WiFi vulnerability caused by SSID forgery in the IEEE 802.11 protocol

Krisztián Juhász
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
krisztianjuhasz@outlook.com

Valéria Póser
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
poser.valeria@nik.uni-obuda.hu

Miklós Kozlovszky
BioTech Research Center, EKIK
Óbuda University
Budapest, Hungary
miklos.kozlovszky@nik.uni-obuda.hu

Anna Bánáti
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
banati.anna@nik.uni-obuda.hu

***Abstract* — One of the most widely used, widespread technology of our day is Wi-Fi, enabling our wireless devices to communicate with each other. Its popularity stems from the fact that it is available in more and more places and is easy to use. Most of the connection steps are automatically done in the background at the client device. Thus, the devices automatically save the data needed to reconnect (the SSID, and the password). So that, when a known network is re-deployed, connection can be totally automatic without any user intervention. However, such convenience feature also poses significant risks as the devices can easily connect and automatically trust the network of an attacker with forged SSID. The aim of our research was to explore, test, such vulnerability and demonstrate adequate security solution against it.**

Keywords—WiFi vulnerability, SSID,

I. INTRODUCTION

Today, Wi-Fi has gained momentum, thanks to its growing availability and ease of use. In order to connect to a network, it is sufficient to choose the desired network from the SSID, or if a protected network requires a password, then any additional tasks are automatically done in the background by the client. Such convenient feature also poses significant risks because devices can easily connected to an attacker's fake network with same SSID. When WiFi devices search and receive an access point distributed information packet, the available SSID information is compared to the SSIDs in their database and automatically send a connection request to the access point. If this application is accepted by the other device, the connection will be successful and this access point (or a third device defined by it) will become our default gateway to the internet. So all our outbound and incoming traffic will pass through it. However, the fact that the choosed SSID belongs to the network we previously trusted is not to guaranteed at all. The main problem here is, that the IEEE 802.11 protocol only identifies Wi-Fi networks solely by SSID [1], [2].

If a malicious user creates a wireless network with an SSID that is already connected to, then our device automatically trusts and connects to it. Based on this, we can say that an attacker has the infinite number of opportunities to gain personal, banking or business information, passwords, or monitor or even modify our network traffic by filling in the role of the DHCP server.

II. IDENTIFIED VULNERABILITY PROBLEMS

A. WLAN architecture

Compared to the less flexible ad hoc network, the infrastructure mode is the widespread implementation of Wi-Fi networks. In its design, a Basic Service Set (BSS) topology is defined in which a central access point associated with a wired network regulates any information exchange of any device connected to its network. All data sent by the clients is transmitted to this destination via this station. As an extension to the BSS topology, the Extended Service Set (ESS) topology can be used to meet primary space requirements of an enterprise environment (where larger office spaces or tall building are required to be covered by the wireless network). In such an environment, a single access point is not enough, so the inclusion of additional access points is inevitable. Overlapping should be designed between each cells and SSID matching is a must to enable smooth transition between the network segments. As a result, a fake network broadcast by an attacker's access point can extend easily the original topology [3].

B. Open and close connection

The connection process of a device is started always by sending a validation frame. This message contains the information needed to identify the station, on which the access point accepts or rejects the connection request and answers it in response to an authentication frame.

In the case of an open network, it is sufficient for the station to send an application to connect, but when using an encrypted private network and pre-shared key authentication, the access point responds with a challenging message to the connection request, which the client must encrypt and send back to the challenging party. If the text is encrypted with the correct key, the access point accepts the device request.

If a station wants to disconnect from another station, it will send a de-authentication frame to it and disconnect it. An attacker can create a frame of this type on behalf of any device with a copy of the MAC address of the device, so it can be used to disconnect the device from the original network and attempt to connect to the attacker's forged network when establishing a new connection. [4]

The association request allows to reserve the access point’s resources. An association request includes the

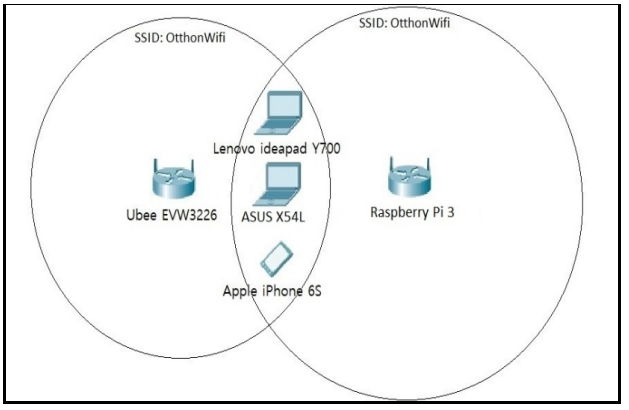

Fig. 1. Topology of the test environment

station-supported features, transfer rate values, and the SSID of the network in which resource allocation can be realized. If the access point accepts the request, it separates some amount of memory for the client, assigns an association identifier and then informs the requesting station in an association response message. If the station's request is accepted by the access point, then the client can communicate with another station on the network. The stations can free up the allocated resources with a dissociation frame. [4]

III. VULNERABILITY TESTS

The revision of the IEEE 802.11 standard confirmed that in many cases the authenticity of each message is missing, so even when receiving a deauthorization framework, we cannot be sure that the message that received comes from the right/trusted device. This way, an attacker can easily misuse such information and simply gain the role of another access point. This vulnerability was tested successfully in our test environment.

A. The realized test environment

As most users use smartphones or laptops for wireless internet, we focused on these test scenarios too. To implement the vulnerability, two access points were required, one that we trusted and one as an attacker, and at least one client device that serves the victim's function. The role of a trusted access point was realized by a Ubee EVW3226 wireless router, provided by an Internet service provider, and available in many households in Hungary. This is a remote device that can be managed remotely by the service provider. The device's default beacon interval is 100ms. The devices used by the attacker must have the power to perform and be configurable and portable enough to be applicable wherever they are used. So, a device needed to operate as a wireless access point, as well as for later use, to be able to work as a WEB and DNS server, was required to operate as a wireless access point. The Raspberry Pi "computer" provided us the most appropriate and easy-to-use solution. This device is an ARM-based, multi-faceted tool integrated on a bankcard sized PCB. The currently available constructions, such as the second and the third Raspberry Pi editions have adequate performance for such tests. Having carefully examined the specifications of the two devices, we chose the third edition.

The most significant difference between the two Raspberry Pi versions is the integrated network adapters, which is an important difference from our point of view. The third edition already has a built-in motherboard with a wireless network adapter that uses IEEE 802.11n, while the earlier version can only connect to a wired network. For this reason, Raspberry Pi 3 is a simpler solution to the task, which is easier to handle, so we have implemented the attacker's environment from a Pi 3. [6] [7] The role of client tools was played by an Asus X54L notebook with Windows 10 operating system and an Apple iPhone 6S smartphone with iOS 10.3.1. Wireless networks were analyzed and monitored using a laptop running the Lenovo IdeaPad Y700 Kali Linux system. The generated topology is illustrated in Fig. 1.

To demonstrate the pre-defined vulnerability, the network provided by the two access points must have the same SSID, which is "OtthonWifi". An unencrypted or password-protected, encrypted network can also be used to illustrate. We did a test on a password-protected network, pointing out that we should be careful not just for open, unencrypted access. The service access point did not require any special configuration, only the SSID value had to be changed to "HomeWifi", and the password could be arbitrary in this case. Raspberry Pi is compatible with a variety of operating systems, but the Linux operating system provides a good foundation for the task, since it can be configured as a DNS and WEB server and as a configurable access point.

Among the various Linux distributions, Kali Linux offers a number of tools beyond the services listed, which may provide a suitable backend for testing later on. The configuration is based on a newly installed Kali Linux operating system ARM 2017.02.

The components installed were: hostapd daemon, dnsmasq. The hostapd daemon that allows us to configure the wireless network adapter on the device to operate as an access point and send/receive Beacon requests for connection to clients. [8] In the hostapd.conf configuration file located in the / etc / hostapd / directory, we have specified the parameters required to run the program. By modifying these parameters, we have customized the wireless network. Here we have specified which interface to use and which driver it belongs to. Following information needed: the SSID, the channel, the authentication process, and its parameters, disabled the MAC address filtering, and enabled SSID scattering.

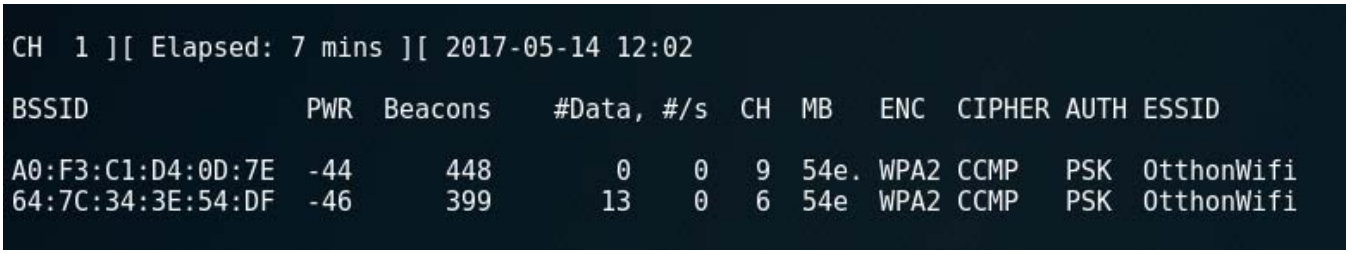

Fig.2. The airodump-ng's output

Dnsmasq is an optimal solution for small IP networks for assigning IP addresses to clients, as well as for sending requests through the appropriate interface, with both DHCP and DNS subsystems, and its functions fully comply with the task. [9] In order to configure it correctly, only a few lines are needed to meet our needs. As a final step, network address translation is still required to forward packets from private network addresses that are generated by Raspberry Pi to the external network on another interface. It is recommended to set up address translation and launching dnsmasq and hostapd services using a single launch script that takes care of all necessary components at the same time.

In the script in Fig 3. we first check if the dnsmasq service is running, if it is not already started with the dnsmasq command. Before specifying the address translation rule, we first delete all the previous rules, avoiding packet forwarding problems might be due to incorrect entries. Then we add a rule to the POSTROUTING chain of the NAT table that applies to packets forwarded to the eth0 interface. (Raspberry Pi connects to this Internet via this interface.) The -j MASQUERADE parameter specifies that the IP address of the client's internal network IP address is overwritten with the IP address of the eth0 interface. [10] Transmission of packets have been enabled in kernel settings. This was done with the sysctl -w command by setting net.ipv4.ip_forward to 1. After that comes the activation of the access point with the hostapd command, with the specified hostapd.conf configuration file. The pkill -f dnsmasq command may be required at exit. If we stop running hostapd, we do not need dnsmasq, so it's better to suspend it, so Raspberry Pi only uses the external DNS server. After the script is started, within a few seconds, the supplied wireless network becomes available for client devices with "HomeWifi".

The launch script contains:

```
#!/bin/bash
if [ -z "$(ps -e | grep dnsmasq)" ]
then
 dnsmasq
fi
iptables --flush
iptables --table nat --append POSTROUTING
        --out-interface eth0 -j MASQUERADE
sysctl -w net.ipv4.ip_forward=1
hostapd /etc/hostapd/hostapd.conf
```

pkill -f dnsmasq [Grab your reader's attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

Fig. 3. The content of the script

B. Exploring the vulnerability

After configuring Raspberry Pi 3, the two access points extend their network with the same SSID: HomeWifi. Both devices use the WPA2-PSK protocol and the same password to authenticate devices. In the first phase of the test, only the trusted router is active so that the first authentication and connection process were performed on client devices. After the connection was successfully completed, the attacker access point was activated but there was no change on the client devices, but there was still only one network called "OtthonWifi" to which they were linked. Only with the appropriate program were we able to show that there are two different access points available for that SSID. We used this Lenovo Y700 laptop, which also ran a Kali Linux operating system. For this task we used the airodump-ng of the Aircrack-ng software package, which is defaulted in the Kali Linux operating system. After setting up the network adapter in the monitor mode, all the packets and messages are became visible. This way, our device can monitor available network data via the wlan0mon interface. To list available network data, we need to issue the following command: In the output of airodump-ng mon0, the most important data on the network is displayed. The output obtained in the created environment is shown in Fig. 2.

The BSSID column indicates the access point's MAC address, which gives the only point to distinguish between networks and devices. These data can be read on the device's configuration interface or externally glued. The physical address of the attacker is A0: F3: C1: D4: 0D: 7E, while 64: 7C: 34: 3E: 54: DF identifies the trusted device. The PWR column shows the network strength of the networks. From the -44 and -46 values, it is not possible to detect the difference between the transmitter power of the devices at a distance of about two meters. [11]

The Beacons column shows the number of Beacon frames received. The fake device was activated later, but the Beacon Interval value resulted in multiple Beacon messages from the trusted device. Because of the lower value, it is more likely that an attacker's information packet will get to the victim seeking the available network.

Since the fake device has not yet been connected, no data frames have been sent yet. This data is read in the #Data column. For this, the following value, # / s, is also associated with the average of the number of data packets received per second for the past ten seconds [11].

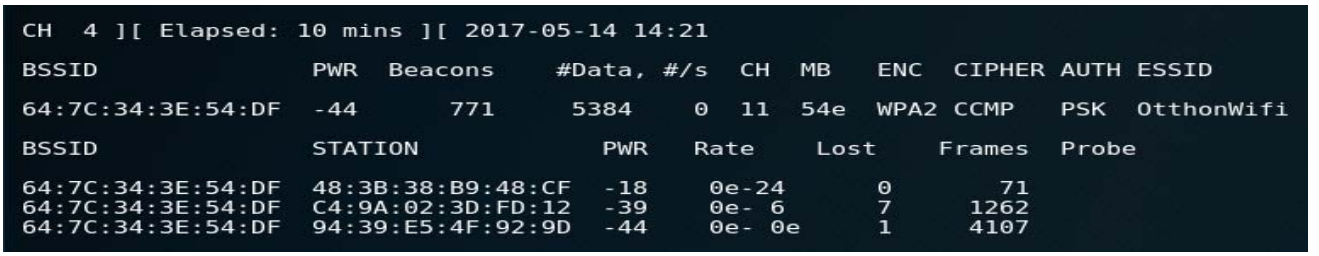

Fig. 4. List of devices connected to a given access point

The CH column specifies the channel used by the given network [11]. Airodump-ng is constantly switching between listening channels until monitoring stops, ensuring that each channel is properly evaluated.

The value 54e indicates the maximum transfer rate supported by the access points supported by the present configuration. The letter "e" followed by the number expresses support for the QoS service [11].

The ENC, CIPHER and AUTH columns are closely related, the encryption protocol and the algorithm and the form of authentication are defined [11]. From the point of view of the test, it was irrelevant that authentication requires access points, but in this case, a more realistic environment can be detected by vulnerability. The last column shows the SSID, which is the same for both devices. It can be seen that in addition to the SSID field, the two networks may differ in many other parameters, but these are ignored by our devices. Once an attacker acquires the MAC address of a trusted access point, it can be filtered out which clients communicate with that device to reveal potential victims. The following command can list all devices: airodump-ng-d 64: 7C: 34: 3E: 54: DF wlan0mon

In the command -d switch, only the MAC address of the access point is required and the name of the used interface. The output of the instruction is shown in Fig. 4.

Figure 5 shows three devices connected to the monitored router. The selected "victim" is the Asus X54L laptop with 94: 39: E5: 4F: 92: 9D physical address. Since the device has previously joined a trusted access point, packets will be sent to 64: 7C: 34: 3E: 54: DF until changes are made to the network. The required change can be triggered by sending deauthorization frames to the client device by impersonating the trusted router. In order to produce such frames, only the MAC address of the access point and the victim is needed. These data are openly available, as shown in the previous example. To deautentize frames, the aireplay-ng device can be used [12].

The sent frames cause the victim's device to disconnect from the trusted access point and look for a network that is available. Based on the incoming Beacon framework, it starts the connection process again to a known network. If the Beacon message of the attacker's device arrives earlier than the trusted router, the victim's device will start the connection process to the fake access point.

For an average user, it is almost unnoticeable that the device is "hijacked" and that it is no longer connected to the original access point. During browsing, for example, in such a deauthorization attack, when the user updates the web site, it is already being reconnected.

In our test, it was enough time to get the device disconnected from the original access point, and immediately built the new connection with the attacker. This was due to the fact that the attacker's Beacon's message had earlier arrived at the victim's device due to the shorter Beacon interval.

IV. PROTECTION AGAINST THE IDENTIFIED VULNERABILITY

Once we have uncovered and demonstrated vulnerability, the most important question is how to protect against vulnerability.

A. Prevention

IEEE 802.11 protocol is currently the only solution for users. Using the WPA2-Enterprise authentication method for companies, and the EAP-TLS protocol, it is possible to use certificates between the client and the authentication server. However, its application requires a properly configured RADIUS server and certificates to be issued and installed on client devices requiring higher levels of network and configuration knowledge.

This solution requires too much effort from the average user and is therefore not used in everyday life. For them, the only thing they can do to prevent them from opting out of automatic connection. However, for Android OS, this is not the case at all, and in the iOS operating system it was not possible until September 2017 to version 11. The only alternative to these systems is that after using the network to delete the network from known networks and manually connect to it every time. This will not allow an attacker anywhere to reach the device to unnoticeably connect to the network.

B. Detection

It is much more difficult to detect the hijacking when a user has already joined a trusted network and the attacker will be able to connect to his network back to his network, as the attacker uses the deauthorization attack described in the previous chapter, since it is natural that we are connected to the SSID network.

There have already been several publications about detecting fake access points, which is not surprising, as we have shown it as a problem that needs to be drawn to people's attention and finding a solution to it. We also examined the results and defense options published by the IEEE Federation.

In many cases, by detecting the position of the trusted device and the position of the attacker device, it detects the

attacker's appearance and its specific position by using the packets sent by the access points.

For example, hyperbolic positioning (HPB). However, this requires a separate dedicated receiver unit that is able to delimit the various access points within the range of the device. [13] Another possible approach to position-based detection is to examine the strength of the received signal (RSSI). [14] These methods are based on knowing where our only reliable access point is and this is the only credible position. Another recommendation published by the IEEE Federation suggests that access points use public keys and apply an SSH-like authentication process in the association process with session identifiers. [15] When writing our article, we found only a single, publicly available and accessible, detective solution that can be downloaded by anyone trying to identify false access points by testing Beacon frames. [16] The program was implemented using Python programming language to run on a Python compiler, the Aircrack-ng program package, and an SMTP server, because it will notify the network operator of the attacker's device as an e-mail form. The program can be used with different commands.

C. *Real Time Detection Software*

Based on the recommendations and solutions examined, they claim to be in the right hardware and software environments, or need to acquire different tools and use high-level networking skills, ie for administrators and operators. By contrast, the solution we have developed is easy to use by anyone, and does not require pre-training from the user, or the acquisition of additional hardware, tools, and other software. The basis for the proposed solution is the follow-up of the change of network parameters. First of all, we can examine the MAC address of the access point, but this identifier may also change when one of our APs converts to an extended wireless network, but it is worth paying attention to simple networks. In addition, authentication and encryption protocols used on the network can be regarded as permanent. Data on the channel may also be important if there is a single access point for the user. For the attacker, only information from the data link layer can be accessed openly, so it can also falsify the above data, but the data required for the operation of the upper layer is not available. The DHCP server configuration, the current private, internal network address remains unknown. It may also be a suspicion if the public network address changes from one moment to the next, with continued use of the Internet. We can conclude from the changes in the listed parameters, either individually or in combination, that new network joining took place, that is different from the previously used network. So we have designed and built a resource management solution running on Windows operating systems that monitors the network data in real time recognizes that the user's device could become a victim of SSID forgery and immediately alert him of the change in the network. After the detection, it logs the altered data so there is some information about the attacker's device and its internet connection, which can serve as evidence in an investigation. It is important to note that the software does not decide on a given network whether it is malicious or not, but it is able to detect signs of the attack described in previous section and determine whether the user has become a victim of an attack.

V. SUMMARY

We have identified vulnerabilities in wireless networks, which are a real threat to users in everyday life. To explore the vulnerability basics, we examined the core functionality of the IEEE 802.11 protocol family. The protocol analysis has confirmed that our Wifi-enabled devices will only scan the SSID of Beacon frames to identify each wireless network and decide on this information whether they know the network. All our devices are enabled by default to connect to a known network automatically, so they trust each access point that advertises its network with a known SSID.

We have demonstrated that not only automatic connection is a risk for users. An attacker can impersonate frames for the victim's devices and force the device to disconnect from the original access point and to connect to another network with the same SSID which was created by the attacker. By minimizing the sending time of Beacon frames, the attacker has even more chance against the original network to take over the connection. Two possible alternatives were made using a Raspberry Pi 3 to exploit the vulnerability that we wanted to illustrate how the vulnerability could work. We have explained what telling signs that changing network parameters should pay more attention for users to keep their data safe and to avoid becoming victims of data theft. However, most users do not have the right network and operating system management knowledge. We have designed and implemented an application running on a Windows operating system that monitors in the background the network information to detect signs of deauthorization attack and alert the user.

REFERENCES

[1] Matthew S. G.: 802.11 Wireless Networks: The Definitive Guide 2nd Edition. O'Reilly Media, 2005

[2] Justin Gibbins: 2.4GHz vs. 5GHz, (http://phorus.com/blog/2.4ghz-vs.-5ghz),

[3] IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (http://standards.ieee.org/getieee802/download/802.11-2012.pdf),

[4] Jim Geier: Understanding 802.11 Frame Types, (http://www.wi-fiplanet.com/tutorials/article.php/1447501/Understanding-80211-Frame-Types.htm),

[5] Speedguide: Ubee EVW3226, (https://www.speedguide.net/routers/ubee-evw3226-eurodocsis-30-voip-cable-router-3388),

[6] Raspberry Pi 2 Model B gyártói specifikáció, (https://www.raspberrypi.org/products/raspberry-pi-3-model-b/).

[7] Raspberry Pi 3 Model B gyártói specifikáció, (https://www.raspberrypi.org/products/raspberry-pi-3-model-b/),.

[8] hostapd Linux Documentation page (https://wireless.wiki.kernel.org/en/users/documentation/hostapd)

[9] Dnsmasq Manual Page (http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html).

[10] IPTABLES Manual Page (http://ipset.netfilter.org/iptables.man.html)

[11] Aircrack-ng - Airodump-ng, (https://www.aircrack-ng.org/doku.php?id=airodump-ng).

[12] Aircrack-ng - Aireplay-ng, (https://www.aircrack-ng.org/doku.php?id=aireplay-ng),

[13] Payal B., Christine L., Michel B.: Solution to the wireless evil-twin transmitter attack (http://ieeexplore.ieee.org/document/5764921),

[14] Zhanyong T., Yujie Z., Lei Y., Shengde Q., Dingyi F., Xiaojiang C., Xiaoqing G., Zheng W.: Detecting Evil-twin Attacks in Smart Homes Using TheReceived Signal Strength Indicator (http://eprints.lancs.ac.uk/81599/1/8_11_DRET_1_.pdf),

[15] Harold G., Kevin Bauer, Janne L., Damon M., Douglas S.: Practical Defenses for Evil Twin Attacks in 802.11 (http://ieeexplore.ieee.org/document/5684213),.

[16] Mohamed H., Mohamed I.: EvilAP_Defender (https://github.com/moha99sa/EvilAP_Defender).