How Vulnerable Is the Public WiFi AP You Are Using?

Ruming Tang†, Haibin Li†, Kaixin Sui†, Zihao Jin†, Xiao Yang‡, Dan Pei†* , Beichuan Zhang§
†Tsinghua University ‡Guizhou University §University of Arizona
†Tsinghua National Laboratory for Information Science and Technology (TNList)

***Abstract*—Millions of WiFi APs are deployed in residential homes and small businesses, exposed to the Internet and/or public use. While WiFi is the primary Internet access method now, there lacks a public awareness on AP security vulnerability. In this paper, we focus on public WiFi APs and propose a fast and accurate approach for measuring an AP's potential vulnerability, and build a mobile App called SAVY that can inform a user about "How vulnerable is the public WiFi AP that you are using." SAVY leverages the fact that majority of the APs typically do not update their softwares, and dynamically maintains a table that maps an AP model to its latest measurement results with scoring metrics. In the wild, SAVY uses a lightweight method to probe the AP's model, and then the score can be easily obtained via searching the mapping table. Our measurement results show that, compared to full vulnerability probing, SAVY reduces the probing time by 93.7%, and achieves a scoring accuracy of 90.5%. Our results also show that AP vulnerability is prevalent in the wild, and the vulnerability scores vary across vendors, models, and stores.**

I. Introduction

Public WiFi has become part of our lives as millions of people use it every day for work and leisure. When you sit in a coffee shop or walk around in a shopping mall, have you ever wondered how secure or how vulnerable is the WiFi access point (AP) that you are using? Compared with enterprise APs and cellular base stations, public APs are exposed to public use with posted or no passwords while not managed by full-time professionals. Compared with personal computers, smartphones, and tablets, public APs do not interact with users and usually do not have automatic security updates. Once purchased and deployed, majority of these commercial off-the-shelf (CoTS) APs are rarely updated but vulnerabilities are discovered and reported over time (e.g., the CVE database [1]), making them less and less secure, yet without public awareness.

Accessing, probing, and potentially exploiting these vulnerabilities on public APs are relatively easy because it often does not require a password or the password is posted. Furthermore there exist mobile applications for users to share WiFi passwords, *e.g.*, WiFi Master Key [2] has more than 800 million users. While easy access gives users the convenience of free Internet, it also opens the door for attackers to scan, probe, and exploit the network. Once the vulnerabilities are exploited by the attackers, the compromised APs can be used to launch serious attacks such as DNS redirect, phishing, and online ID/password theft [3], [4]. Therefore, given that WiFi is the primary Internet access method [5], [6] nowadays, we argue that public WiFi APs have become the weakest link of mobile Internet security.

A major factor in the lack of security of public APs is the lack of awareness of the problem. The APs do not interact with users or business owners to prompt for security updates, and vendors do not release software fixes in a timely manner. Given the large number of vendors and devices on the market, it is also difficult to determine which one is more secure or more responsive to security problems. We propose to change the situation by building a mobile App that can quickly scan a public AP and give a security score. Knowing the security score of an AP helps users and business owners choose which APs to purchase and use, and incentivize vendors to improve the AP's security. There are popular tools (such as [7], [8]) to scan and rank the security level of laptops and smart phones, and even facilitate the security patching update. However, to the best of our knowledge, there is no such tool for CoTS WiFi APs deployed in residential homes and small businesses.

Intuitively, such a tool would need to know (1) an AP's open ports and services, (2) the *actual* vulnerabilities on this AP, and (3) a quantitative measure (score) of the corresponding vulnerabilities. However, there are two major practical constraints when building such a tool. First, authoritative vulnerability database such as CVE/NVD [1], [9] only provides description of the vulnerabilities (and mainly uses human language only, which requires semantic parsing), and existing tools such as Nmap [10] only have probing scripts for a very limited set of these vulnerabilities. Thus the amount of efforts to write scripts to test the vulnerabilities would be prohibitive given the large number of vulnerabilities (2686 out of 75274 are relevant to APs based on our investigation). Second, in contrast to the scanning tool for laptops/smart phones where the tool can run *on* the device being scanned, we can only *probe* an AP from *outside*. To maximize the usability of the tool, the probing tool is better run as a mobile App on user's smartphone. Therefore scanning one AP should be quick enough before users lose patience. Due to these practical constraints, we cannot afford to probe all the vulnerabilities even if we had all the scripts.

For the purpose of raising user awareness of AP security, and based on aforementioned practical constraints, our problem statement is to **build a mobile App that can probe the AP in use to obtain its vulnerability score in a fast and accurate way**. The key challenge is to efficiently balance the

* Dan Pei is the corresponding author.

tradeoff between the probing speed and scoring accuracy. For example, if the App uses Nmap [10] to probe all potential vulnerabilities, the result will be accurate but the time it takes will be too long. Therefore, in this paper, we will focus on an AP's *potential vulnerabilities* without actually verifying whether these vulnerabilities are actually penetrable.

The core idea for tradeoff between speed and accuracy is based on a couple of assumptions. **Assumption 1:** *if there is a match of an open port/service and a vulnerability in CVE database, we consider that this AP has the corresponding vulnerability*. For example, if our probing shows that port 80 is open and the web server is Apache httpd 2.2.3 on an AP, then the CVE scores of all vulnerabilities for Apache httpd 2.2.3 will be added to the overall vulnerability score of the AP. With this assumption, we just need to probe an AP's open port/service without testing the actual vulnerability. **Assumption 2:** *Majority of APs in the wild typically do not update their port and service configurations after the initial deployment*. This means the majority of APs of the same vendor/model in the wild have the same (factory default) service/port configurations. Thus we can simply probe an AP's model instead of probing its service/software. In other words, the model is considered as an estimator of the AP's open ports/services to further calculate the scores. In order to test and verify our second assumption, we made a survey about some AP manufacturers. The results show that lack of updates is common for APs deployed in residential homes and small businesses, especially for *non-smart* APs. We also conducted a simple experiment on our testbed to check the effects of the firmware updates. More details are presented in §**??**.

Above approach has a very fast probing speed, thus the remaining challenge boils down to building an accurate table, which maps AP models to corresponding vulnerability scores. To bootstrap this mapping table, we first build an initial AP Knowledge Base (AKB) which contains the mapping from AP models to open ports/services, based on a testbed consisting of different AP models in our lab. We then crawl the CVE database for all vulnerabilities related to APs and their individual scores, and build a Vulnerability Knowledge Base (VKB) which contains the mapping from a service to its vulnerability score, which is no-trivial since a vulnerability in CVE is for a range of software version (e.g. Apache httpd up to version 2.2.3). Joining these two tables, we get an initial mapping table from AP models to their scores, and use this table to seed the mobile App for volunteers to use. Any new models detected in the wild that do not exist in the AKB will be probed using Nmap by the App, and then the probing results are sent to a server to update the AKB. AKB's accuracy is actively maintained: the App randomly samples an AP's ports/service (via Nmap) when the AP model does match the AKB, and any new discovery is incorporated into the AKB. The latest AKB are periodically updated to the App.

The mobile App, called SAVY (*S*core of *A*P *V*ulnerabilit*Y*), is built with 2000 lines of code, and volunteers have measured more than 800 APs from different public areas mainly in Beijing, China. To the best of our knowledge, this work is the first measurement study on public AP vulnerability. Our results show that we could tell similarities and disparities between APs based on vendors, types and stored they are deployed. Our results also show that SAVY reduces the probing time by 93.7% compared to full Nmap probing, and achieves scoring accuracy of 90.5% based on our knowledge bases.

The contributions of the paper are summarized as follows:

- For the first time in the literature, we highlight the prevalence of AP vulnerabilities in the wild, and the urgent need of a quantitative evaluation of AP vulnerability by users.
- We propose a fast and accurate approach for measuring AP vulnerabilities via our mobile App SAVY. As a crowdsourcing tool, SAVY's speed and accuracy evolves for the better as more and more users use it.
- We present that AP vulnerability is prevalent in the wild, and the scores vary across vendors, models, SSIDs.
- Our results show disparities between APs based on their types: (1) APs managed by commercial AP operator (such as WiWide), are usually more vulnerable than other APs due to more open ports and services. As a result, APs at some chain stores (STARBUCKS, JACK & JONES, Gap Stores, Ochirly, *etc.*, often managed by commercial AP operators such as WiWide and Zhima tech) are more vulnerable than those managed by the store staff; (2) Compared to traditional APs, the smart APs have higher vulnerability score because they open more ports to run more services on the AP.

The rest of the paper is organized as follows. §II reviews the overall architecture of our proposed system. §III and §IV present the details and the building of the AP Information Knowledge Base and the Vulnerability Knowledge Base, respectively. §V presents the detailed design and implementation of the App. §VI shows the real world experiments and the results. §VII concludes the work of the paper.

II. System Overview

In this section, we present the overview of our system architecture. First, we present the design goal of the system. Then we explain the architecture and work flow of the system.

A. Consequences of Compromised AP

With the wide usage of the wireless devices, people come cross with wireless network almost everywhere in their daily lives. The popular usage of wireless devices raise the risk of being compromised. Worse still, the ease of accessing the public APs makes them more vulnerable.

Once corrupted, the APs could be taken as part of a botnet [11], used to further infect other devices via LAN, or become the launchpad of some specific attacks. Although many methods can be used for protecting users' information, such as *https* [12], [13] and *VPN* [14], they cannot cover all the situations. Besides, the lack of awareness of the problem leads to the fact that most users, especially ones with little knowledge of IT, will not initially pay attention to the security issues unless they are specifically asked to.

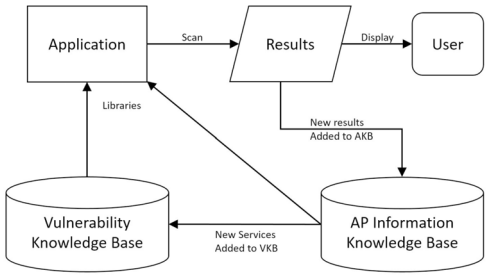


Fig. 1. The feedback of the overall system.

Attackers who take control of the APs could launch all kinds of attacks from the compromised APs to steal sensitive information from users. For example, by DNS redirecting, the attackers can reroute the user's legitimate requests to attackers' own hosts, conducting a phishing attacks via WiFi [15]. Such an attack is relatively easy to launch on a compromised public AP, but can attack many users given there are often many users for a public AP.

B. Design Goal

For the purpose of raising user awareness of AP security, and based on aforementioned practical constraints in §introduction, our problem statement is to *build a mobile App that can probe the AP in use to obtain its vulnerability score in a fast and accurate way*.

C. System Architecture

The overall system can be divided into three parts: the **AP Information Knowledge Base (AKB)**, the **Vulnerability Knowledge Base (VKB)** and the **Application** (called **SAVY**). Fig. 1 shows the relationship of the three parts of the system: the App will run the vulnerability test and display the results to the users, while taking advantage of the two knowledge bases as the libraries. In addition, the new results will be processed and added to the AKB. Once the AKB gets the new model/services information, the new services information will be sent to the VKB as the keywords for related vulnerability entries updating.

The **AP Information Knowledge Base** can be considered as several relational databases of the model and the service information mapping. The entries of AKB get updated by the new service results detected by our App. Given a specific model, we are able to find out the firmware version that the device might be using, the versions of the services (*e.g.*, web servers, SAMBA.) running on the device. Such information will be used to help building the VKB.

The **Vulnerability Knowledge Base** can be considered as the mapping of the services and the relevant vulnerability scores. It is selected and merged from many other public vulnerabilities and exploitations databases, including the *Common Vulnerabilities & Exposures* (**CVE**). This knowledge base mainly serves as the library for vulnerabilities querying and scoring.

The **Application** will test the vulnerability of the APs in the wild, whilst providing brief results for users to check, further reconfigure and optimize. Given the assumption that many client-side devices lack in both computing resources, in most cases, we are unable to perform a comprehensive scanning against the AP. Our idea is based on assumption 2 (majority of the APs in the wild typically do not update their APs upgraded), evidenced by previous studies [16] (especially *non-smart* APs which do not perform auto-update). Furthermore, most of the update packs are provided by the vendor, which do not have major changes in the OS and services. Therefore, we are able to give a rough estimate of the vulnerabilities an AP may have via the AKB and the VKB, which results in saving both time and computing resources. Furthermore, the probing results are sent to the server to update the AKB and VKB.

D. Application Workflow

Fig. 2 shows the overview flowchart of our application. To generate a score of the vulnerability level of a certain AP, the system will take the following steps (the step numbers correspond to the circled numbers in Fig. 2):

1) **Model Probing**: The application will try to detect the model of the AP.
2) **Searching AKB**: When having the model probing result, the application will search AKB for a match in order to get the service information. If no match is found in AKB or no model results are available, then the application will start scanning to detect such information.
3) **Searching VKB**: When having the services information, *etc.*, the application will select the relevant potential vulnerabilities based on the VKB.
4) **Scoring Metrics**: Based on the VKB and the service version, the application will score the vulnerability level of the AP.

III. AP Information Knowledge Base

During the study, we present the assumption that many APs do not get upgraded during their lifetimes [16]. In other words, the results from the same AP model are likely to be the same. If we can get the knowledge of one certain AP's model, we are highly likely to know the service version and other information. Therefore, the AP models which were scanned before can be gathered and merged into a knowledge base, *a.k.a*, the AP information Knowledge Base. The AKB provides the mapping between the AP model types and the relevant service information, and is built and updated by the App results. At the very beginning, we build a small testbed consisting of 26 APs (based on ranking websites and the sales volume data from the top online shopping websites) and use the results to build the initial knowledge base. With the help of the knowledge base, we are able to give a rough estimate of the vulnerability level just by knowing the certain AP's specific model type. In this section, we will describe the building of the AKB.

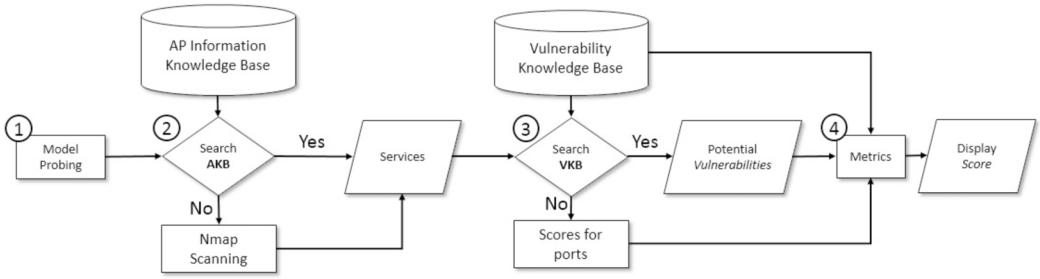


Fig. 2. Architecture of the application.

A. Building the Testbed

At the very beginning of our study, we build a small testbed consisting of 26 APs. The picking of the models are mainly based on their market shares and popularity. Based on the ranking website and the sales volume data from the top online shopping website, we list the most widely used models of the residential wireless routers and the most popular vendors in China. In order to have a more comprehensive study of the different models, we ignored several similar models from the same vendor and chose some of the most popular routers from the same vendor.

In general, the testbed mainly helps us in these fields:

- The testbed results are used to build the initial AKB (and also the initial VKB). The two knowledge bases will help when running the App in the wild.
- The APs in the testbed can update to different versions of the firmware, in order to test our *Assumption 2*.
- The testbed results can be considered as an evaluation for the App results. Since we have the knowledge of the model types in the testbed, we can test and verify the method of model probing.

We build a small testbed in our lab, consisting of popular APs from different vendors. All these APs are connected to a switch via LAN ports, with their DHCP service shut down. In order to get access to them, these APs are divided into different subnetworks. Any client connected to the switch can easily get access to a certain AP using static IP settings. A server is connected to the switch as the main controller of the whole testbed during the study.

B. Update of APs

As mentioned in §I, we made an assumption that APs do not get updated frequently. The details of the assumption are reiterated here: *Majority of APs in the wild typically do not update their port and service configurations after the initial deployment*. This assumption has two meanings. First, majority of APs do not update their softwares in their lifetimes. Second, even some APs do update the softwares, such updating do not change the characteristics of their port/service configurations. As a result, APs of the same vendor/model in the wild will have the same service and port configurations. A detailed scanning of the open port/service is very time-consuming along with extreme heavy overhead, especially on the mobile devices. With this assumption, we can turn the port/service probing of one AP into a simple model probing. That is, we do not need to probe each AP in details. Instead, we just need to build a table that maps an AP's model to its vulnerability score; then in the wild, the mobile App can just use a very lightweight method to probe the AP's model, and then look up the table to find the AP's vulnerability score.

For most APs, especially residential ones, lack of updates is always the one of dominant reasons for AP security issues [17]. We found that 9 out of the current top 10 best-selling AP models on JD.com, one of China's largest online retailer, are non-smart APs which have no automatic update functions, ranked by the number of customers' comments, with Mercury MV309R taking the lead by sales volume of 540,000. It is worth noting that non-smart APs are still dominant and very popular in market and have an overwhelming sales due to wide range of optional brands and cost performance compared to smart APs. Even though some smart APs have auto-update function, the firmware update usually can not be finished without user's manual confirmation. Most smart APs only prompt update notification for users to check, rather than fulfilling update in a completely unattended way. If APs work normally without any failures, the issue of firmware update may be easily ignored by users, which leads to the common phenomenon of non-updated firmware of APs, either non-smart or smart.

We also conducted some experiments on our testbed to verify our assumption. With some exceptions, the APs on our testbed are all non-smart APs, which do not present functions like auto-updating and third-party add-ons. Users of these APs need to manually operate the overall updating of their devices. Most manufacturers will provide the update firmware files on their tech-support websites. One user needs first search the update files which exactly match his or her AP model, and then download them on the PC or laptop. Afterwards, he or she needs to login the configuration page of the AP, upload the files to the device and start updating. Finally, the AP will reboot automatically to finish the updating.

From the release dates of the firmware files, we can get a rough idea of the updating of APs. Table **??** shows the dates of the firmware updates for 6 AP models with the same brand *TP-LINK*. The dates of the files could be identified by the build version numbers of the firmwares and the website. Most models do not provide their original version of firmware on tech-support websites. The base dates (original firmware dates)

are estimated by the model release dates and the documents. From the table we could tell that most TP-LINK APs on our testbed may have two updates in their lifetime, the time gaps are always 1-4 months or longer. The results for other APs are similar to the TP-LINK APs, which usually have 1-3 firmware updates. And the time gaps for different updates are several months or even longer. For example, the model CVR100W (Cisco) have three firmware versions on its tech-support website, and the release dates are 20 May 2013, 31 Dec 2013 and 26 Jan 2017, respectively.

Besides, the firmware updates are usually for bug fixing and performance optimization, and do little change to the software and port configurations. We ran several scanning tests against different firmware versions for the same model on our testbed. And the results show no differences between the ports, services and OS. An example is shown in Fig. **??**, the two firmwares are released in May 2014 and July 2015 respectively, with 14 months in between. And the scanning results are almost the same except for the *Slowloris DOS Attack* (which is only labeled as **LIKELY VULNERABLE**).

```
PORT      STATE SERVICE   VERSION
80/tcp    open  http      TP-LINK WDR6300 WAP http config
1041/tcp  open  danf-ak2?
1900/tcp  open  upnp      ipOS upnpd (TP-LINK TL-WDR6300 WAP 1.0;
UPnP 1.0)
49152/tcp open  upnp
49153/tcp open  upnp

Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.23 - 2.6.38
Network Distance: 1 hop
Service Info: OS: ipOS 7.0; Device: WAP; CPE: cpe:/h:tp-
link:wdr6300, cpe:/h:tp-link:tl-wdr6300, cpe:/o:ubicom:ipos:7.0
```

(a) Build 140331

```
PORT      STATE SERVICE   VERSION
80/tcp    open  http      TP-LINK WDR6300 WAP http config
| http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2007-6750
|       Slowloris tries to keep many connections to the target web
server open and hold
|       them open as long as possible.  It accomplishes this by
opening connections to
|       the target web server and sending a partial request. By
doing so, it starves
|       the http server's resources causing Denial Of Service.
|
|     Disclosure date: 2009-09-17
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?
name=CVE-2007-6750
|_      http://ha.ckers.org/slowloris/
1041/tcp  open  danf-ak2?
1900/tcp  open  upnp      ipOS upnpd (TP-LINK TL-WDR6300 WAP 1.0;
UPnP 1.0)
49152/tcp open  upnp
49153/tcp open  upnp

Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.23 - 2.6.38
Network Distance: 1 hop
Service Info: OS: ipOS 7.0; Device: WAP; CPE: cpe:/h:tp-
link:wdr6300, cpe:/h:tp-link:tl-wdr6300, cpe:/o:ubicom:ipos:7.0
```

(b) Build 150513

Fig. 3. The Nmap Scanning Results for AP Model TL-WDR6300.

C. Probing the AP Model

The **AP information Knowledge Base** maps the model and the relevant service information. The AP model is used as the keyword to search the AKB. In order to take advantage of the knowledge base and get the information of the detailed services, we need to fetch the AP model type first. A simple approach is to use the login page as the *HTML fingerprint* of the AP. Many APs just present their model types directly on the login page. This webpage can be easily accessed without any authentication. Anyone who is able to connect to the AP can visit the webpage and the model type is often in the *<title></title>* field.

Although we could get detailed model information from some certain APs, there are still some that cannot be detected by this simple method. We conducted small scale real world experiments for several times to verify the validity of this method. During the experiments, we tested over 100 APs in the wild using a simple smartphone App. And over half of the APs' model types were detected via the *<title></title>* field. We also found that several APs put the name of the vendor instead of the model in that field. In such situation, although we did not detect the specific model type, but we could narrow down the range of the possible types. Further, since we have downloaded the overall page, we are free to take advantage of the remaining parts. And we can still find regularity of the *HTML* contents based on the invariant contents related to the vendors/models.

D. Building AP Information Knowledge Base

The AKB presents the related service information when querying one AP model type. In specific, the service information consists of the open ports of the AP and the associated software/service versions. During our study, we used the open source NMAP tool as our main scanning toolkit. With the help of the scan tool, we are able to get the information of the open ports and the services (application name and version) offered by the hosts.

When get both the AP model types and the service information, we are able to build the AKB. The AKB provides us with the mapping between the AP model type and the services.

For each entry of the certain AP model type, the knowledge base will list the open ports and the associated services. An example of the knowledge base is shown in Table **??**. Noticed that we only list the web server here. In the actual knowledge base, all the services for every open port will be listed. Besides, in the real world test, the scanning results for the same AP model may have slight differences, so the AKB also lists the number of the APs detected for a certain model-ports/services pair, which will be presented in §**??**.

The results show that the most opened port of the APs is #80, which is intuitive since almost all the APs provide the web login and configuration pages. The web server versions of the APs are different and are mainly based on the vendors. Most of the vendors use one specific web server for a series of productions. And different vendors may use the same popular third-party web server, *e.g.*, *GoAHead*, *Apache* and *Nginx*.

Besides the #80 port, some APs also open some other ports for different usage, including #53, #443 and #1900. And few APs open the #23 port for *Telnet* connection. This is extremely risky since the *Telnet* username/password are quite simple (*admin/admin* in most cases).

TABLE I
RELEASE DATES OF AP UPDATING FIRMWARE FILES.

#	Model	Base	#1 Update	#2 Update
1	TL-WDR6300	20140317	20140520	20150720
2	TL-WDR7400	20150909	20160612	20160914
3	TL-WR742N	20150330	20150720	20151104
4	TL-WR842N	20150906	20160125	20160505
5	TL-WR885N	20140625	20140725	20140814
6	TL-WR886N	20141117	20150407	N/A

TABLE II
AP INFORMATION KNOWLEDGE BASE.

#	Open Ports	Web Server
1	80, 1041, 1900, 49152, 49153	TP-LINK http config
2	80	GoAhead WebServer
3	80	GoAhead WebServer
4	23, 80	Apache Httpd
5	80, 81, 1723	Apache Httpd
6	80	Boa HTTPd 0.94.14rc21
7	80	Nginx 1.5.5
8	80	Nginx
9	53, 80, 81, 82, 83, 139, 443, 445	QWS
10	53, 80, 81, 82, 83, 139, 443, 445	QWS

E. Updating AP Information Knowledge Base

As described in §II-C, the AKB is updated by the new results collected from SAVY. Once a new model is detected, the results will be sent to the server to update the AKB. Besides, in order to get more accuracy information for one certain AP model, SAVY will trigger the port scanning by randomly sampling even the AP model type has been recorded.

For the APs with the same model type, multiple results will be generated and sent to the server from different users over time. Most of these results would be the same (more detailed results will be presented in §VI-C), but there would be results with different open ports status. Each of the open ports status will be recorded and the dominant one will be the default result for the AKB querying. On the other hand, the services probing results for the same port of the same model are always invariant. During our study, the results for service of the same port for the same AP model will be in two status: one certain service or nothing detected. Based on the assumption that users rarely update their APs, we present a hypothesis that the services for the same ports of one very model should be the same. In such condition, the AKB will merge the results to form a more detailed port-service mapping. In case of a conflict, the AKB will take the dominant results like we do to the different open ports status.

IV. VULNERABILITY KNOWLEDGE BASE

Based on several public databases of vulnerabilities and exploitations, we build a specific knowledge base, which lists the vulnerabilities and the corresponding vulnerability scores.

A. Building Vulnerability Knowledge Base

We collected vulnerabilities for APs from two major sources: i) NMAP and its default vulnerability check scripts; ii) the *Common Vulnerabilities & Exposures* (CVE) list.

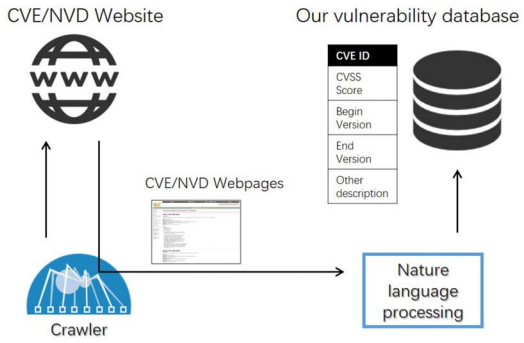


Fig. 4. Building the vulnerability database.

There are several kinds of vulnerability databases provided by different websites and organizations, among which CVE/NVD covers vulnerabilities both early and up-to-date, being one of the most frequently referenced and commonly used vulnerability database. CVE/NVD provides a database of information security vulnerabilities and exposures with additional analysis and a fine-grained search engine, containing 75274 CVE vulnerabilities by the time of this study. Further more, The NVD uses the *Common Vulnerability Scoring System* (**CVSS** [18]) Version 2, which is an open standard for assigning vulnerability impact scores that is widely used. Thus, we build our vulnerability database based on CVE/NVD.

Service related vulnerabilities are the primary part of our knowledge base. To collect such vulnerabilities, we build a simple python web crawler to collect relevant vulnerabilities from NVD, using regular expression to parse the information, as shown in Fig. 3. The crawler grabs all the related vulnerabilities from NVD, obtaining the CVE-IDs, then extracting the CVSS base scores and sub scores, description and affected versions of each CVE-ID (ranges from begin version to end version).

B. Scoring Version Segments

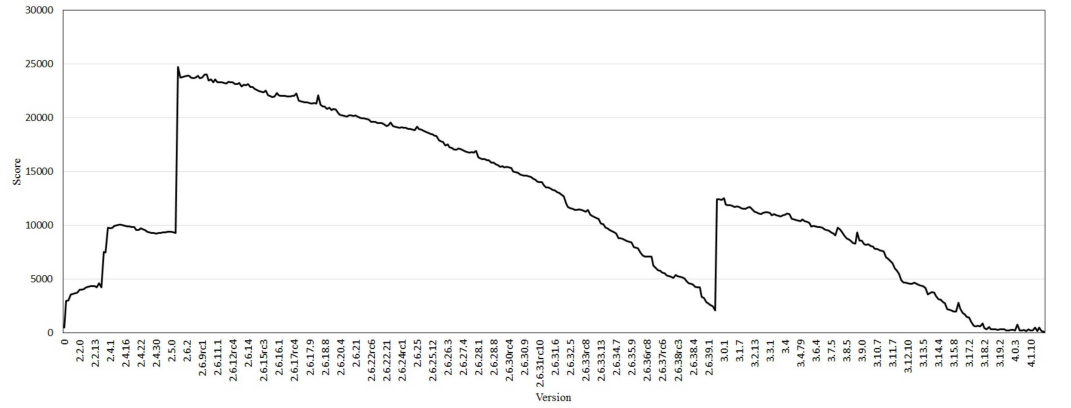
Based on the AKB and the VKB, we can obtain the knowledge of a specific AP's services and these services' relevant vulnerabilities. Here, we present the method to score one certain service by its software version.

In general, the scoring metrics should follow these two characteristics:

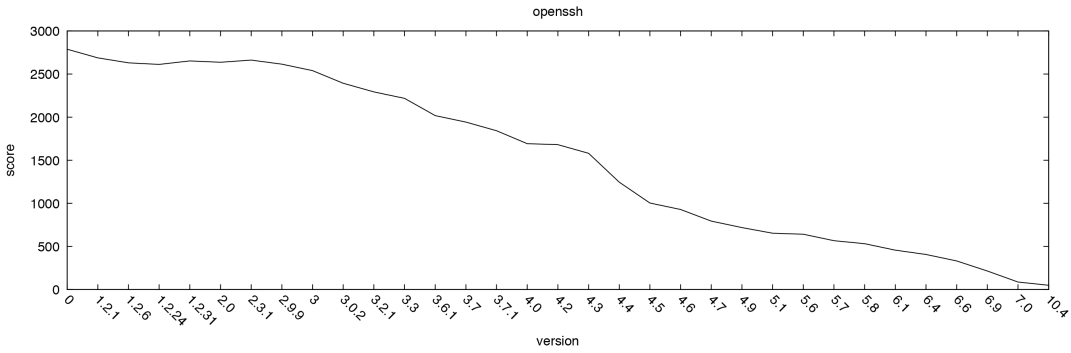
- The more potential vulnerabilities the AP may have, the higher the score will be.
- The more severe the vulnerabilities are, the higher the score will be.

The CVE list represents a scoring metrics for individual vulnerability named CVSS. To achieve our metrics, a simple approach is to sum up the individual scores of different potential vulnerabilities found on a certain AP. For example, an AP with three vulnerabilities was found, while the CVSS scores are 5, 5 and 7, respectively. Then the score of the AP will be 17.

In fact, many vulnerabilities are not specific to one certain version, but affect several continuous versions. Based on this,



(a) Linux



(b) Openssh

Fig. 5. Version segment scoring for different services. The X axis represents the versions of the services. The Y axis represents the scores calculated by the metrics. An unknown version can be given a score due to the metrics by querying the certain range of X.

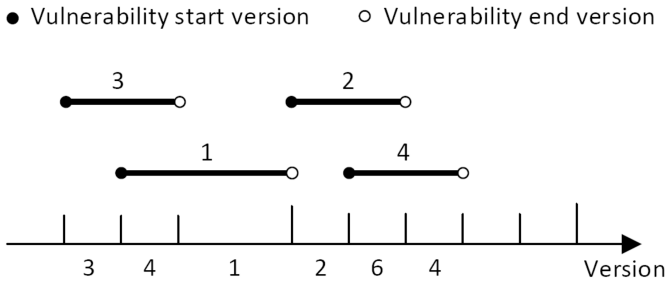


Fig. 6. An example of the version segments. Each interval on the axis represent one single version. In this example, the axis is divided into six version segments with the score of 3, 4, 1, 2, 6 and 4 respectively.

we could divide the range of the version into several individual parts (called version segments during our study) and score the different parts respectively. This is similar to the Median Active Vulnerabilities [19] and its variants [20], but uses the scores instead of the count of their quantities.

An example of the version segments is shown in Fig. 5. Each interval above the axis represents the affected versions of one certain vulnerability, the number over the interval represents the score of the vulnerability. In order to get non-overlapping parts, all the vulnerability ranges are left-closed. All the endpoints divide the axis into several intervals, *a.k.a*, the version segments. We score the segments by accumulating the scores of the relevant vulnerabilities which overlaps with the certain version segment.

Some examples of relationship between the version and vulnerability score are plotted in Fig. 4, based on our VKB results. The values varies over time with a trend of getting smaller after patching. Fig. 4(b) shows the scores of the *openssh*, it is typically getting less and less vulnerable while being updated over time. A more complex example is shown in Fig. 4(a) (the scores for different *Linux* versions). In this figure, we can see that each primary change accompanied with new features introduces a mass of vulnerabilities (*Linux 2.6* and *Linux 3.0*, respectively). Also the number of vulnerabilities taper off with the release of new minor and patch versions.

As presented above, given a certain service version, we can search the VKB to find out the relevant version segment. Then we can give the score without recalculating. In case of an uncertain version, we can get the potential version range, which may contain several version segments with different scores, and use the average value as the final score.

V. Application Implementation

After designing the architecture of the whole system, now we will take a look into the detailed steps. In this section, we will present the implementation of the application: model probing, searching AKB, searching VKB and scoring.

A. Model Probing

As mentioned in §II-D, the first step in Fig. 2 is the model probing. In order to take advantage of the AKB, we need to detect the model of the targeted AP. Once successful, we can bypass the time-consuming scanning procedure, list the potential vulnerabilities and go directly to the scoring step.

The procedure of the model probing is the same as described in §**??**. During our real world experiments, over 50% APs in the wild had *<title></title>* field with certain information. Although not all these fields put the model name right here, most of them are recognizable. A WiFi AP deployed by a commercial WiFi company at a store almost always put the name of the store here, but they do not change the other part of the webpage, which can still be used to detect the vendor and model. For example, the invariant part of such a web page often contains the links to its official website or some logo pictures, and the elements in the page can be used to determine the model type by comparing with others. If all these methods fail, *e.g.*, the HTTP connection is rejected, we can get the vendor instead of the model by querying the *OUI*. The vendor information is very limited but could also help to estimate the service/port information and speed up the scanning phase by specifying scanning parameters.

B. Searching AP Information Knowledge Base

After the model probing, the application will look up the AKB for further information. If we could find the results in the AKB, then we can get the service information.

When the results fail to perform a valid match (or in the case that the model probing failed), we cannot get the score directly. In such a case, we need to use the scanning toolkit embedded in the application as the backup. Such procedures are always time-consuming compared to the table-searching and simple calculating. Meanwhile, the new model type (if there is any) will be sent to the server along with the port/service information to update the AKB.

C. Searching Vulnerability Knowledge Base

When having the knowledge of the service information, the application will then look up the VKB in order to get the scores. If we could find the results in the VKB, then we can get the scores directly.

When the searching fails in the VKB (which is highly likely after the AKB searching failure, since a known AP model always has the same services), the new service information will be sent to the server to update the AKB and the VKB. For scoring, the application will use the port number instead for a rough scoring based on the average score for the port.

D. Scoring the Vulnerability Level

As presented in §IV-B, we can get a score for one certain service. All the scores for different services make up with the overall vulnerability score of the AP. In the App, the scores will be accumulated together. In order to make a clear display for the users, we normalize the scores based on the former results collected from the App. That is, instead of the absolute score, we display the percentile among all the results already known. In other words, if the final percentile is 0%, it means that the AP has the lowest score, and a percentile of 100% means the most vulnerable AP in our dataset. Just like the AKB and the VKB, the percentile would change over time since the dataset is updated when more results are collected via the mobile App.

VI. Results in the Wild

We conducted a field experiment to measure the APs in the real world using our application. In this section, we will first describe the details of the experiment. Then, we will present the results and the analysis of the collected data.

A. Measurements in the Wild

With the App SAVY, we are able to measure the APs and collect the data in the real world. To evaluate the performance of our application, and to understand the vulnerabilities of the APs in real public/residential environments, we installed our application on volunteers' android devices. The volunteers will probe the public APs they encountered while going out or staying home. In order to get rid of the effects from the non-public APs, we filtered the hosts and only considered the APs when the clients get a valid LAN IP address.

The volunteers were sent to places with different densities of public APs from low to high around the city. **Malls** and **plazas** are always crowded with different public APs. These places have lots of individual stores and cafeterias, which often provide their own WiFi for customers. Measurement results from such places are variant based on vendors, types, and locations. Moreover, the APs within the same chain stores showed strong similarities, while with different owners came the disparities. For instance, we found that many stores' WiFi devices are deployed by certain commercial companies. **Office buildings** and **campus** also provide public WiFi for their employees and visitors. Such APs are enterprise APs with centralized controllers and specialized operators. Most of these *thin APs* do not provide NAT and/or DHCP services. In such cases, we are not able to probe the APs themselves since all the scan packets will go through the access points and directly to the routers or gateways. **Other public places** (*e.g.*, railway stations, air ports) also provide the wireless networks but with more strict limitation for authentication.

We also ran the measurements in residential scenarios. Unlike the public APs, the residential WiFi APs are always protected by passwords and are hard to access. As a result, we did not get large number of the measurement results from residential environments, but the workflow of the application still work in such conditions.

B. Results Overview

We have collected over 800 results from the APs in the public places over this period. During the real world measurements, we also found that using the full Nmap scanning will takes 3 minutes or longer. Such long time may cause the users more likely to abandon the scanning. With the

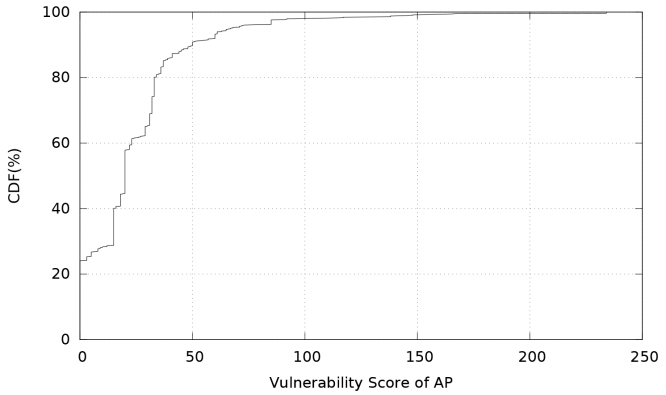


Fig. 7. CDF for scores of all tested APs. X axis for the score and Y axis for the percentage.

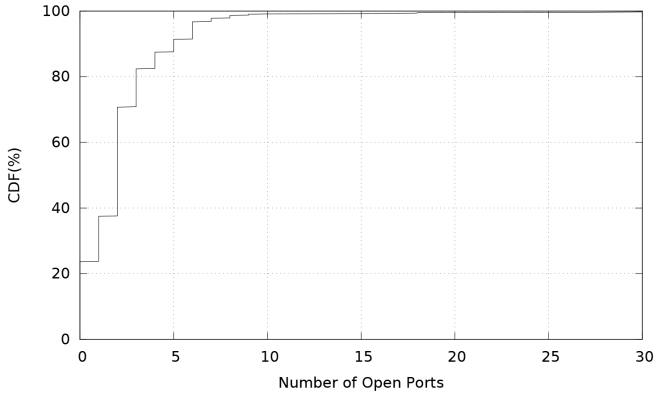


Fig. 8. CDF for the numbers of the open ports. X axis for the #ports and Y axis for the percentage.

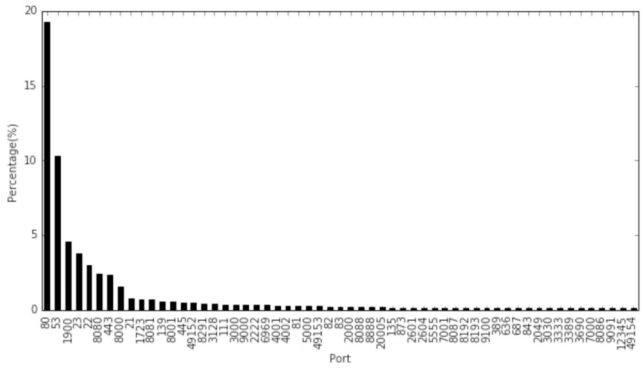


Fig. 9. Open ports results in the wild. X axis for the port number and Y axis for the percentage.

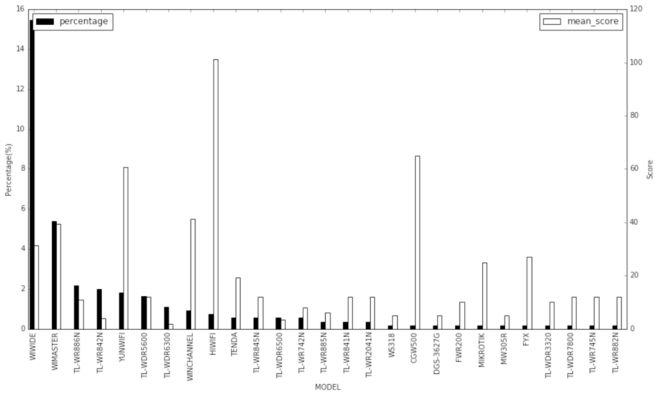


Fig. 10. Results in the wild. X axis for the model and Y axis for the percentage and the scores.

help of the AKB and VKB, the results could be displayed after model probing in 10 to 20 seconds. In addition, we record the timestamps of each measurement step in Fig. 2 using the SAVY Mobile App. The timestamps of our 800 measurement results show that SAVY reduces the probing time by 93.7% compared to full Nmap probing on average, which is a significant improvement on the probing speed. Besides, we also make some optimizations for the Nmap scanning based on our results (*e.g.*, specific ports range) to reduce the scanning time to about 30 seconds when Nmap is called.

The scores for all the measured hosts are shown in Fig. 6. Almost 40% of the hosts' scores are smaller than 20, over 80% of them are smaller than 40, and nearly 95% are under 100. These data also help in the normalization of the scores when displaying to the users (in §V-D). For example, in order to make it more clear to understand for the users, based on these data, we can divide the APs into three categories by the scores: **safe** when the score is under 20, **vulnerable** when over 100, and **medium** when in between.

Fig. 7 shows the numbers of open ports of all the APs we measured. 80% of the APs open 4 ports or less. The results for the open ports is shown in Fig. 8. Port #80 is the most common port of the APs, indicating that the web server is the dominant service. Besides the #80 port, the most common ports are #53, #1900, #23, #22, #8080, #443 and #8000. Except for the HTTP/HTTPS ports, the other popular services are DNS, Telnet, UPNP and SSH. Such ports and services often indicate more vulnerabilities.

Fig. 9 shows the numbers of the different AP model types based on our measurements, along with the mean scores detected by the App. A certain model WIWIDE dominates with 16% of the overall measured hosts, followed by the second dominant model WIMASTER. Both models are from a same commercial WiFi company *Wiwide* which provides WiFi service such as commercial WiFi deployments and technical support. Many models from the vendor *TP-LINK* also account for great proportion of the overall data. Unlike *Wiwide*, the TP-LINK APs are not deployed or operated by the manufacturer. As a result, the TP-LINK APs open less ports and the vulnerability scores are low since their services are just the factory default (minimum) set of ports. On the other hand, the Wiwide APs open more ports (*e.g.*, #22, #23) for more functions. The AP model with the highest score is HIWIFI, which is a *smart* AP model with many open ports. This implies that more functions (of smart APs) might come with more security vulnerabilities if not coped with carefully.

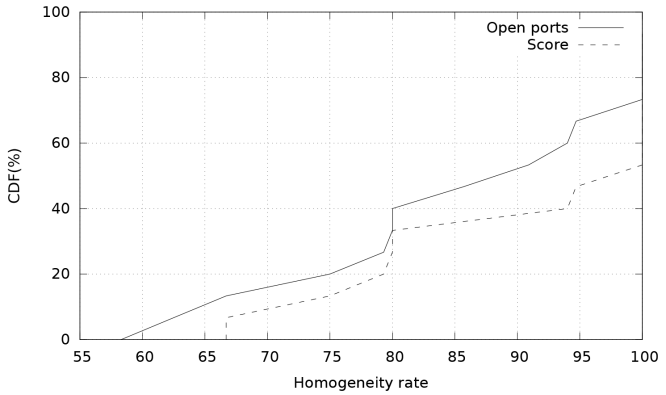


Fig. 11. CDF for the homogeneity. The homogeneity represent that how many APs detected by the NMAP share the same status of ports/score with the AKB result.

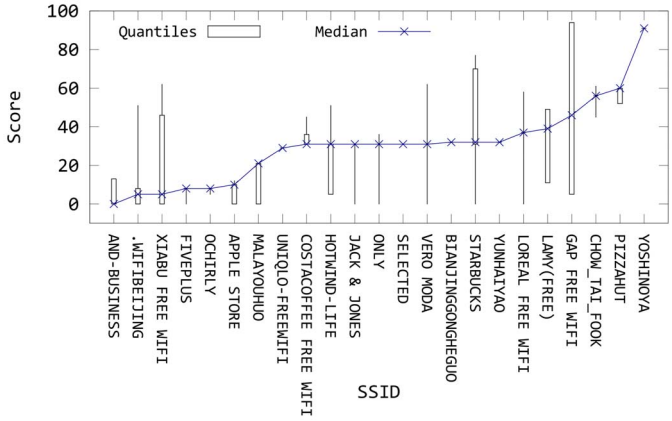


Fig. 12. Scores distribution for each SSID.

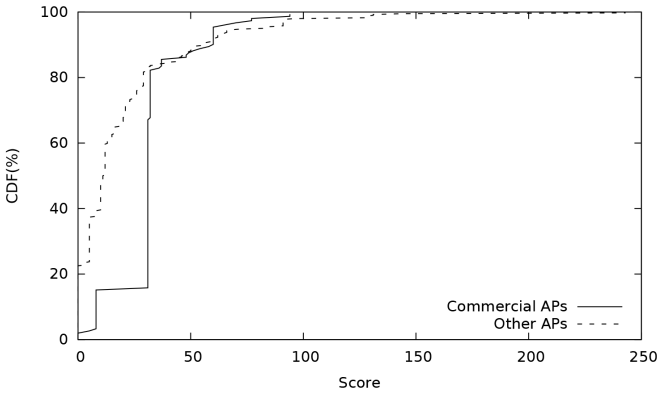


Fig. 13. CDF of scores for different AP categories. X axis for the score and Y axis for the percentage. Red line for the APs deployed by WiFi service vendor. Black for self-deployed APs.

C. Accuracy of the AKB

As mentioned before, we simply probe an AP's model as an estimator of the vulnerability scores based on assumption 2. Then the AKB provides the mapping from the model types to the services/ports. To further verify this in the wild, we set the service scan of the App in the **compulsive mode** during the volunteer measurements. That is, the App would start scanning for service information regardless of whether the AKB return a valid result or not. This way, we were able to measure the APs with the same model type.

The AKB presents a standard scanning result for one certain model type. For several APs with the same model type, we compared the results of the actual scanning. Here we use homogeneity to denote the percentage of the # of AP with same status as the AKB results, *e.g.*, a homogeneity of 100% for open ports indicates that all the APs have the same open ports detected. The homogeneity of the open ports/scores is shown in Fig. 10. We have filtered the model with less that 4 APs in the dataset. The result shows 60% of the model have a homogeneity rate over 80%, 40% of the model have a homogeneity rate over 90%. Most model types from the real world results show high homogeneities, which means that most APs with same model types have the same services/ports characteristics. Some model types with lower values are mainly because of two reasons: i) scanning errors of the scanning toolkit; ii) uncertain model probing results, which AKB will provide a coarse model type category.

With the NMAP results we can calculate the scores. Compared to the ones supported by the AKB, we can tell whether the AKB scores are accurate. Fig. 10 also shows the homogeneity of the score for the same AP model type. Most of them have a better result than the port homogeneity, which indicates a higher similarity. This is because the differences of the open ports status are always related to the non-reserved port numbers, which have little effect to the final score.

D. Score Variations

1) Similarity in SSID: During the test, we also found that for one certain brand and its chains, the choices for the vendors and the models have a strong homogeneity. The stores under the control of the same company are more likely to choose the same AP vendor, especially the APs deployed by the WiFi service providers. Fig. 11 shows the scores for different SSIDs in candlestick chart. The vertical segments denote the ranges of the scores for one SSID. The two endpoints of each segment denote the maximin and the minimum value, while the cross denotes the median value. It is clear to see that many SSIDs have concentrated distributions. The exceptions of them are related to the choices of the models. Several chain stores, *e.g.*, Starbucks and GAP, often choose the same vendor for WiFi APs, but some vendors may offer different model types, which results in the variations of scores.

2) Disparity between AP Vendors: Fig. 12 indicates that APs managed by commercial AP operator (such as WiWide and Zhima Tech) are usually more vulnerable than self-deployed APs, for them often open more ports (*e.g.*, port #22 and #23) for certain usage such as SSH and Telnet. The difference in results between commercial and non-commercial APs is related to the results in Fig. 9, which shows the popularities and scores in our dataset for different vendors. In particular, the dominant non-commercial AP is TP-LINK

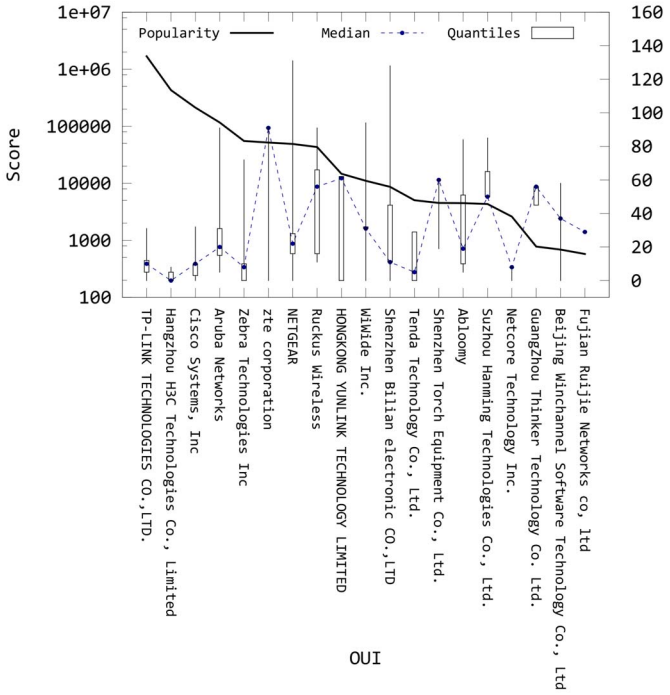


Fig. 14. Scores of different vendors. Each number on X axis denotes each vendor and Y axis for the range of the scores.

with lower scores comparing to the others. Fig. 13 shows the scores for different vendors in candlestick chart. The vendors are sorted by its popularity in reverse order. The popularity of AP vendors is derived from a separate dataset, which has the 4.1 million AP logs in China, including the SSID, the BSSID, and the geographic coordinate of APs. In Fig. 13, the vertical segments denote the ranges of the scores for one vendor. And the two endpoints of each segment denote the maximum and the minimum value, while the dot denotes the median value. The scores vary between different vendors, and it is clear to see that popular AP models (*e.g.*, TP-Link, H3C, and Cisco) are less vulnerable than the others.

VII. Conclusion

In this paper, we present a mobile App, SAVY, the first in its kind, to measure an AP's potential security vulnerability in the wild. SAVY achieves a good tradeoff between probing speed and accuracy. Compared to the alternative approach, SAVY reduces the probing time by 93.7%, and achieves a scoring accuracy of 90.5%. We also present the first measurement results on AP vulnerabilities in the wild using SAVY. The observed prevalence of non-trivial AP vulnerabilities highlights the urgency of better understanding and improvement of AP security. The variations of vulnerability scores indicate different default practices by vendors, chain store owners, and commercial AP operators *etc.*, and call for a systematic solution by the industry to improve AP security in the wild.

As our future work, we plan to release the SAVY App to Google Play and other Android App stores to increase the user base, raise users' awareness of AP security, and gain a more comprehensive understanding of the AP vulnerabilities. We also plan to extend our App's functions like checking if the AP is compromised or not and giving more detailed advices.

Acknowledgments

We thank the valuable reviews given by those anonymous reviewers. We are also thankful for those nameless shopkeepers who offered help during our real world test in different public areas. Our study has been supported by the National Key Basic Research Program of China (973 program) (Grant No.2013CB329105), National Natural Science Foundations of China (NSFC) under Grant 61472210 & 61472214, the Sate Key Program of NSFC under Grant 61233007, the Tsinghua National Laboratory for Information Science and Technology key projects, the Global Talent Recruitment (Youth) Program and the Cross-disciplinary Collaborative Teams Program for Science, Technology & Innovation of Chinese Academy of Sciences-Network and system technologies for security monitoring and information interaction in smart grid.

References

[1] "Common Vulnerabilities and Exposures List," https://cve.mitre.org/, accessed: 2016-01-31.
[2] "WiFi Master Key hits 800M users," http://www.mobileworldlive.com/apps/news-apps/wifi-master-key-hits-800m-users/, accessed: 2016-05-10.
[3] S. Abu-Nimeh and S. Nair, "Bypassing security toolbars and phishing filters via dns poisoning," in *Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE*. IEEE, 2008, pp. 1–6.
[4] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions." in *USENIX security*, 2003, pp. 15–28.
[5] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-E, A. Bhartia, and D. Aguayo, "Large-scale measurements of wireless network behavior," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. ACM, 2015, pp. 153–165.
[6] I. Cisco, "Cisco visual networking index: Forecast and methodology, 2013–2018," *CISCO White paper*, 2013.
[7] "The best antivirus software for Android," https://www.av-test.org/en/antivirus/mobile-devices/android/january-2016/, accessed: 2016-02-18.
[8] "The best antivirus software for windows," http://www.dnhys.com/, accessed: 2016-02-18.
[9] "National Vulnerability Database," https://nvd.nist.gov/, accessed: 2016-01-31.
[10] "Nmap," https://nmap.org/, accessed: 2016-01-31.
[11] Z. Y. e. a. Knysz M, Hu X, "Open wifi networks: Lethal weapons for botnets?" in *INFOCOM, 2012 Proceedings IEEE*, 2012, pp. 2631–2635.
[12] "HTTPS," https://en.wikipedia.org/wiki/HTTPS, accessed: 2016-01-21.
[13] R. M. Callegati F, Cerroni W, "Man-in-the-middle attack to the https protocol[p]," in *IEEE Security & Privacy, 2009, 7(1): 78-81*. IEEE, 2009, pp. 78–81.
[14] L. A. Seid H A, "Virtual private network: U.s. patent 5,768,271[p]. 1998-6-16," *Virtual private network*, 1998.
[15] W. D. Felt A P, "Phishing on mobile devices[m]," *Phishing on mobile devices*, 2011.
[16] E. Karamanos, "Investigation of home router security," 2010.
[17] "Turris Omnia," https://omnia.turris.cz/en/?winzoom=1, accessed: 2017-02-10.
[18] "Common Vulnerability Scoring System," https://www.first.org/cvss, accessed: 2016-01-31.
[19] J. L. Wright, "Software vulnerabilities: lifespans, metrics, and case study," Ph.D. dissertation, University of Idaho, 2014.
[20] D. R. Thomas, A. R. Beresford, and A. Rice, "Security metrics for the android ecosystem," in *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*. ACM, 2015, pp. 87–98.