2017 IEEE 37th International Conference on Distributed Computing Systems

City-Hunter: Hunting Smartphones in Urban Areas

Xuefeng Liu*†,Jiaqi Wen*
*Department of Computing
Hong Kong Polytechnic University
†School of Elec. Info. and Comm.
Huazhong Univ. of Sci. and Tech.
Email: {csxfliu,elliot.Wen}@gmail.com

Shaojie Tang
Naveen Jindal School of Management
The University of Texas at Dallas
U.S.A.
Email: Shaojie.Tang@utdallas.edu

Jiannong Cao, Jiaxing Shen
Department of Computing
Hong Kong Polytechnic University
Hong Kong
Email: {csjcao,csjxshen}@comp.polyu.edu.hk

***Abstract*—The security issue of public WiFi is gaining more and more concern. By listening to probe requests, an adversary can obtain the SSID list of the APs to which a smartphone previously connected, and utilizes this information to trick the smartphone into associating to it. However, with the enhancement of security level, most smartphones now do not proactively disclose their SSID lists, making these attacks obsolete. In this paper, we propose City-Hunter, an attacker that can lure nearby smartphones without knowing their SSID information. City-Hunter establishes and maintains an SSID database by integrating both offline and online information. Meanwhile, it smartly chooses some SSIDs to hit a smartphone according to the past record and freshness. We evaluate the performance of City-Hunter in different public places. The results demonstrate that City-Hunter is able to successfully hit $12\% \sim 18\%$ smartphones without knowing their SSID information, which is about $4 \sim 8$ times improvement compared to the similar attacks like KARMA and MANA.**

I. Introduction

Nowadays, with the explosive growth and popularity of free public Wi-Fi available in dense urban areas including airports, restaurants, and coffee shops, the security of using public Wi-Fi is gaining more and more concern. According to [1], 68% of public and unsecured Wi-Fi users fell victim to cybercrime.

One major issue that can arise when attempting to use free public Wi-Fi is of joining a rogue Wi-Fi access point (AP). In such a case, an attacker clones an AP that a client has previously connected to, tricks the client into associating to it, and then obtains the client's confidential data. This is generally referred to as the *evil twin attack* (ETA) [2].

KARMA attack [3] is the most popular approach to implement ETA. KARMA attack leverages the fact that mobile phones keep a list of SSIDs of the wireless networks it has connected to in the past on the Preferred Network List (PNL). Every time the Wi-Fi interface is turned on, a mobile phone periodically sends multiple probe requests (also referred to as *direct probe requests*) which contain the SSIDs in the PNL. After receiving a direct probe request, KARMA attacker will reply with an appropriately crafted probe response mimicking the SSID in the probe request to trick the device.

For example, if your mobile phone sends out a direct probe request with the SSID 'AP123' (the SSID of an AP it has connected to), KARMA attacker replies with a probe response mimicking the SSID 'AP123'. If this AP happens to be an open AP, the following association and authentication will pass easily and your mobile phone will automatically connect to it.

Due to the severity of ETA, a remarkable amount of work has been done and most of the solutions are focusing on how to detect the presence of the rouge APs [4][5][6][7]. In the meantime, the Wi-Fi standard itself is keeping improving the level of security and becomes more and more resilient to different kinds of attacks including the ETA. We noticed that to launch ETA, an adversary relies on the SSIDs of the APs to which a client has previously connected. Therefore, a viable approach is to change the way of how clients send probe requests. This matches well with the current Wi-Fi protocols of most Android and iOS operating systems. Most Wi-Fi clients now only send 802.11 *broadcast probe requests*, generically asking for available APs nearby [8]. These broadcast probe requests do not contain any SSIDs. Upon receiving a broadcast probe request from a nearby client, an AP sends a response containing its own SSID. The client then looks at whether the AP's SSID is contained in its PNL and connects to the AP if there is a match. From the discussion above, we can easily see that without the SSID information, ETA like KARMA cannot work well as before.

Correspondingly, an enhanced attack called MANA [9] is recently proposed that is able to launch ETA under the current Wi-Fi protocols. MANA leverages the fact that some unsafe mobile devices still send direct probes containing SSIDs. MANA stores these SSIDs and utilizes them to respond to a broadcast probe from a client. If one of the SSIDs happens to be in the PNL of the client, a successful hit occurs and the client will be associated to the attacker.

As an illustration, we deployed a MANA and a KARMA attacker in a canteen[1]. The two tests were carried out at the same time and the distance between the two attackers is about 40 meters to avoid any interferences. Table I compares the corresponding results from MANA and KARMA attackers.

We can see that, compared to KARMA attacker which only associated 24 out of 614 mobile phones (with hit rate $h = 24/614 = 3.9\%$), MANA attacker has significantly higher performance, connecting 46 out of 688 mobile phones (with

[1]Before doing any experiments described in this paper, we have obtained the ethical and legal approval from the corresponding authority. All information collected will be kept strictly confidential and will only be used for research purposes. We are willing to provide the approval documents when required.

1063-6927/17 $31.00 © 2017 IEEE
DOI 10.1109/ICDCS.2017.148

hit rate $h = 6.6\%$). The improvement mainly comes from the fact that MANA successfully connected 16 clients sending broadcast probe requests.

To evaluate the actual performance of MANA on hitting clients sending broadcast probes, we define the *broadcast hit rate*, denoted as $h_b = \frac{n_b}{N_b}$. h_b defines that among N_b clients sending broadcast probes, n_b are finally connected. Note that $h_b = 0$ for a KARMA attacker since it cannot lure any clients sending broadcast probes. However, the fact that $h_b = 3\%$ indicates that there still might be much room for MANA to improve.

TABLE I: Comparing the results of KARMA and MANA

Attack	Total probes	Direct/ Broadcast	Clients connected	h	h_b
KARMA	614	85/529	24 (direct); 0 (broadcast)	3.9%	0
MANA	688	103/585	27 (direct);19 (broadcast)	6.6%	3%

Therefore, we ask the following questions: (1) Why MANA has such a low hit rate for clients sending broadcast probes? (2) Is it possible to launch an ETA with a much higher performance than MANA?

To answer these questions, we first analyze the problems of MANA based on tests in different public places. Then we propose City-Hunter. The fundamental architecture of City-Hunter, in some sense, is similar to MANA: upon receiving a broadcast probe from a client, an attacker responds with some previously stored SSIDs, hoping to hit the PNL of the client. However, City-Hunter is different from MANA in all of the major aspects: (1) the sources of SSIDs (2) the way of maintaining these SSIDs, and (3) the way of selecting SSIDs to hit a client. City-Hunter establishes and maintains an SSID database by integrating both offline and online information. Meanwhile, it smartly chooses SSIDs to maximize the hit rate.

We evaluate the performance of City-Hunter in public places with different crowd density (the number of people per square metre) and different mobility pattern (the average walking speed of people). The results demonstrate that the average broadcast hit rate h_b of City-Hunter is $12\% \sim 18\%$, about $4 \sim 8$ times improvement compared to MANA.

II. Related works

Recently years, how to launch and thwart an evil twin attack (ETA) has attracted many attentions. KARMA attack [3] is the most popular approach to implement the ETA. Correspondingly, a remarkable amount of work has been done to detect the presence of ETA. According to who will be responsible for the ETA detection, existing solutions can be classified into two categories. The works proposed in [6][7] rely on the network operator which runs a system to detect ETA. On the other hand, in [10][4][11][5][12], mobile clients are responsible for detecting rouge APs.

Besides the detection of ETA, the current Wi-Fi protocols of most Andriod and iOS operating systems have significantly improved the level of security. Most Wi-Fi clients now only send 802.11 *broadcast probe requests* which do not contain any SSIDs. Without the SSID information, ETA like KARMA cannot work well as before.

Correspondingly, an enhanced attack called MANA [9] is proposed. MANA stores these SSIDs collected from unsafe mobile devices which still send direct probes containing SSIDs, and utilizes them to respond to a broadcast probe from a client. However, the real deployment of MANA shows that there is still much room for MANA to improve.

In this paper, we take an attacker's perspective and focus on finding the answer to the relatively low attacking rate of MANA. Based on the analysis of MANA, we propose a new attacking strategy, called as City-Hunter. The deployment of City-Hunter demonstrates that it can achieve much higher attacking rate compared to KARMA and MANA.

III. Preliminary design

In this section, we wish to find out why MANA has low efficiency to hit clients sending broadcast probes, then we describe the corresponding solutions.

As we previously discussed, MANA implements two tasks: (1) upon receiving a direct probe request, MANA adds the contained SSID into the database; (2) upon receiving a broadcast probe request, MANA responds with all the SSIDs in the database. However, further analysis shows that there are some problems when MANA implementing these two tasks.

A. Changing the way of sending SSIDs

Intuitively, a larger SSID database should help MANA hit more clients than a smaller one. However, our experimental results show a different story.

We utilize the data collected in our previous experiment in the canteen. Fig. 1(a) shows the size of the SSID database and the total number of clients that were connected. Note that the latter only considers the clients sending broadcast probes, which directly reflects the efficiency of the SSID database.

We can see that in this 30-minute test, both of these two curves increase steadily with time. This however indicates that the increase of SSID database does not help MANA hit more clients. To observe it more clearly, we define the real-time broadcast hit rate as $h_b^r = \frac{n_b^r}{N_b^r}$. h_b^r shows that within a certain time window, n_b^r out of N_b^r clients are successfully hit by the SSIDs in the database. h_b^r reflects the real-time efficiency of the SSID database. h_b^r in this test is shown in Fig. 1(b), in which we calculate h_b^r for every 2-minute time window. From Fig. 1(b), there is no salient evidence to show that the increase of the size of SSID database can help to improve the efficiency of the SSID database.

After further analysis, we found the reason. According to IEEE 802.11 standard, after having sent a probe request, a client will wait for about 10ms for a first probe response, and if no probe response is received within 10ms, it waits for another 10ms. This means that for an AP working on a certain channel, a client can only wait at most 10ms after receiving a first probe response. Considering transmitting one probe response from an AP to a client takes about 0.25ms

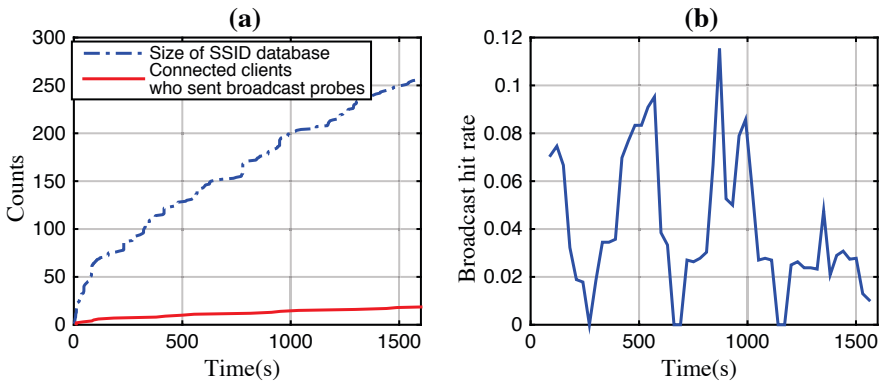

Fig. 1: (a) The size of the SSID database and the corresponding number of clients that were connected to the attacker using the SSID database. (b) The real-time broadcast hit rate h_b^r.

[13], a mobile client hence can only receive about 40 probe responses in one scanning period from a certain AP.

From the discussion above, we can see that in MANA's way of sending the SSID database all at one time, only the first 40 SSIDs can be received by the client. This is one of the most important reasons of the low efficiency of MANA. There is a simple solution to handle this problem: upon receiving a broadcast probe request from a client, the attacker will reply 40 SSIDs in the database that *have not been sent to this client before*. This requires that the attacker should record the MAC addresses of all the clients it tried to connect but failed in the past, and maintains an un-tried SSID list for each of them. We can expect that by effectively utilizing more SSIDs, the hit rate can be improved.

B. Incorporating more SSIDs

MANA purely relies on received direct probes to establish its SSID database. The consequences are twofold. First, the size of SSID database is small at the initial stage, and may also increase slowly if MANA attacker is deployed in a less crowded area. Second, the efficiency of SSID database is purely determined by the 'quality' of received direct probes, which we cannot control in real conditions. We may end up with a large database but the contained SSIDs have low probability to hit other clients.

Intuitively, if we can obtain, using sources other than direct probes, some popular SSIDs that may exist in the NPLs of many mobile phones, then adding these SSIDs in the database can help to improve the hit rate. Fortunately, information of these SSIDs can be found from Wireless Geographic Logging Engine (WiGLE) [14], whose database contains all the wireless networks with their geographic locations.

With data from WiGLE, shall we incorporate all SSIDs found in the WiGLE? The answer is negative. Doing this way can generate a very large SSID database. As only 40 SSIDs can be tested for each broadcast probe received, it is quite possible that a MANA attacker does not have a chance to send a large number of SSIDs to a certain client, even the latter is in a relatively static environment like canteens. Therefore, we only select part of the SSIDs in the WiGLE.

The selection is based on the following two criteria. First, SSIDs of APs which are geographically near to the attacker are selected. This is based on the assumption that many mobile phones passing by the attacker may have connected to the nearby APs before. In City-Hunter, we simply select 100 the SSIDs near to the attacker.

Second, we also found that some SSIDs, although their APs are not located close to the attacker, have high probability to hit the nearby clients. There are two types of APs under this category. SSIDs of the first type have APs city-wide distributed. Among the examples are the SSIDs of some small but popular brand-shops, like '7-Eleven' and 'Starbucks'. For example in Hong Kong, 924 APs have the same SSID '7-Eleven Free WiFi'. SSIDs of the second type have APs located in some important areas like large railway station, airport, and shopping centers. For example, 231 APs have the SSID '#HK Airport Free WiFi' in Hong Kong airport. This SSID can have high hit rate since many people have been to the airport and connect to the APs.

For the SSIDs belong to the second type, we simply count the number for their APs in the whole city and select those with high counts. Finally, we should note that only SSIDs belong to free APs from WiGLE are selected, which allows further association and authentication to be implemented automatically without user interaction.

C. Experiment result

In this section, we show the performance of City-Hunter after incorporating the two improvements described above.

Experiment in the canteen. We carried out a 30-minute experiment in the canteen where we tested KARMA and MANA. Table II shows the results of City-Hunter. For comparison, the results for MANA deployed in the canteen are also illustrated.

In this test, City-Hunter received probes from 626 mobile phones, and successfully connected 120 out of them. The hit rate $h = 19.1\%$. Compared to MANA which has $h =$

TABLE II: Comparing the results of MANA and City-Hunter incorporating the two improvements

Attack	Total probes	Direct/ Broadcast	Clients connected	h	h_b
MANA	688	103/**585**	27 (direct);**19** (broadcast)	6.6%	**3%**
City-Hunter	626	85/**541**	34 (direct);**86** (broadcast)	19.1%	**15.9%**

6.6%, City-Hunter has a significantly higher hit rate. This improvement is due to the SSID database of City-Hunter, using which City-Hunter has successfully hit 86 out of 541 clients ($h_b = 15.9\%$). In contrast, using its SSID database, MANA only hit 19 out of 585 clients, with $h_b = 3\%$.

Why City-Hunter has a higher h_b than MANA? This can be attributed to the two improvements we described previously. The effect of the first improvement, changing the way of sending SSIDs in the database, can be illustrated in Fig. 2(a). The figure shows the number of SSIDs that have been sent to each of the connected clients. We can see that for these connected mobile phones, the number of SSIDs that have been tried ranges from 20 to 250, with an average of 130. As a comparison, in MANA, at most 40 SSIDs are received for each client. Sending more SSIDs to a client obviously can improve the h_b of the attacker.

The effect of the second improvement, incorporating SSIDs from WiGLE, can be observed by the fact that among 86 connected clients sending broadcast probes, 64 (about 74%) are hit by the SSIDs from WiGLE.

Experiment in the subway passage. We carried out another experiment in the subway passage, and the results are shown in Table III. We can see City-Hunter deployed in the passage has much lower performance than in the canteen. The key problem is the low efficiency of the SSID database in the passage: the broadcast hit rate h_b in the passage is only about 4.1%, much lower than 15.9% in the canteen.

TABLE III: Performance of City-Hunter in the subway passage

Scenarios	Total probes	Direct/ Broadcast	Clients connected	h	h_b
Subway Passage	1356	178/**1178**	37 (direct);**49** (broadcast)	6.3%	**4.1%**

Why in the passage City-Hunter has a much lower h_b? Further analysis shows that it is due to the different movement patterns of the people in these two conditions. In the canteen, most people near the attacker are sitting still, and therefore more SSIDs can be tried for each client. While in the passage, considering people nearby are keeping moving, City-Hunter normally is not able to send as many SSIDs to the clients as in the canteen, and its h_b is therefore lower.

As a demonstration, Fig. 2(b) shows the histogram of the number of SSIDs that have been tested for all the 1178 mobile phones sending broadcast probes in the subway passage. We can see that for most clients (about 70%), only 40 SSIDs have been tested; and 80 SSIDs have been sent to 22% clients. Compared to the case in the canteen where on average 130 SSIDs have been tested for each connected mobile phone, the number of SSIDs sent in the passage is much smaller.

We can see that City-Hunter is not able to work well in an area where people are constantly moving. This problem will be addressed in the next section.

IV. Advanced Design of City-Hunter

In this section, we will describe the architecture of the final City-Hunter that we designed.

A. System overview and design principles

In an place where people are moving continuously, an attacker normally is only able to test a limited number of SSIDs for a client. Therefore, upon receiving a broadcast probe from a client, it is natural that *the SSIDs which have higher probability to hit the client should be sent earlier*. This is the key idea for City-Hunter to improve its hit rate when deployed in a subway passage-like environment. Then we have one question to answer: given an SSID, how do we estimate the its probability to hit a client?

We argue that an SSID should have a higher chance to hit a client if one of the following criteria is satisfied:

- Popular in WiGLE: The SSID has many APs located near the attacking location or are city-wide distributed.
- Good record of hit: Using this SSID, the attacker has successfully hit many clients after deployment.
- Good freshness: It is not long since the SSID's latest successful hit.

The first two statements are self-explanatory and will not be justified for brevity. The third requirement about freshness is obtained from our observation. We found that if an SSID just successfully hits a client, then within a short period of time, it has a relatively high probability to hit more clients. Further analysis reveals that this is because many people walking together have certain kinds of social relationship (e.g. families, friends). As a result, their mobile phones usually share some SSIDs in common. Therefore, if an SSID has just hit one person, it has higher probability to hit his/her companions.

To evaluate an SSID's probability for a potential hit, a straightforward strategy is to first assign three values, one for each of the three criteria, and then take the linear combination as the indicator of the probability of the SSID to hit a client. However, this requires careful tuning of many parameters including the values for three criteria and the coefficients of combing these values afterwards. Instead, we use the following method to incorporate these three types of information.

The basic idea is that when selecting SSIDs to hit a client, we select from two buffers, one containing a number of SSIDs with good popularity and the other includes a few SSIDs with good freshness. Here the *popularity* of an SSID is initially set by WiGLE and then updated according to its hit record after the attacker is deployed. The freshness of an SSID is purely determined by its latest time of hit.

Fig. 3 shows the logic flow of how City-Hunter addresses broadcast probes. Note that for the direct probes, City-Hunter utilizes the same approach as in KARMA and details are omitted. From Fig. 3, we can see that City-Hunter implements the following four steps:

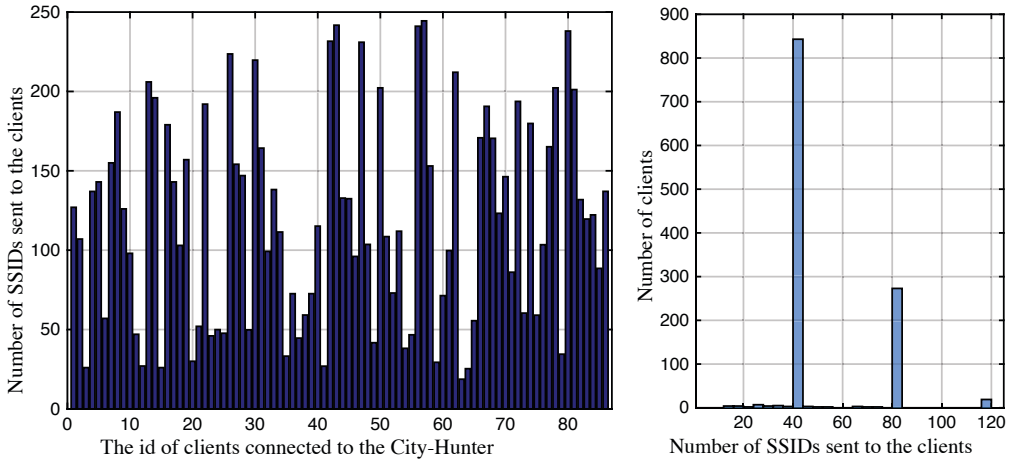

Fig. 2: (a)The number of SSIDs that have been sent to each of the connected mobile phones in the canteen, (b)The histogram of the number of SSIDs that have been tested for all the 1178 mobile phones sending broadcast probes in the passage.

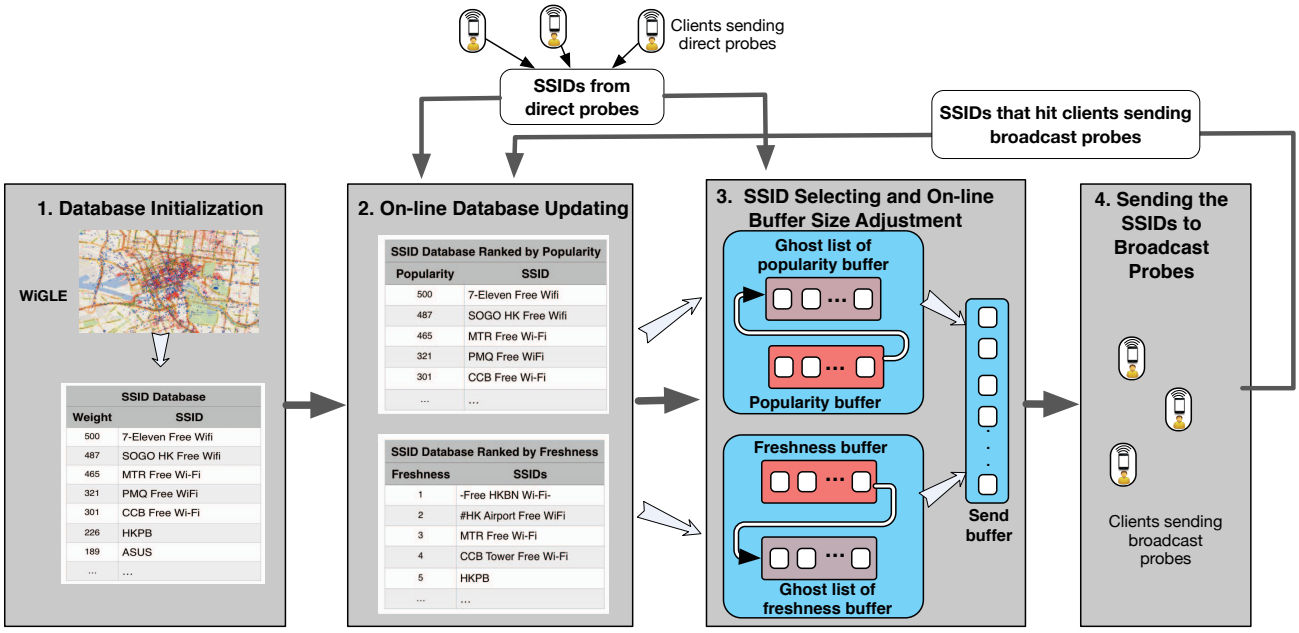

Fig. 3: The logic flow of City-Hunter.

1. **Database initialization**. This step generates an SSID database before the attacker is deployed. Some SSIDs from WiGLE are added into the database. Each SSID is given a corresponding weight indicating its popularity in WiGLE.

2. **On-line database updating**. This step is to maintain the SSID database by adding new SSIDs into the database or updating weights of SSIDs after the attacker has been deployed on site. The weight of each SSID is initially determined by WiGLE and then updated according to its actual hit record. In addition, we maintain a fixed-sized buffer which contains a certain number of SSIDs with recent hit record. To summarize, we can say that we maintain two databases, one ranked by popularity, and the other ranked according to freshness.

3. **SSID selection and buffer size adjustment**. This step is to select the SSIDs for responding to a broadcast probe. From the previous two databases, we obtain two buffers, one containing SSIDs with good popularity, and the other with good freshness. In addition, subject to the limit of total size (40), we wish the size of the two buffers can be adjusted automatically according to the real conditions. We

will describe this part in detail in Section IV-C.

4. **Sending SSIDs to broadcast probes**. This step is to send the selected SSIDs to respond to the broadcast probe. Then we go to the step 2 and the procedures repeat. In the following sections, we will describe some details of the four steps above.

City-Hunter shown in Fig. 3 only sends 40 SSIDs that can maximize the hit rate. Therefore, it is mostly suitable for areas like subway passage where people have high mobility pattern. For conditions where City-Hunter has chance to send more than 40 SSIDs to a client, we modify it slightly in the similar way as described at the end of Section III-A. City-Hunter records the MAC address of all the clients it tried to connect and the corresponding SSIDs that have been sent this client. Upon receiving a probe request, it will reply 40 SSIDs in the database that have not been sent to this client before.

B. Database initialization and updating

Having added SSIDs from WiGLE into the database, we need to assign an initial weight for each SSID. A natural way of assigning the weight is according the number of APs. This way of assigning weight seems to appropriate for SSIDs with APs distributed city-wide like '7-Eleven Free Wifi', but can underestimate some SSIDs with APs located in some important functional areas of a city. For these SSIDs, although the number of corresponding APs may be limited, the high hit rate of these SSIDs in our tests indicates a greater weight should be assigned than using the APs' number. For example, there are 231 APs with the SSID '#HK Airport Free WiFi'. According to the number of the APs, this SSID only ranks 13, but the hit rate of using this SSID is constantly among top 5 in all of our tests carried out in Hong Kong.

The reason for the high hit rate of these SSIDs with limited APs lies in their location: The APs of these SSIDs are located in hot areas that many people may have visited. This indicates that a more accurate way of assigning initial weight should not only consider the number of APs, the number of people (including residents, tourists, commuters, etc.) around these APs must also be included. Therefore, we need something like a heat map of the city, with the heat value at a location reflecting the number of people at that location. For a certain SSID, we find out the heat values at the locations of all its APs, add them together, and take the summation as the final heat value for this SSID. An SSID with a large heat value generally has APs located in crowded areas, and hence has higher probability to hit a client.

However, such a heat map is not readily available. Using district-level population distribution of a city is too coarse-grained and can even be biased since commuters and tourists are not taken into consideration.

We take the following approach to generate a heat map. The information of a number of people in a certain area is estimated by the number of panoramio photos of the area that people posted to Instagram. We assume that the number of photos of an area posted roughly reflects the number of people there. Fig. 4 shows the heat map we generated for two districts of Hong Kong (Kowloon and Lantao Island). The green areas have few photos and the red areas have more photos geotagged. From Fig. 4(a), we can see some red areas in Kowloon include iSQUARE and the ONE, the two large shopping malls in Hong Kong. In Fig. 4(b), Hong Kong airport is a hot area in Lantao Island. All these red areas on the heat map are crowded places with a large number of people. The fact justifies the applicability of our proposed approach.

Table IV compares the top 5 SSIDs ranked by the number of APs and the top 5 SSIDs with the largest heat value. We can see that by considering the number of people, '#HKAirport Free WiFi' now is in the top 5. Another SSID that is upgraded into top 5 is 'Free Public WiFi'. This SSID has about 400 APs mostly deployed in various crowded locations.

TABLE IV: The top 5 SSIDs selected using different criteria

Rank	Top 5 SSIDs with maximum APs	Top 5 SSIDs with maximum heat values
1	-Free HKBN Wi-Fi-	Free Public WiFi
2	7-Eleven Free Wifi	#HKAirport Free WiFi
3	-Circle K Free Wi-Fi-	-Free HKBN Wi-Fi-
4	CSL	FREE 3Y5 AdWiFi
5	CMCC-WEB	7-Eleven Free Wifi

With the heat value of a SSID, we determine its weight using the ratio method proposed in [15].We first rank the 200 SSIDs selected according to their heat values. The top SSID will be assigned with weight 200 and the lowest SSID will be assigned with weight 1. The same rule also applies to the selected 100 SSIDs that are nearby to the attacking location.

After deploying City-Hunter, its database will be updated by adding new SSIDs into the database and by adjusting the weight of SSIDs according to the real conditions. The updating occurs in the presence two conditions: (1) an SSID just successfully hits a client sending broadcast probe, and (2) the attacker receives an SSID contained in a direct probe.

C. Dynamic buffer size adjustment

Selecting a SSID in the database to respond to a broadcast probe should consider both its popularity and freshness. City-Hunter maintains two buffers. One buffer, called Popularity Buffer (**PB**), contains SSIDs with highest weight in the SSID database and the other buffer, called as Freshness Buffer (**FB**), contains SSIDs with recent hit. The problem then becomes under the constraint of total size 40, how to determine the size of PB and FB. Moreover, instead of setting fixed sizes like 35 v.s. 5, we feel it is more appropriate that the size of the two buffers can be automatically adaptive to the real conditions that may change with time and locations. In other words, we need to balance the popularity and the freshness according to real conditions.

Our current problem of selecting SSIDs to hit a client, in some aspects, is similar to cache algorithms which decide which pages should be stored in the cache to maximize the hit ratio (defined as the frequency of a searched-for item is actually found in the cache). Furthermore, most of the classic cache algorithms, when updating the items in the cache, also

(a) Kowloon (b) Lantao Island

Fig. 4: The heat map generated for three districts of Hong Kong. (a) Kowloon, and (b) Lantao Island.

consider the freshness (generally called as recency) and/or the popularity (called as frequency) of items. In particular, the Adaptive Replacement Cache Algorithm (ARC) [16] is able to balance between recency and frequency dynamically according to the real workload patterns.

Inspired the ARC, we design the following approach to adaptively adjust the size of the PB and FB. The key idea lies in the two ghost lists of PB and FB (see step 3 in Fig. 3). The ghost list of PB contains SSIDs which are less popular than those in PB, and the ghost list of FB contains SSIDs that are less fresher than those in FB. When selecting SSIDs, some SSIDs in the two ghost lists are also included. Furthermore, the performance of SSIDs in the two ghost lists (whether we have a successful hit using these SSIDs) will be utilized for adjusting the size of PB and FB. For example, if we have a successful hit using an SSID in the ghost list of PB, this is a sign that the PB is to small, then the size of the PB is increased by one. Considering the constraint of 40, the size of FB will be decreased by one at the same time. Likewise, a hit using an SSID in the ghost list of FB will increase the size of FB.

With this mechanism, City-Hunter adapts itself to the different conditions. If currently people near the attacker are walking in groups and share similar SSIDs in their PNLs, it would have more hits in the ghost list of the FB and thus the size of FB will increase, favoring more to SSIDs with recent hit. And vice versa, if people nearby do not have certain kinds of relationship, it would have more hits on the ghost list of PB and thus the size of PB will be increased, favoring more to SSIDs with high popularity.

In City-Hunter, the size of both ghost lists is 20. Each time when choosing SSIDs, besides selecting SSIDs in the PB and FB, we randomly select 2 SSIDs (10%) from each of the ghost lists to replace the lowest two in the PB and FB.

V. Experiments and Evaluation

A. The evaluation in different areas

We prototype City-Hunter with the commodity Raspberry Pi. The transmission power is set to be 100 mW. We deployed City-Hunter in four different places. The first two places are the subway passage and the canteen described in Section, and the last two places are a shopping center and a railway station. People in these areas have different mobility patterns: In the subway passage, almost all the people are moving relatively fast. While in the canteen, most people are static or moving with low speed. People in the last two places have a hybrid mobility pattern in the sense that some of them are relatively static while others are moving fast.

In each place, a number of tests were carried out at different time slots from 8am to 8pm. Each test last about 1 hour. Note that the database of City-Hunter were initialized before each test. The results are shown in Fig. 5. The four upper figures of Fig. 5 show the number of the clients whose probes were received in these places. These clients are further classified into four categories: those sending broadcast probes that were finally connected/not connected, and those sending direct probes that were finally connected /not connected. The bottom figures of Fig. 5 show the hit rate h and broadcast hit rate h_b in these tests.

For example, from the first stacked bar shown in upper figure of Fig. 5(a), we can see that in the passage test from 8am $\sim$ 9am, City-Hunter received probes from a total of 2562 clients. Furthermore, from top to bottom, the stacked bar shows that for clients sending broadcast probes, 312 were finally connected and 1968 were not; for clients sending direct probes, 66 were connected and 216 were not. The corresponding hit rate $h = 14.7\%$, and the broadcast hit rate $h_b = 13.6\%$. From Fig. 5, we have the following observations.

First, the number of clients whose probes were received (indicated as the height of bars in Fig. 5) in these tests shows

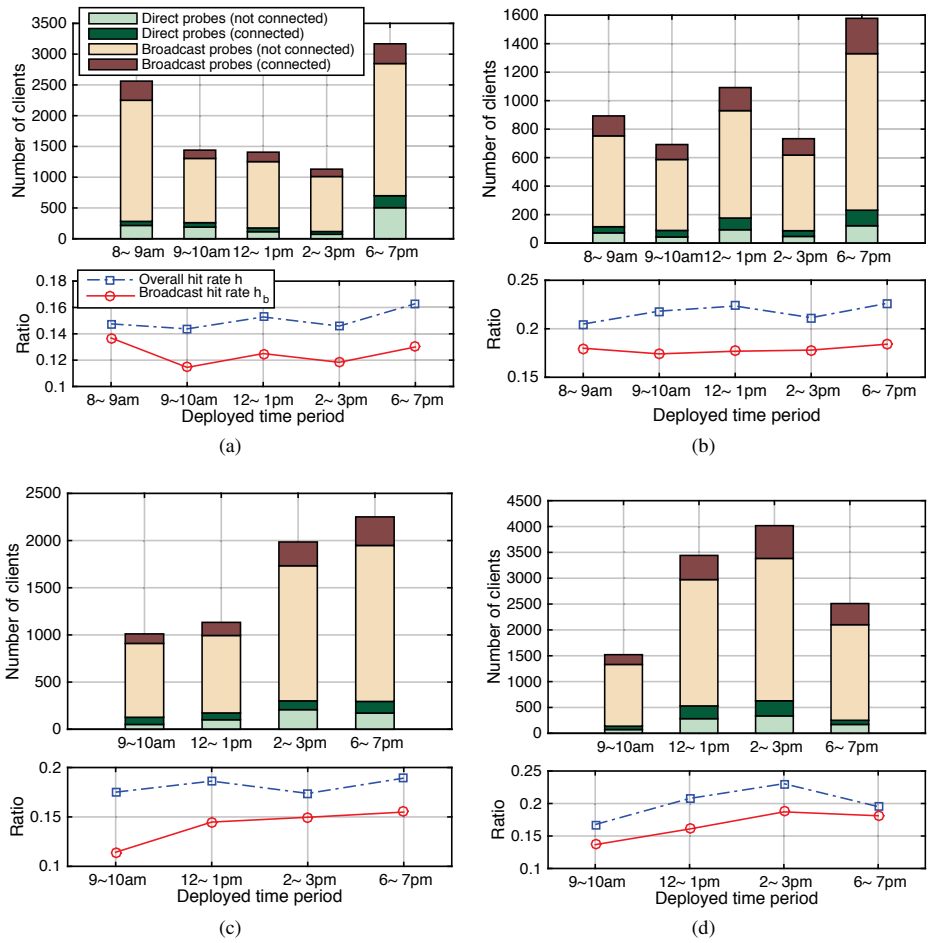

Fig. 5: The performance of City-Hunter in (a) a subway passage, (b) a canteen, (c) a shopping center, and (d) a railway station.

salient temporal pattern. For example, subway passage tests have two peaks corresponding to the two rush hours 8am $\sim$ 9pm, and 6pm $\sim$ 7pm, respectively; and tests in the canteen show three peaks during typical mealtime.

Second, in all tests, the overall hit rate h is always larger than the corresponding broadcast hit rate h_b. This is due to the fact that it is much easier for City-Hunter to trick a client sending direct probes than to connect a client sending broadcast probes.

Third, City-Hunter performs differently in these places. For example, the average h_b is 12% in the subway passage while is 17.86% in the canteen. The average h_b for the shopping center and the railway station is about 14% and 16.6%, respectively. We believe that the difference is mainly due to different mobility patterns. In a place where people are static or with low mobility, City-Hunter can sent more SSIDs to try to hit their mobile phones, and therefore can achieve higher h_b.

Finally, at a certain place, City-Hunter performs differently at different time slots. Particularly, it is interesting that both h and h_b, especially the latter, are higher in the rush hours. This pattern is salient in Fig. 5(a), and can still be observed in the remaining three figures. We suspect that there are two reasons. First, in rush hours when we have many people nearby, the SSID database can accumulate a large amount of SSIDs from direct probes within a short period of time, thus achieving higher hit rate. Second, in rush hours we may have more people walking in groups, and hence City-Hunter has more chance to use a recent SSID to hit one's companions.

To evaluate the effect of some designs in City-Hunter, such as using WiGLE or combining popularity with freshness, we further breakdown the SSIDs that successfully hit the clients sending broadcast probes. According to the source of these 'successful' SSIDs, they can be classified as from WiGLE, or from direct probes. These SSIDs can also be classified as those from the popularity buffer (and its ghost list), and from the freshness buffer (and its ghost list).

Fig. 6 shows the detailed breakdown of the SSIDs in these tests. For example, the first two stacked bars in Fig. 6(a) correspond to the passage test during 8am$\sim$ 9am. The first bar shows that among the 312 SSIDs that hit clients sending broadcast probes, 243 are from WiGLE and 69 are from direct probes. The ratio between two numbers, 243/69 = 3.5, is

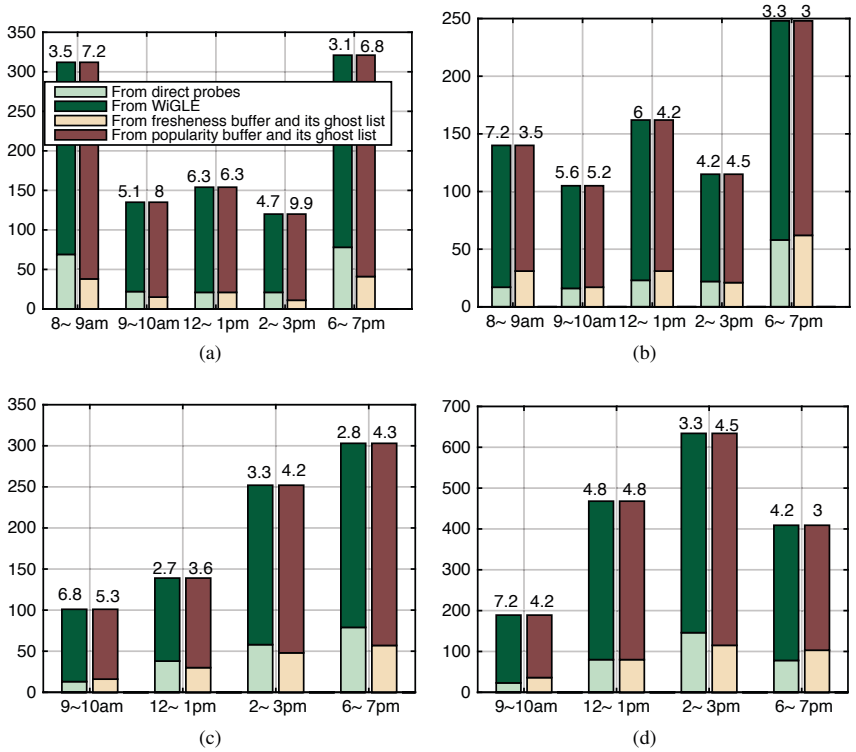

Fig. 6: The breakdown of the SSIDs that successfully hit the clients sending broadcast probes when City-Hunter is deployed (a) in a subway passage, (b) in a canteen, (c) in a shopping center, and (d) in a railway station.

shown on top of the bar. The second bar shows how these 312 SSIDs can be classified as from the popularity buffer and the freshness buffer. The ratio between them is also shown on top of the bar. From the breakdown of these SSIDs, we have the following observations.

First, compared to the direct probes, WiGLE contributes more SSIDs that make successful hit. Similarly, compared to the freshness buffer, more successful SSIDs are sent from popularity buffer.

Second, although WiGLE is more important than direct probes in contributing successful SSIDs, the contribution from the latter is higher during rush hours than in non-rush hours. For example in Fig. 6(a), the contribution from direct probes and from WiGLE is 1:3.5 during 8am∼9am, but decreases to 1:5.1 for 9am ∼10am. This may be because City-Hunter receives more direct probes in rush hours, and they are more likely to hit nearby people walking in groups.

Third, although popularity buffer contributes more successful SSIDs than the freshness buffer, the percentage of their contributions is different at different places. In the railway passage, the contribution from freshness buffer and from the popularity buffer is between 1:6.3 ∼ 1:9.9, while increases to 1:3∼1:5.2 in the canteen. The reason might be that in canteen, many people eating together have some social relationship and hence share common SSIDs. Therefore, SSIDs in the freshness buffer have a relatively higher chance to hit a client. On the contrary, most commuters in subway passage do not have such a strong pattern.

B. Further improvement in some particular conditions

We notice that in some conditions, there are several potential approaches to further improve the performance of City-Hunter. First, we observed that a client which has established a connection with an authenticated public AP barely sends out the probe request frames, which prevents the malicious from launching attacks to it. In order to address this issue, we can adopt an attack method called de-authenticated attack [17] to disconnect the clients from the AP and force them to initialize a new scanning process. This approach enables City-Hunter to work well in conditions where many people are already being connected to some local APs.

Besides, we noticed that iOS initially stores the SSIDs of mobile carriers' APs in the PNLs. For example, an iPhone that subscribes the service of the mobile carrier PCCW will automatically connect to the carrier's APs (SSID:PCCW1x) even the user has not used any public Wi-Fi services before. Therefore, we could efficiently lure the iOS users who subscribe those mobile carriers' services by adding the corresponding SSIDs into the database. Note that the SSID information of mobile carrier generally cannot be obtained from WiGLE, or from direct probes.

VI. Conclusion

In this paper, we introduce City-Hunter, a new attacking approach that can lure nearby mobile devices efficiently and stealthily in crowded urban areas. City-Hunter establishes an SSID database using the online AP distribution information and maintains the database in a real-time manner. Meanwhile, it chooses the most probable SSIDs to hit a client according to real conditions. We evaluate the performance of City-Hunter in different places at different time slots, and the results demonstrates the performance of City-Hunter. At last, we should emphasize that existing techniques to detect evil twin APs, either solutions relying on the network operator, or client-based solutions deployed on users devices, can still work as effective countermeasures for the City-Hunter.

VII. Acknowledgement

The work presented in this paper was supported in part by the NSF of China with Grant 61572218, NSFC Key Grant with project No: 61332004, and the NSFC/RGC Joint Research Scheme with RGC No: N PolyU51912.

References

[1] A. Minnaar, "'crackers', cyberattacks and cybersecurity vulnerabilities: the difficulties in combatting the'new'cybercriminals," *Acta Criminologica: Research and Application in Criminology and Criminal Justice*, pp. 127–144, 2014.

[2] B. Potter, "Wireless hotspots: petri dish of wireless security," *Communications of the ACM*, vol. 49, no. 6, pp. 50–56, 2006.

[3] D. A. Dai Zovi, S. Macaulay, *et al.*, "Attacking automatic wireless network selection," in *IAW 2005*, pp. 365–372, IEEE, 2005.

[4] C. Yang, Y. Song, and G. Gu, "Active user-side evil twin access point detection using statistical techniques," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 5, pp. 1638–1651, 2012.

[5] O. Nakhila and et al., "User-side wi-fi evil twin attack detection using ssl/tcp protocols," in *CCNC 2015*, pp. 239–244, IEEE, 2015.

[6] L. Ma, A. Y. Teymorian, and X. Cheng, "A hybrid rogue access point protection framework for commodity wi-fi networks," in *INFOCOM*, IEEE, 2008.

[7] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 3, pp. 449–462, 2010.

[8] J. Freudiger, "How talkative is your mobile device? an experimental study of wi-fi probe requests," in *SIGSAC 2015*, p. 8, ACM, 2015.

[9] W. Dominic and V. Ian de, "Manna from heaven: Improving the state of wireless rogue ap attacks," in *DEF CON 22 Hacking Conference*, 2014.

[10] H. Gonzales, K. Bauer, J. Lindqvist, D. McCoy, and D. Sicker, "Practical defenses for evil twin attacks in 802.11," in *GLOBECOM 2010*, pp. 1–6, IEEE, 2010.

[11] H. Mustafa and W. Xu, "Cetad: Detecting evil twin access point attacks in wireless hotspots," in *CNS 2014*, pp. 238–246, IEEE, 2014.

[12] F.-H. Hsu and et al., "A client-side detection mechanism for evil twins," *Computers and Electrical Engineering*, 2015.

[13] G. Castignani, N. Montavont, and A. Arcia-Moret, "Analysis and evaluation of wifi scanning strategies," in *Proceedings of the 5th Conference on Electrical Engineering, Merida*, 2010.

[14] Wigle.net, "Wigle: Wireless network mapping," in *http://www.wigle.net*.

[15] F. H. Barron and B. E. Barrett, "Decision quality using ranked attribute weights," *Management Science*, vol. 42, no. 11, pp. 1515–1523, 1996.

[16] N. Megiddo and D. S. Modha, "Arc: A self-tuning, low overhead replacement cache.," in *FAST*, vol. 3, pp. 115–130, 2003.

[17] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions.," in *USENIX security*, pp. 15–28, 2003.