Wi-Fi network testing using an integrated Evil-Twin framework

André Esser, Carlos Serrão
Instituto Universitário de Lisboa (ISCTE-IUL), Av. das Forças Armadas, 1649-026 Lisboa, Portugal
Information Sciences, Technologies and Architecture Research Center (ISTAR-IUL)
Lisboa, Portugal
Andre_Esser@iscte-iul.pt, carlos.serrao@iscte-iul.pt

***Abstract*—This work intends to present a newly developed Wi-Fi vulnerability analysis and exploitation framework with the objective of increasing Wi-Fi security. The developed framework focuses primarily on client-side vulnerabilities, currently a weakness on Wi-Fi connections, but can be extended to support any type of Wi-Fi attack. The framework was designed and is intended to be used by security auditors when performing intrusion tests on Wi-Fi networks. It can also be used as a proof-of-concept tool meant to teach and raise awareness of the risks involved when using Wi-Fi technologies. The developed framework is based on open-source software and is also available as open-source software, allowing developers to extend its functionality.**

***Index Terms*—Wi-Fi, security, evil-twin, attacks, pentesting, auditing**

I. Introduction

Currently, there is a proliferation of mobile devices that communicate with external services and among each other using wireless communication. Freedom offered by these wireless communication devices are unmatched. However, the data airwaves exchanged by the different mobile devices are filled with sensitive data, and face important security challenges [1].

In IEEE 802.11 wireless networks [2], there is a set security measures necessary to provide the security requirements for secure wireless communication [1], both on the access point site and on the end user device side.

The popularity and widespread of wireless communication technologies makes its deployment and usage a challenge for users without a minimum security criteria. This implies that weak security measures are deployed (or no security measures at all), weak security protocols are used and default router administration accounts and default security keys are used as well. This is far from an ideal security scenario and affects not only the security of the network infra-structure and services but also user devices security [1].

Although existing Wi-Fi security mechanisms are primarily focused on the protection of the network, the client side of this technology has been fairly neglected. Attacks such as the "Evil-Twin", where an attacker impersonates an access point, are still feasible and pose increased risk on Wi-Fi clients [3]. These attacks used to be somewhat complex to perform and required deep technical understanding. Now there are many free and open-source solutions that make them possible to exploit by attackers with little experience. The major vulnerability lies in the behavior of Wi-Fi clients where they actively search for access-points to connect to and without user interaction with the device [4]. This vulnerability, although discovered in 2004 [5], has not yet been effectively patched [6]. Even though there are proposed mitigations [7] [8] [9] [10], these have not been implemented in commercial products and do not solve the problem entirely.

Organizations conduct internal or external technical security audits. These audits, carried by information security experts, identify security flaws that affect organization - this is usually conducted at different levels, such as the network and system level. The phases involved on a Wi-Fi penetration test [11] include a set of pre-engagement interactions, intelligence gathering and threat modeling as well as vulnerability analysis, exploitation and post-exploitation and finally reporting [11], [12]. During the exploitation phase, auditors perform attacks on the Wi-Fi network according to the previously found vulnerabilities, this may include attacks on Wi-Fi clients as well. The post-exploitation mostly refers to attacks done inside the network once access is granted and also exploits on Wi-Fi client devices up to a persistent infection. Conducting an audit or penetration test on a Wi-Fi network can be a cumbersome job for the auditor. It requires the usage of a multiplicity of different tools with different purposes to test for the existence of serious vulnerabilities. This requires time and skills that can increase the duration of the audit and increase its costs. The need to run multiple tools that only serve a single purpose concurrently, unnecessarily increases the complexity of the penetration test and consequently decreases its efficiency.

This article presents a newly developed open-source Python-based framework [13] aimed to tackle the difficulties and problems involved with Wi-Fi networks auditing and penetration testing. The "Evil-Twin Framework" (the framework's name) increases the efficiency of a Wi-Fi penetration audit by cooperatively integrating the various features necessary to conduct the test into one single tool and place. The framework was developed having into consideration aspects such as third-party contribution, customization and extensibility. By providing an extensible platform with access to all core features of Wi-Fi communications it is possible to implement virtually every type of Wi-Fi attack on top of it. This eliminates the need for excessive amounts of tools running concurrently and hours of configurations as well as having to create new tools from

scratch.

II. Wi-Fi security threats and attacks

There are a variety of vulnerabilities affecting the different protection types used in Wi-Fi. It is important to notice that the vulnerabilities are present on different levels of the security protocol. For example, in WEP most vulnerabilities are introduced by using a weak encryption scheme while in WPA(2)-PSK it is the authentication mechanism that introduces one important vulnerability in the Wi-Fi security protocols. Therefore, throughout the history of Wi-Fi, there have been some moments where vulnerabilities affected the security of Wi-Fi users.

Evil-Twin and Karma are attack examples on known vulnerabilities that can best be described as a form of Wi-Fi phishing. They work by launching wireless hotspots that look and behave identically to trusted ones, luring regular users into connecting to them [10]. However, these hotspots do not connect to the network they mimic, they are mostly used in order to sniff the users traffic in order to get sensitive information such as credentials provided to websites [7], [10]. This is one of the main reasons why client-side Wi-Fi needs to be properly audited as well, and the developed Evil-Twin Framework, presented in the next section, can present an important added-value to the increase of the Wi-Fi security.

III. Evil-Twin Framework (ETF)

Wi-Fi security is a complex security issue, in particular on what concerns the end-user equipment. There are currently a significant number of tools that can be used to conduct Wi-Fi security audits however, such tools are sparse and suffer from lack of integration. From a security auditor perspective, in order to test different attack scenarios, it would require multiple different tools for each of them. Therefore, in order to facilitate the Wi-Fi auditing processes, and offer the possibility for the auditor to experience multiple scenarios and integrate multiple tools, a new Wi-Fi auditing framework was developed - the "Evil-Twin Framework" (ETF) - that primarily focus on the analysis of vulnerabilities on the client-side.

A. Architecture

ETF was entirely developed using the Python (version 2) programming language [13]. Python was chosen as the development language since it is very easy to read and therefore to contribute. In addition to that many useful libraries used on similar auditing tools, such as "`scapy`", were developed for Python and only support version Python version 2.

The ETF architecture (Figure 1) is composed by a set of different modules that interact with each other. The framework's settings are all configured on a single configuration file. The user can verify and edit the settings through the user interface via the "`ConfigurationManager`" module. All other modules can only read these settings and b executed according to them.

The ETF is prepared to offer multiple user interfaces that enable the auditor interaction with the framework. Currently, only an interactive console interface similar to the "`Metasploit`" framework is present. A graphical user interface and a command line interface are currently under development. These interfaces are being developed for desktop use, however mobile interfaces could also be developed in the future. The user can edit the settings in the configuration file by using either the interactive console or the future Graphical User Interface (GUI). The user interfaces can interact with every other module of the framework.

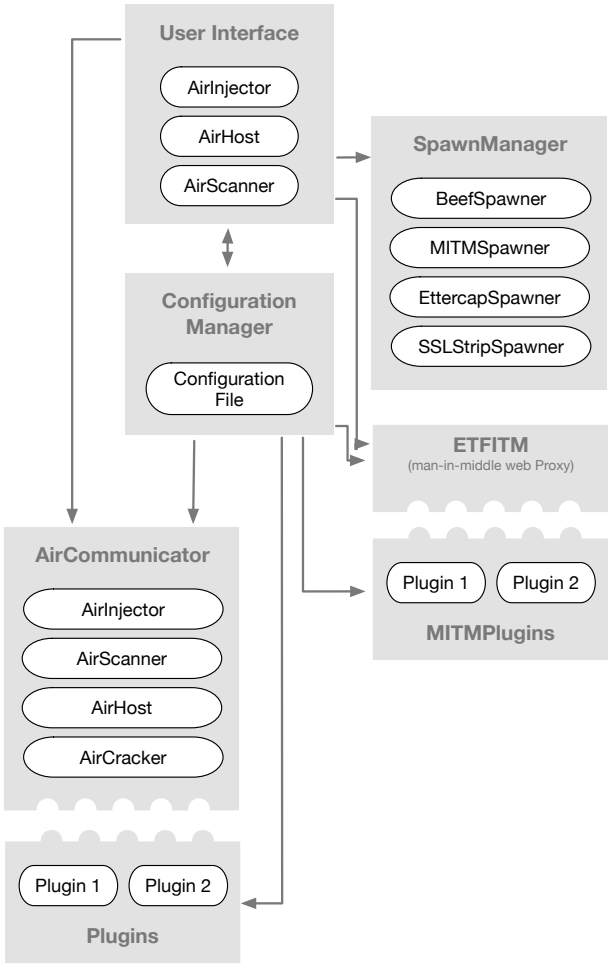

Fig. 1. Architecture overview of Evil-Twin Framework.

The Wi-Fi module ("`AirCommunicator`") was built in order to support a wide range of Wi-Fi capabilities and attacks, therefore the framework identifies three basic pillars of Wi-Fi communication. The three pillars are packet sniffing, custom packet injection and access point creation. Therefore, the three main Wi-Fi communication modules are "`AirScanner`" (packet sniffing), "`AirInjector`" (packet injection) and "`AirHost`" (access point creation). The three modules are wrapped inside the main Wi-Fi module "`AirCommunicator`" which reads the configuration file before starting the services. Any type of Wi-Fi attack can be built using one or a combination of these core features. The ETF was specifically designed to be extended by third party developers, so it would be easy to improve its functionalities.

Since an important part of attacking the Wi-Fi client is performing Man-In-The-Middle (MITM) attacks [14] the framework also has an integrated MITM module called "ETFITM" (Evil-Twin Framework-in-the-middle). Using this module, the security auditor can easily create a web proxy capable of intercepting and manipulating any HTTP/HTTPS traffic.

There are many existing tools that can take advantage of a MITM position or other features of the ETF but, for different reasons, cannot be natively integrated in the framework. Therefore, the ETF includes support for interaction with other tools, without having to integrate them inside the framework. Instead of having to call them separately it is possible to add whatever program to the framework by extending the "`Spawner`" class. Using this functionality a security auditor can call the program with a preconfigured argument string from within the framework, therefore taking advantage of the framework itself but at the same time augmenting its core functionalities.

The other two main ways of extending the framework are through the usage of plugins. The plugins can be divided in two categories, "`Wi-Fi plugins`" and "`MITM plugins`". The "`MITM plugins`" are scripts that run while the MITM proxy is active. The proxy passes the HTTP(S) requests and responses through the plugins' code where they can be logged or manipulated. The "`Wi-Fi plugins`" follow a more complex flow of execution but still expose a fairly simple API to a contributor to develop and use their own plugins. Wi-Fi plugins can be further divided into three categories, one for each of the core Wi-Fi communication modules. Each of the core modules have certain events that trigger the execution of a plugin. For instance, the "`AirScanner`" has three defined events to which a response can be programmed. The events usually correspond to a setup phase before the service starts running, mid-execution phase while the service is running and a teardown or cleanup phase after a service finishes running. Since Python allows for multiple inheritance there can be a single plugin that subclasses more than one of the base plugin classes.

The presented diagram displays a summary of the framework's architecture. Lines pointing away from the "`ConfigurationManager`" mean that these modules read information from it, lines pointing towards it mean that these modules can write/edit configurations.

B. Using and testing the ETF

In order to demonstrate the ETF usefulness and relevance and, at the same time, test it, some specific use-cases were defined. Three different test scenarios with a specific security goal in mind have been designed. Every test will be completed using both ETF and also a combination of other existing open-source Wi-Fi hacking tools. The same Wi-Fi cards were used for every tool on every test. The evaluation criteria will take into consideration the efficiency and the difficulty level to perform the test. Also, a well-defined start and finish time for the tests was also take into consideration. The finish time was considered as a main indicator of efficiency. The easiness of use evaluation considers the amount and complexity of the needed configurations, number of executed commands and the number of open terminals (or foreground programs) needed to finish the task at hand for the security auditor. In this description, the setup commands will include some abbreviations for the access point SSID (APS), access point BSSID (APB), access point channel (APC) and the client MAC address (CM).

TABLE I
RESULTS FROM THE FIRST USE-CASE

	Evil-Twin Framework	**Other tools**
Successful de-authentication attacks	5/5	5/5
WPA handshakes caught	5/5	5/5
Number of open terminals	1	2
Number of setup commands	6	2

For the tests, the three following use cases were considered: a) the capture of a WPA 4-way handshake after a de-authentication attack, b) launching an ARP replay attack and cracking a WEP network and c) launching a catch-all honeypot.

1) Capturing a WPA 4-way handshake after a de-authentication attack: This use-case considers two main aspects: the de-authentication attack and catching a 4-way WPA handshake. It starts with a running WPA/WPA2 enabled access point with one connected client device. The goal is to de-authenticate the client with a general de-authentication attack and then capture the WPA handshake once he tries to reconnect. The reconnection will be done manually right after the device being de-authenticated.

This test is mostly about reliability, and the goal is to find out if the tools are always able to capture the WPA handshake. The test will be performed five times with each tool in order to check its reliability when capturing the WPA handshake. This test will compare the ETF against the combination of the tools "`airodump-ng`" and "`aireplay-ng`".

There is more than one way to capture a WPA handshake using the Evil-Twin Framework. It is possible either to use a combination of the "`AirScanner`" and "`AirInjector`" modules or simply use the "`AirInjector`" with the "`CredentialSniffer`" plugin. For this test, the combination of both modules will be used. ETF launches the "`AirScanner`" module and analyzes the IEEE 802.11 frames to find a WPA handshake. In succession, the "`AirInjector`" can be used to launch a de-authentication attack to force a reconnection.

In the case (Table I) of the other tools, this attack requires at least two open terminals. One terminal has to run the "`airodump-ng`" script which continuously captures packets. "`Aireplay-ng`" launches the de-authentication attack on another terminal.

The ETF was able to perform an efficient and successful de-authentication attack on every test run. The ETF was also able to capture the WPA handshake on every test run. However, it is important to note that this test was performed in an

environment with a medium level of Wi-Fi traffic. It is known that "scapy" may miss packets when there is more traffic, this means one cannot be sure if this efficiency holds up in environments with more traffic.

Using the "airodump-ng" and the "aireplay-ng" tools, the de-authentication attack and the consequent WPA handshake capture were 100% successful on every test.

In conclusion, the attack was performed with 100% success by both tools. However, since the "aircrack-ng" suite is written in the C programming language it is much more efficient in reading packets which probably makes it a better choice for scenarios with high volume of traffic. On the other hand, the ETF configuration commands are very simple and the tool seems to be just as efficient as the "aircrack-ng" tools.

2) Launching an ARP replay attack and cracking a WEP network: This second use-case focus on the efficiency of the ARP replay attack and the speed of capturing the WEP data packets containing the initialization vectors (IVs). The same network may require a different number of caught IVs to be cracked so for this test the limit is 50,000 (fifty thousand) IVs. If during the first test the network is cracked with less than the limit of 50,000 IVs then that number will be the new limit for the second test. The cracking tool to be used will be "aircrack-ng". The test scenario starts with an access point using WEP encryption and an offline client that knows the key - the key is '12345'. Once the client connects to the WEP access point it will send out a gratuitous ARP packet, this is the packet meant to be captured and replayed. The test ends once the limit of packets containing IVs is captured (Table II).

The time starts when the client starts connecting with the WEP access point, this will be done through user interaction. The time stops once the network key is cracked or once the limit of captured packets is reached. This test will again compare the ETF against the combination of "airodump-ng" and "aireplay-ng" tools.

The Evil-Twin Framework uses the Python's "scapy" library for packet sniffing and injection which is rather slow. With some tweaks and usage of lower level "scapy" libraries it was possible to speed up packet injection significantly. For this specific scenario, the ETF uses "tcpdump" as a background process instead of "scapy" for more efficient packet sniffing. However, "scapy" is still used to identify the encrypted ARP packet.

The whole "aircrack-ng" suite is written in C, this makes it very efficient for packet injection and sniffing. This attack requires at least two open terminals. On one terminal one has to run the "airodump-ng" script which continuously captures packets and another terminal running "aireplay-ng" to look for the encrypted ARP packet and replay it.

The ETF correctly identified the encrypted ARP packet and then successfully performed an ARP replay attack which resulted in cracking the network.

TABLE II
RESULTS FROM THE SECOND USE-CASE

	Evil-Twin Framework	**Other tools**
Captured IVs	9958	10000
Execution time	62.2 sec.	36.6 sec.
Number of open terminals	1	2
Number of setup commands	6	2

With the "aircrack-ng" tools the attack was also successful. However, the results show that the tools are more efficient since the time it took it to capture the same amount of traffic was less than the one of the ETF.

This attack is heavily impacted by the packet injection and capture speed. The Evil-Twin Framework uses "scapy" primarily for these tasks which can be disappointingly slow, although with a few tweaks it ends up giving good performance.

The "aircrack-ng" suite on the other hand is very efficient when it comes to packet injection and capture. It is almost twice as fast as "scapy" in sending out packets.

Regarding the number and complexity of the setup commands, ETF needed more configuration commands than the other tools. However, these commands are very simple and the interactive console of ETF provides tab auto-completion and suggestions for a speedier setup. The "airodump-ng" and "aireplay-ng" commands are generally more complex and require copy and pasting of information to fill out the needed parameters.

Both tools showed very good performance, stability and reliability. The "aircrack-ng" tools are however slightly more efficient but the ETF makes up for it by making the setup much easier and just as complete.

3) Launching a catch-all honeypot: This final use-case (Table III) consists of creating multiple access points with the same SSID. By launching multiple access points with all security settings, the client will automatically connect to the one that matches the security settings of the locally cached access point information.

ETF configures the "hostapd" configuration file and then launches the program in the background. "Hostapd" supports launching multiple access points on the same wireless card by configuring virtual interfaces and since it also supports all types of security configurations it is possible to set up a complete catch-all honeypot. For the WEP and WPA(2)-PSK networks a default password is used, for the WPA(2)-EAP an "accept all" policy is configured. This test will again compare the ETF against "airbase-ng".

In order to set up a catch-all honeypot with "airbase-ng" it is necessary to create multiple different virtual interfaces on the same physical card. Then create an access point on each of the newly created virtual interfaces.

The ETF is capable of launching a complete catch-all honeypot with all types of security configurations. The "airbase-ng" script is able to successfully launch access points on non-managed virtual interfaces. The script however does not support the creation of WPA(2)-EAP access points.

TABLE III
RESULTS FROM THE THIRD USE-CASE

	Evil-Twin Framework	**Other tools**
Number of open terminals	1	4
Number of setup commands	4	8

Both tools are able to set up catch-all honeypots although "`airbase-ng`" lacks the support for WPA(2)-EAP networks. Furthermore, when the ETF launches an access point it automatically launches the DHCP and DNS servers which allow clients to stay connected and use the internet. To do this with "`airbase-ng`" a lot more commands and configurations would have been necessary. Alternatively, it is possible to use "`hostapd`" to set up the catch-all honeypot manually but the configuration file can be complicated to write, especially when setting up multiple access points on the same interface.

Ultimately the ETF offers a better, faster and more complete solution to create catch-all honeypots.

IV. CONCLUSIONS AND FUTURE WORK

ETF was developed as an open-source project and it is available as a Github project (https://github.com/Esser420/EvilTwinFramework/).
Therefore, contribution to the development of the framework is open to the community and continuous improvement is always present, so more attacks and plugins will be added by external developers. Results obtained from the tests conducted validate ETF capabilities of performing well-known attacks on Wi-Fi networks and clients. The results also validate that the architecture of the framework enables the development of new attacks and features on top of it while taking advantage of the pre-existing capabilities that the platform provides. This reality will accelerate future development of new Wi-Fi pentesting tools since a lot of the code is already provided by the ETF open-source project. Furthermore, the fact that complementary Wi-Fi technologies are all integrated in one tool will make Wi-Fi penetration testing simpler, more efficient and facilitate the job of the security auditor.

Even though the performance of the ETF, in some cases, was less efficient than the tools it was compared against on the "ARP replay attack" test case, it still finished the test in little over a minute which is still considered to be very efficient. When taking all the other tests cases into account, the ETF did not show any drawbacks in efficiency nor reliability. These tests however are not conclusive since more experimentation is needed, especially in more demanding and realistic environments.

The tool does not yet completely replace all the other tools because some capabilities, such as attacks on WPS, are not yet implemented. However, this is meant to change in the future, due to the open nature of the framework, that allows anyone to implement and contribute with such functionalities.

One of the limitations of Wi-Fi auditing nowadays is the lack of ability to log important events during the tests. This makes reporting of the found vulnerabilities a lot more difficult and also less accurate. The framework could implement a logger that can be accessed by every class in order to produce a report of a penetration testing session.

Yet another very useful contribution would be adding a database to the framework. This database could hold information about vulnerabilities in known routers such as default WPA key and WPS pin generation algorithms. This increases the chance of discovering hidden Wi-Fi vulnerabilities and therefore the efficiency of the security testing.

ETF covers many aspects of Wi-Fi testing facilitating the phases of Wi-Fi reconnaissance, vulnerability discovery and attack. Also, the tools do not offer any feature that facilitates the reporting phase. Adding the concept of session and a session reporting feature, such as the logging of important events during a session, would greatly increase the value of the tool for real pentesting scenarios.

Another valuable contribution would be extending the framework to facilitate Wi-Fi fuzzing. The IEEE 802.11 protocol is very complex and considering there are multiple implementations of it, both on the client and access point side, one can safely assume that these implementations contain bugs and even security flaws. These bugs could be discovered by fuzzing IEEE 802.11 protocol frames. Since "`scapy`" allows for custom packet creation and injection a fuzzer can be implemented through it.

V. ACKNOWLEDGEMENTS

The authors would like to acknowledge the FCT Project UID/MULTI/4466/2016.

REFERENCES

[1] Marco Domenico Aime, Giorgio Calandriello, and Antonio Lioy, "Dependability in wireless networks: Can we rely on wifi?", *IEEE Security & Privacy*, vol. 5, no. 1, 2007.

[2] Bernhard H Walke, Stefan Mangold, and Lars Berlemann, *IEEE 802 wireless systems: protocols, multi-hop mesh/relaying, performance and spectrum coexistence*, John Wiley & Sons, 2007.

[3] Lisa Phifer, "Anatomy of a wireless" evil twin" attack", *Retrieved July*, vol. 22, 2007.

[4] Sergey Bratus, Cory Cornelius, David Kotz, and Daniel Peebles, "Active behavioral fingerprinting of wireless devices", in *Proceedings of the first ACM conference on Wireless network security*. ACM, 2008, pp. 56–61.

[5] Charlie Miller, "Mobile attacks and defense", *IEEE Security & Privacy*, vol. 9, no. 4, pp. 68–70, 2011.

[6] Michel Barbeau, Jyanthi Hall, and Evangelos Kranakis, "Detecting impersonation attacks in future wireless and mobile networks", in *Secure Mobile Ad-hoc Networks and Sensors*, pp. 80–95. Springer, 2006.

[7] Liran Ma, Amin Y Teymorian, and Xiuzhen Cheng, "A hybrid rogue access point protection framework for commodity wi-fi networks", in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008, pp. 1220–1228.

[8] Somayeh Nikbakhsh, Azizah Bt Abdul Manaf, Mazdak Zamani, and Maziar Janbeglou, "A novel approach for rogue access point detection on the client-side", in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. IEEE, 2012, pp. 684–687.

[9] Ninki Hermaduanti and Imam Riadi, "Automation framework for rogue access point mitigation in ieee 802.1 x-based wlan", *Journal of Theoretical and Applied Information Technology*, vol. 93, no. 2, pp. 287, 2016.

[10] Songrit Srilasak, Kitti Wongthavarawat, and Anan Phonphoem, "Integrated wireless rogue access point detection and counterattack system", in *Information Security and Assurance, 2008. ISA 2008. International Conference on*. IEEE, 2008, pp. 326–331.

[11] "The Penetration Testing Execution Standard", 2012.
[12] Chuck Easttom, "A model for penetration testing", 2014.
[13] Guido Van Rossum et al., "Python programming language.", in *USENIX Annual Technical Conference*, 2007, vol. 41, p. 36.
[14] Hyunuk Hwang, Gyeok Jung, Kiwook Sohn, and Sangseo Park, "A study on mitm (man in the middle) vulnerability in wireless network using 802.1 x and eap", in *Information Science and Security, 2008. ICISS. International Conference on*. IEEE, 2008, pp. 164–170.