-z]+[-_]?[A-Za-z!]*[0-9]+[-_]?[A-Za-z0-9]**, which can cover most devices in our evaluation. The device model can be found if the regular expression is satisfied.

(c) Firmware version The NER based on semantic rules is used to extract the firmware version. Figure 4b shows an example of firmware version on the webpages. We search for version-related keywords such as firmware, software and hardware on the webpages and analyze the DOM tree structure to extract the tags that may contain the version number. While a simple analysis may lead to a higher false-positive rate, since these keywords may also appear in other places, play a role of description, description, guidance, etc. To address this issue, we develop parsing rules based on semantics and part of speech. In particular, we process the word segmentation and filter out the tags with irrelevant version numbers such as verbs, conjunctions and determiners. For example, *[('current', 'JJ'), ('firmware', 'NN'), ('version', 'NN')]* is highly correlated with the firmware version, while *[('No', 'DT'), ('later', 'JJ'), ('version', 'NN')]* obviously does not contain the valuable information. Then, according to the DOM tree structure, we can obtain the potential firmware version numbers by sequentially searching the adjacent tags.

3.4.3 Entity checker

To further improve the rule accuracy, we design an entity checker module to verify whether the obtained tuple of device identification is reliable and correct. The detection process can be divided into two steps. First, we check the overall entity relevance and

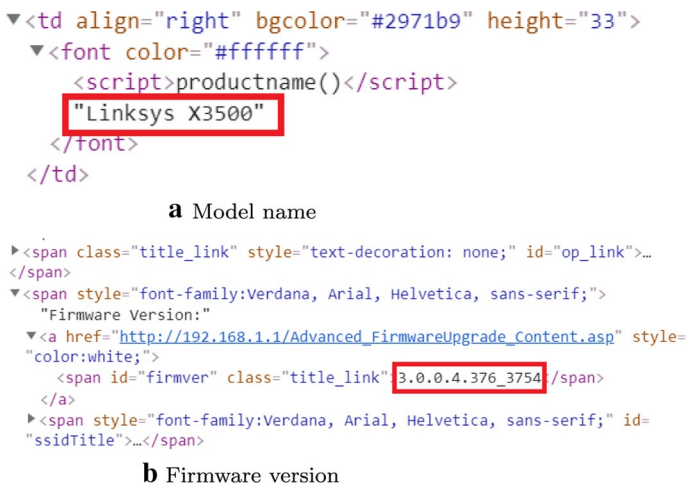

Fig. 4 Identification information on webpages

detect meaningless information. The identification tuple can be leveraged as the keyword via Google search, where we compare the cosine similarity between the entity and the title of searching results. If the similarity is extremely low, the entity can be considered as irrelevant, which is not the device information of the router.

In the second step, we perform a reliability check on the device identification tuple. Based on the statistical results of the device information collected in advance, the credibility of the entity combination can be figured out. Specifically, the vendor, device model, and firmware version are given certain weights (e.g., 0.4, 0.3, 0.3). When a new device is connected to the system, we can calculate the credibility by the product of the weight and the frequency of each attribute combination in the vulnerability database. Taking the entity of *Tp-Link TL-WR886N 7.0_1.1.0* as an example, based on the entity checker, we can obtain the relevance of 0.63 and the credibility of 0.81.

3.4.4 Rule update

With more devices involved, existing rules may not be able to cover all the situations. For this issue, we design a semi-automated rule update method, where users can assist in the identification period with minimal intervention. A simple built-in browser based on the WebKit kernel is pro- vided to both network administrators and users. All users only need to manually access the webpage of device information and mark the identification tuple. In this process, the browser records the access path, and updates the access and parsing rule database. As a result, the coverage of our rules can be improved. Please note that, unlike device fingerprints, the update of the rules does not occur when each new device is connected and is only used in rare cases. The details will be evaluated in Sect. 4.

3.5 Device identification

Due to the significant differences in the design of firmware and user interfaces, it is a challenge to automate the process of device identification. In this work, we design a rule-based method that is workable for most devices from different manufacturers. Fig. 5 shows how to implement the device identification module, including simulated login, information collection, and entity extraction.

3.5.1 Simulated login

The simulated login is used to access the management page of routers. It is difficult to design a specific scheme for each style of pages manually. In this work, we use Selenium ChromeDriver [29] that can provide APIs to launch the Chrome browser and navigate

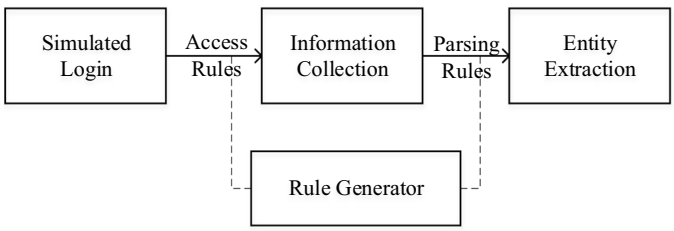

Fig. 5 The process of device identification

it to a particular URL. A headless browser is a kind of web browser without a graphical interface, which enables automatic control of webpages via command line. The purpose of combining the Selenium framework with a headless browser is to make automated access more easily.

The router authentication usually includes two major ways: *login form submission* and *HTTP basic authentication*. In general, the structure of login form is relatively single one or two input-textboxes (may not need to enter a username), which are usually the form of *input[type=text]* or *input[type=password]*, with a submit button (can be *input[type=submit]*, *input[type=button]*, *button*, etc). The Selenium framework is leveraged to capture these web elements, automatically fill out the form and press the button to simulate a login. The system determines whether the login is successful according to the change of page elements before and after a login trial. In addition, for some old devices that use basic access authentication, we add "username:password" to the URL string to avoid login prompts.

3.5.2 Information collection

As described earlier, we introduce how to automatically generate access rules, which can be used to collect information. The rule generator collects the key information from webpages, and applies the previously trained model for generating rules for access paths automatically. According to the access rules, we can fleetly track and visit the webpages containing device information, grab and wait for the next round of analysis and processing. Please note that when the router is connected in guest mode, and the management authority cannot be obtained, the module can still work independently, scanning the reachable pages for limited device identification.

3.5.3 Entity extraction

Similarly, we leverage parsing rules to extract the target entities from the collected information pages. The obtained device identification tuple, including the vendor, device model and firmware version, would be the final output of the device identification phase and the input to the vulnerability scanning phase. The results with low relevance and reliability would be sent to users for judgment, i.e., whether to update the rules.

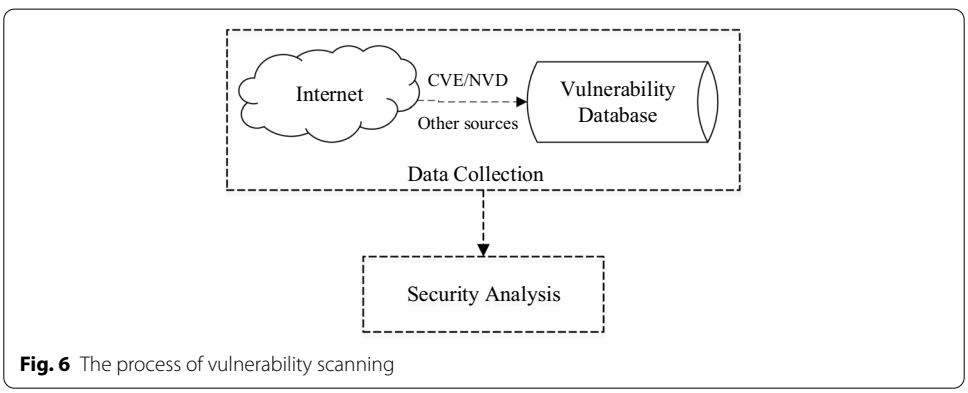

Fig. 6 The process of vulnerability scanning

3.6 Vulnerability scanning

In this section, we introduce the design and implementation of vulnerability scanning. As shown in Fig. 6, the module includes the collection of vulnerability data and security analysis of the device. The identification tuple is used as input and the detection is performed by querying the AP vulnerability database.

3.6.1 Data collection

The major vulnerability platforms on the Internet can provide useful data support. Such data is usually well-archived for retrieval and analysis. Through the National Vulnerability Database (NVD) [22], we can obtain detailed information about the specified vulnerabilities, including published dates, descriptions, solutions, known affected software configurations, etc. Based on the aforementioned life cycle of a vulnerability, we can further obtain the latest vulnerability information from security mailing lists and public forums like [30].

3.6.2 Security analysis

Based on the information obtained during the device identification phase and the established vulnerability database, we can scan for device vulnerabilities. By matching the identification tuple (such as vendor, device model, firmware version) with the field of *affected entities* in the database, we can accurately identify known vulnerabilities and obtain the confirmed vulnerability information. If there is a security risk regarding the device, the system can give an alarm to remind the user of updating the firmware in time. Moreover, the system can provide the corresponding solution and countermeasure information for further judgment and processing.

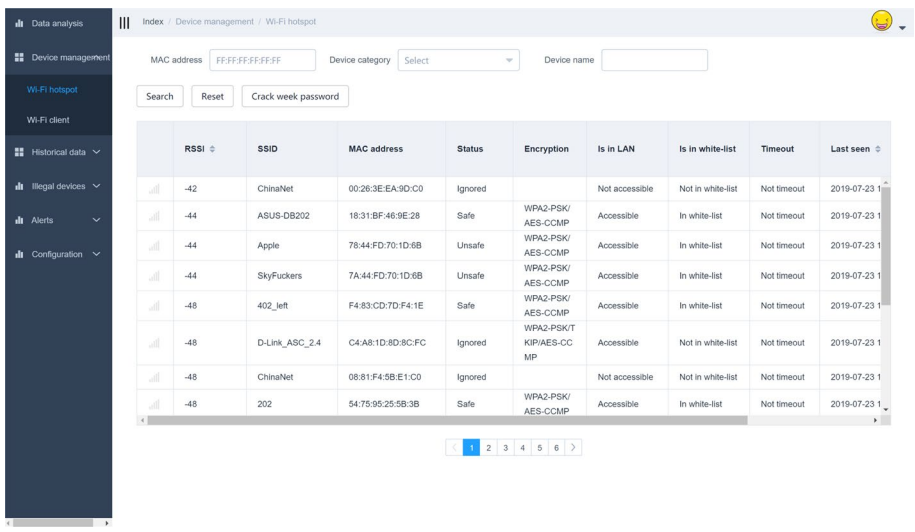

Fig. 7 User interface of the system

3.7 Prototype system

Based on the aforementioned framework, we build a prototype system to help detect wireless network vulnerabilities. Figure 7 shows the system and user interface. We can obtain the nearby Wi-Fi hotspots in real-time, and can effectively manage intranet APs after entering the basic information of a device. The detailed information can be obtained via the rule-based device identification. Then we can use the identification tuple to perform vulnerability scanning and determine the security status of the device.

4 Experimental evaluation

In this section, we perform an evaluation to investigate the performance of WNV-Detector. We first introduce the experimental settings and show how the system can identify most router models from major manufacturers in an accurate way. We also compare WNV-Detector with some similar security mechanisms like the intrusion detection system (IDS). In the end, we analyze the collected vulnerability data and openly discuss the limitations of the system.

4.1 Setting

We select 22 routers from 13 vendors in our evaluation, such as Cisco, D-Link, Netgear, Huawei, ASUS, etc. We take into account the market share and device sales of major manufacturers. The selections are mainly based on their representativeness and the coverage, where most of our selected devices are still on the market. In addition, we also selected some old devices that cannot be easily replaced by new devices. It is worth noting that these old devices may not have the latest factory firmware. Some of the devices used in our experiment are shown in Fig 8. The experiment was carried out in a general household scene. In addition to the wireless routers we manage, other wireless signals can also be received. We include all the devices in the system management and apply the proposed scheme for evaluation. We evaluate the overall

Fig. 8 Part of the devices used in the experiment

performance of the WNV-Detector framework, and apply access rules and parsing rules separately to observe the effectiveness of the rule generator. WNV-Detector is provided for network administrators or device owners by default. For visitors without sufficient permissions, we also apply rules on reachable pages to evaluate the success rate of device identification. We compare the output by the system with the correct data collected manually to determine whether the identification is successful, that is, the device label (vendor, device model, firmware version) is correctly obtained.

4.2 Results

To provide a comprehensive and objective evaluation of WNV-Detector, we design the following tests to understand the algorithm reliability and the system performance.

4.2.1 The overall performance of WNV-Detector

The system runs a complete process from device access, including device identification and vulnerability scanning, and finally obtains the security status of the device. We scan all the above 22 devices, and the overall success rate reaches 95.5%. That is, the complete process is correctly executed, and the security information of the corresponding models is obtained from the vulnerability database.

4.2.2 The success rate of device identification

The device identification process is highly dependent on the integrity of rule database, and is susceptible to interference from various uncontrollable factors. At the current stage, we only utilize the initial automated-generated rules to identify the device sequentially through simulated login, information collection, and entity extraction. A success rate of 86.3% is achieved in the comparison with the benchmark information previously collected manually. We then perform the semi-automated rule update and after re-detection, the success rate could increase to 95.5%.

4.2.3 The dependability of access rules

While generating access rules, we take advantage of the similarity between the path label text and utilize the Word2vec framework to build the model. The choice of similarity threshold will affect the efficiency and reliability of rule generation. We set a threshold of 0.62, and access the information page through a path like *Administration->Software Upgrade*. In the test of all devices, we successfully reach the target pages, and the average number of pages visited is 2.1.

4.2.4 The reliability of parsing rules

As mentioned earlier, the parsing rule uses NER to grab the target entity from the admin page. To evaluate the reliability, we manually collect the information page, execute the parsing rules, and obtain the device identification tuple (including vendor, model, firmware version). Based on the prior experience, we set the threshold of

relevance and credibility as 0.22 and 0.65 for entity checker, respectively. It's worth noting that a low threshold of relevance is selected here. In fact, for correct identifications, due to the replacement of multiple software versions, it is not always possible to retrieve consistent information. In addition, even if the identification of a non-IoT device conforms to our regular rules, such as "PAU EH12A", we can obtain a very low relevance score (e.g., 0.04) from search results. Therefore, our rules can effectively screen legitimate entities. In this way, we set up and verified the above 22 routers and more than 1,900 device IDs in the vulnerability database. An accuracy rate of 100% confirms the reliability of our parsing rules.

4.2.5 The situation of visitors

For other visitors connected to the router, they do not have the authority to obtain the management key. WNV-Detector can achieve available information from the reachable pages without logging in. By scanning the above-mentioned devices as visitors, we obtain the vendor for 86% of the devices, and realized the identification of the model for 68% of the devices.

Result. The above results indicate the effectiveness and reliability of our WNV-Detector. More specifically, by querying the vulnerability database, it is found that 7 out of 22 devices had security risks, and 45 CVE vulnerabilities were detected. Table 2 summarize the evaluation result.

4.3 Comparison

Device identification is often helpful to various security mechanisms like IDSs in determining relevant security solutions for different devices. We compare our WNV-Detector with some similar schemes as below.

Gao et al. [18] apply wavelet analysis to the resultant packet sequence and leverage the heterogeneity of the device to identify wireless APs from various manufacturers. OWL [11] utilizes fields and parameters as key-value pair features, and builds a Multi-view wide and deep learning (MvWDL) model to identify devices. For IoT Sentinel [6], GTID [7], and WDMTI [10], they aim to capture data packets and build a classifier based on random forests, artificial neural networks (ANN), k-Nearest Neighbors (kNN), and Hierarchical Dirichlet Process (HDP), respectively. The main limitation of these general device fingerprinting schemes is that collecting a data set sufficient to identify the firmware version is a herculean task and requires a large number of training data sets. In

Table 2 Evaluation result

Device	Vulnerabilities
TP-Link TL-WR886N 7.0_1.1.0	16
Tenda AC7 V15.03.06.44_cn	12
ASUS RT-N56U 3.0.0.4.376_1665	7
D-Link Dir-816 A2 1.11	4
Huawei WS851 1.1.20	3
Netgear WGT624 V2_V4.2.11_1.0.1	2
Linksys WRT54G 4.30.4	1

Table 3 Comparison of similar schemes

Proposal	Feature/solution	Granularity
WNV-Detector	Rule-based parsing	Concrete firmware
Gao et al. [18]	Egress traffic	Abstract vendor
IoT Sentinel [6]	Burst traffic	Abstract model
GTID [7]	Time-varying of network traffic	Abstract model
ARE [8]	Acquisitional rules	Concrete model
WDMTI [10]	DHCP features	Abstract model
OWL [11]	Broadcast/multicast features	Abstract model

Table 4 Top 10 vendors by total number of vulnerabilities

Vendor	Vulnerabilities
Cisco	432
D-Link	95
Netgear	59
Zyxel	43
TP-Link	41
Linksys	38
ASUS	36
Tenda	30
Huawei	28
Belkin	22

addition, the model needs to be retrained for new devices, resulting in poor scalability. ARE [8] cannot extract more detailed information from the response data, resulting in insufficient identification granularity and degrading the performance. Table 3 shows the comparison between similar schemes.

Security vulnerabilities in IoT devices are much relevant to their firmware versions. Without accurate identification, it is hard to protect those devices in an appropriate manner. In the evaluation, we find that WNV-Detector can identify specific device models and versions, and thus have great advantages in granularity. In addition, there is no need for a special API that reports the corresponding version number, resulting in a wider range of applications. By considering these merits, we consider WNV-Detector can outperform and complement the existing approaches.

Security vulnerabilities in IoT devices are closely related to their firmware versions. As vendors release new upgrade patches, previous vulnerabilities are fixed and new security issues may also arise. Without accurate identification of the model and version information, devices can hardly be protected in a targeted manner. WNV-Detector is undoubtedly more advantageous in the identification and protection of wireless network devices.

4.4 Analysis

To build the vulnerability database, we collected many open resources on the Internet. Till January 27, 2021, a total of 1987 related vulnerability data have been collected. In

this section, we provide statistics and analysis of the collected data in the aspects of vendor, severity and category.

4.4.1 Vendor

The vulnerability data involved more than 1,900 devices from 305 vendors. According to the statistics, the number of vulnerabilities is positively related to the shipments of manufacturers' products and their market share. Table 4 presents the top 10 vendors according to the number of device vulnerabilities, accounting for more than 40% of the total.

4.4.2 Category

We extract the category information of each vulnerability from our collected data. As shown in Table 5, it is found that improper input validation, cross-site scripting and buffer errors are the most common types. A majority of these vulnerabilities may be caused during the code development process, due to the design or implementation errors.

4.5 Discussion

WNV-Detector collects critical identity information of wireless APs/routers based on rules automatically, and evaluates the security status of the device by scanning its known vulnerabilities. To the best of our knowledge, this is an early study regarding the device identification and vulnerability scanning of wireless routers. Below we discuss some open challenges and limitations, including insufficient information, data source, access key management, application scenario, and scalability.

4.5.1 Insufficient information

Generally, sufficient information about routers can be obtained in their admin pages and be leveraged for device identification. However, the system designs and user interface of major manufacturers are quite different, and some devices do not have enough information on their pages. For such case, we cannot obtain accurate information for device

Table 5 Vulnerabilities by type

CWE ID	Type	%
20	Improper Input Validation	12.5
79	Cross-site Scripting	9.4
119	Improper Restriction of Operations within the Bounds of a Memory Buffer	7.4
287	Improper Authentication	6.2
352	Cross-Site Request Forgery	5.8
78	OS Command Injection	5.2
-	Insufficient Information	4.8
787	Out-of-bounds Write	4.5
200	Exposure of Sensitive Information to an Unauthorized Actor	4.3
264	Permissions, Privileges, and Access Controls	3.4

identification. At this point, it can be supplemented with relevant information on the Internet, such as Wikipedia and the vendor support websites.

4.5.2 Data source

The purpose of WNV-Detector is to scan known security vulnerabilities in devices, rather than undisclosed zero-day vulnerabilities. In this case, the data source has a significant impact on the evaluation results. In practice, a considerable number of vulnerabilities may have not been included in various official vulnerability databases like CVE and NVD, which can degrade the effectiveness of our approach and other similar schemes.

4.5.3 Access key management

WNV-Detector requires the admin account and password of the router to provide the most fine-grained device identification. The information is held by the device owner or the network administrator and should not be available to other visitors connected to the AP. For WNV-Detector, the access key is not stored in the local database to prevent additional security risks. Instead, it can be managed by the security platform (such as 360 [31] or Synology Router Manager [32]), or saved by online password managers (e.g. Google Password Manager and LastPass).

4.5.4 Application scenario

As introduced early, we show the application scenario for WNV-Detector. As security service providers have to ensure the security of their devices and access networks, there is an increasing need to manage wireless routers that provide key functions. WNV-Detector complements the existing schemes and uniformly manages the different models and version numbers of routers. Another scenario is in a large enterprise network, WNV-Detector helps network administrators to manage devices in different security states uniformly. WNV-Detector can identify the device accurately, obtain the status of the device in time, and report the vulnerable device to ensure the security of the internal network. In the future, we plan to consider some other application scenarios and validate the performance of WNV-Detector.

4.5.5 Scalability

Our WNV-Detector proposed in this paper aims to identify vulnerabilities at wireless APs and routers, while with a unified device management website, it can be also extended to other IoT devices, such as gateways, smart cameras and speakers. For example, we can obtain the information of other managed devices through the portal of smart home platform to perform detection of known vulnerabilities.

5 Conclusion

The security of IoT devices has received widespread attention in recent years. We observe that most existing methods cannot perform an accurate device identification, as the device vulnerabilities are highly related to the specific model and software/hardware versions. In this paper, we focus on the wirelests router that is an important

entrance of IoT, and propose WNV-Detector, a universal and scalable framework for detecting wireless network vulnerabilities. It can automatically generate rules to identify wireless APs and perform a quick scan and security risk assessment on the devices. We also implement a prototype of WNV-Detector and perform experiments to evaluate its performance. Our results indicate that our WNV-Detector can provide high reliability and high accuracy. Based on our collected data, we identify that there are a large number of devices with known vulnerabilities on the market, which are highly vulnerable to various threats. In the future, we hope to extend WVN-Detector to more IoT devices, and further study the identification of various types of vulnerable devices that apply different protocols. Our work attempts to stimulate more research in enhancing the security of wireless devices and networks.

Abbreviations
IoT: Internet of Things; NER: Named entities recognition; NLP: Natural language processing; AP: Access point; WLAN: Wireless Local Area Network; NVD: National Vulnerability Database; CVE: Common Vulnerabilities and Exposures.

Authors' contributions
YH and LL conceived the study. FZ designed the study and wrote the manuscript. WM and LT collected the data and revised the manuscript. SH and RY drew the figures and improved the structure of the manuscript. All authors read and approved the final manuscript.

Funding
This work is supported by the Aeronautical Science Foundation of China under Grant 20165515001, the National Natural Science Foundation of China under Grant No. 61402225, State Key Laboratory for smart grid protection and operation control Foundation, and the Science and Technology Funds from National State Grid Ltd.(The Research on Key Technologies of Distributed Parallel Database Storage and Processing based on Big Data). Weizhi Meng is also supported by H2020-SUICT-03-2018: CyberSec4Europe.

Availability of data and materials
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China.
[2] Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark.

Received: 8 February 2021 Accepted: 30 March 2021
Published online: 07 April 2021

References

1. X. Liu, X. Zhang, Noma-based resource allocation for cluster-based cognitive industrial internet of things. IEEE Trans. Ind. Inf. **16**(8), 5379–5388 (2019)
2. X. Liu, X.B. Zhai, W. Lu, C. Wu, Qos-guarantee resource allocation for multibeam satellite industrial internet of things with noma. IEEE Trans. Ind. Inf. **17**(3), 2052–2061 (2019)
3. X. Liu, X. Zhang, Rate and energy efficiency improvements for 5g-based iot with simultaneous transfer. IEEE Internet Things J. **6**(4), 5971–5980 (2018)
4. M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, et al. Understanding the mirai botnet. In *26th USENIX Security Symposium, USENIX Security*, pp. 1093–1110 (2017)
5. A Vulnerability in Broadcom Wi-Fi Client Devices. https://nvd.nist.gov/vuln/detail/CVE-2019-15126
6. M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, S. Tarkoma. Iot sentinel: automated device-type identification for security enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2177–2184 (2017). IEEE
7. S.V. Radhakrishnan, A.S. Uluagac, R. Beyah, Gtid: a technique for physical deviceanddevice type fingerprinting. IEEE Trans. Depend. Secure Comput. **12**(5), 519–532 (2014)

8. X. Feng, Q. Li, H. Wang, L. Sun, Acquisitional rule-based engine for discovering internet-of-things devices. In *27th USENIX Security Symposium, USENIX Security)*, pp. 327–341 (2018)
9. T.D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, A. Sadeghi, Dïot: A federated self-learning anomaly detection system for IoT, pp. 756–767 (2019)
10. L. Yu, T. Liu, Z. Zhou, Y. Zhu, Q. Liu, J. Tan, Wdmti: wireless device manufacturer and type identification using hierarchical dirichlet process. In *2018 IEEE 15th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 19–27 (2018). IEEE
11. L. Yu, B. Luo, J. Ma, Z. Zhou, Q. Liu, You are what you broadcast: identification of mobile and IoT devices from (public) wifi. In *29th* {USENIX} *Security Symposium* ({USENIX} *Security* 20), pp. 55–72 (2020)
12. F. Zhu, L. Liu, W. Meng, T. Lv, S. Hu, R. Ye, Scaffisd: a scalable framework for fine-grained identification and security detection of wireless routers. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1194–1199 (2020). IEEE
13. Netcat, Network Debugging and Investigation Tool. https://nc110.sourceforge.io
14. Nmap, the Network Mapper. https://nmap.org/
15. J. Matherly, Complete guide to shodan. Shodan, LLC (2016-02-25) **1** (2015)
16. Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, J.A. Halderman, A search engine backed by internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 542–553 (2015)
17. Q. Xu, R. Zheng, W. Saad, Z. Han, Device fingerprinting in wireless networks: challenges and opportunities. IEEE Commun. Surv. Tutor. **18**(1), 94–104 (2015)
18. K. Gao, C. Corbett, R. Beyah, A passive approach to wireless device fingerprinting. In *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*, pp. 383–392 (2010). IEEE
19. M. Abomhara et al., Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. J. Cyber Secur. Mob. **4**(1), 65–88 (2015)
20. M. Shahzad, M.Z. Shafiq, A.X. Liu, A large scale exploratory analysis of software vulnerability life cycles. In *2012 34th International Conference on Software Engineering (ICSE)*, pp. 771–781 (2012). IEEE
21. R. Anderson, Security in open versus closed systems—the dance of Boltzmann, Coase and Moore. Technical report, Cambridge University, England (2002)
22. U.S. National Vulnerability Database. https://nvd.nist.gov/
23. Common Vulnerabilities and Exposures. http://cve.mitre.org/
24. X. Feng, X. Liao, X. Wang, H. Wang, Q. Li, K. Yang, H. Zhu, L. Sun, Understanding and securing device vulnerabilities through automated bug report analysis. In *Proceedings of the 28th USENIX Security Symposium* (2019)
25. Search Results of Router Manger in Google Play. https://play.google.com/store/search?q=router
26. T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
27. Beautiful Soup, a Python Package for Parsing Documents. https://www.crummy.com/software/BeautifulSoup/
28. S. Bird, Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pp. 69–72 (2006)
29. Selenium, Web Browser Automation. https://www.selenium.dev/
30. Full Disclosure Mailing List. https://seclists.org/fulldisclosure/
31. Router Manager Keeps Your Network Environment Secure. https://blog.360totalsecurity.com/en/router-manager-keeps-your-network-environment-secure/
32. Revolutionize Your Networking Experience. https://www.synology.cn/en-global/srm

Publisher's Note