Undesired Relatives: Protection Mechanisms Against The Evil Twin Attack in IEEE 802.11

Fabian Lanze
Interdisciplinary Centre for Security, Reliablity and Trust
University of Luxembourg
fabian.lanze@uni.lu

Andriy Panchenko
Interdisciplinary Centre for Security, Reliablity and Trust
University of Luxembourg
andriy.panchenko@uni.lu

Ignacio Ponce-Alcaide
Escuela Técnica Superior de Ingeniería Informática
University of Málaga
iponce@uma.es

Thomas Engel
Interdisciplinary Centre for Security, Reliablity and Trust
University of Luxembourg
thomas.engel@uni.lu

ABSTRACT

Commonly used identifiers for IEEE 802.11 access points (APs), such as network name (SSID), MAC (BSSID), or IP address can be trivially spoofed. Impersonating existing APs with faked ones to attract their traffic is referred to in the literature as the *evil twin attack*. It allows an attacker with little effort and expenditure to fake a genuine AP and intercept, collect, or alter (potentially even encrypted) data. Due to its severity, the topic has gained remarkable research interest in the past decade. In this paper, we introduce a differentiated attacker model to express the attack in all its facets. We propose a taxonomy for classifying and structuring countermeasures and apply it to existing approaches. We are the first to conduct a comprehensive survey in this domain to reveal the potential and the limits of state-of-the-art solutions. Our study discloses an important attack scenario which has not been addressed so far, i.e., the usage of specialized software to mount the attack. We propose and experimentally validate a novel method to detect evil twin APs operated by software within a few seconds.

Categories and Subject Descriptors

C.2.0 [**COMPUTER-COMMUNICATION NETWORKS**]: General—*Security and protection*; C.2.3 [**COMPUTER-COMMUNICATION NETWORKS**]: Network Operations—*Public networks*

Keywords

Evil Twin Attack; Security; Wireless Access Point; Fake Access Point; 802.11; Survey; Rogue Access Point; Software-based Access Point

Q2SWinet'14, September 21–26, 2014, Montreal, QC, Canada.

http://dx.doi.org/10.1145/2642687.2642691.

1. INTRODUCTION

In recent years, Internet usage shifted from stationary to mobile devices such as laptops, tablets, or smartphones with a wireless connection to the network. Nowadays, people are used to being online all the time, independent of their location. Wi-Fi access points offer fast and cost-effective Internet connectivity. They are available almost everywhere: in offices, on university campuses, and in public places such as cafés, shopping malls, hotels, or airports. Although mobile cellular networks (e. g., 3G) have gained an increasing influence, the importance of Wi-Fi networks remains crucial. Generally, they provide faster connectivity, offer service whenever mobile networks are unavailable, overloaded, or overpriced (e. g., in roaming). They are indispensable for devices that do not have hardware to access mobile cellular networks, e. g., laptops or many tablets.

The only identifiers provided by the IEEE 802.11 standard for a user to verify the authenticity of an Access Point (AP) are its network name (SSID) and its MAC address (BSSID). Since these can easily be spoofed, an attacker can fake an AP without the user being able to notice. Commonly, this attack is referred to as the *evil twin attack* but other terminology is also used, such as *rogue AP* or *spoofed AP*.

Once a user has inadvertently connected to a spoofed AP, the attacker can mount various attacks, including interception, collection, or manipulation of transmitted data. This remains possible even if the user explicitly uses encryption, e. g., by enabling SSL. Since the attacker has already established his AP as intermediary, he can easily act as a man in the middle. Nowadays, this does not require special skills, as deployable tools such as SSLstrip[1] (which transparently removes SSL encryption) and BurpProxy[2] (which can create faked certificates on the fly) are freely available and easy to use. Given that most users incautiously accept unsigned or wrongly-signed SSL certificates [25, 6], malicious APs are able to conduct man-in-the-middle attacks on encrypted traffic (i. e., can read and modify the data) and to hijack sessions.

The danger of the described attacks arises from their simplicity: all common mobile operating systems including An-

[1] `http://www.thoughtcrime.org/software/sslstrip/`
[2] `http://portswigger.net/burp/proxy.html`

droid and iOS are capable of creating a wireless AP. Hence, this process can be performed directly from smartphones, without attracting the attention of anybody in the vicinity. Additionally, fully-automated tools are available that are capable of spoofing APs, e. g., *rfakeap*[3], or the *aircrack-ng*[4] suite.

Due to its severity, the attack has gained a remarkable amount of interest in the media and research community, and much work has been done on solving the problem. Nevertheless, the papers in this area do not follow a consistent direction. Most only target particular aspects of the attack and do not solve the problem as a whole. The solutions proposed so far have different limitations regarding requirements, ease of deployment, attacker model, and detection efficacy. Moreover, there is no consistent terminology or classification, hence, central concepts such as *fingerprinting* and *fake detection* are mistakenly confused. This makes existing work difficult to compare.

Contribution

Our contribution is twofold:
(*i*) We provide a clear definition of different attack scenarios for evil twin attacks in 802.11 network environments. We then define a taxonomy for existing countermeasures. We apply our taxonomy to classify proposed solutions and reveal what can already be achieved, what differences between the solutions exist, and what limitations remain. We show which problems are still unsolved to point researchers to future research directions and to enable them to integrate their work into this complex field. Our survey discloses several facets of the evil twin attack for which no reliable protection exists.
(*ii*) We tackle an important issue raised by the survey, i.e., the likely scenario in which an attacker utilizes sophisticated software tools to mount the attack, in particular, the popular *aircrack-ng* suite. We propose a novel method to detect evil twins set up using such software which exploits accuracy flaws that are introduced through the required emulation of hardware behavior. We experimentally validate that this method reliably separates software access points from genuine hardware APs.

2. ATTACK SCENARIOS

To overcome inconsistencies regarding terminology in related work we recapitulate the most commonly used definitions: an **evil twin** is a hard- or software-based 802.11 AP that spoofs the identity of a legitimate AP by cloning its characteristics in order to trap a user to hijack his connection. The terms *evil twin*, *rogue AP*, *spoofed AP* and *fake AP* are used synonymously in the related work. Apart from this, the term **rogue AP** is also often used to describe the setup and integration of an AP without permission in an enterprise network, leading to a different type of threat. In order to avoid confusion we will refer to this scenario as an **unauthorized AP** in the following.

For the evil twin attack we propose to differentiate the following scenarios:
1. Replacement: The legitimate AP is switched off and replaced by the evil twin at the same location.
2. Coexistence: Both legitimate AP and evil twin coexist at the same location. The Fake AP tries to capture users, e.g., by providing a higher signal strength. Furthermore, we distinguish between evil twins with **(a) their own Internet connection** and **(b) routing via the legitimate AP**.
3. Remote clone: The evil twin is set up at a different location. If a profile for the legitimate AP exists, the client device will automatically connect to the faked AP.
4. Ad hoc clone: The attacker listens for probe requests, which clients use to probe for networks which they have used in the past. He then provides an evil twin matching one of the requested profiles. This can be done, e.g., using *airbase-ng* (see Section 5).

We explicitly do not consider the situation where the attacker takes over the legitimate AP and routes its traffic via his own device/network (physical intrusion into the system).

3. CLASSIFICATION OF COUNTERMEASURES

Existing solutions to protect against the evil twin attack differ both concerning which of the attack scenarios they aim to detect, and in whether they rely on special hardware or protocol modifications, whether they can be detected and so on. We identify and propose the following categories for classification:
Device fingerprinting vs. evil twin detection: A device fingerprint is a set of (remotely observable) attributes that is as unique as possible to an AP. In contrast, a method for evil twin detection merely makes it possible to distinguish between the scenario where an attacker is mounting the attack and the legitimate network. Obviously, a fingerprinting method is implicitly an evil twin detection method but not vice versa.
Single-AP vs. Group-of-APs: While a single-AP approach treats an AP as isolated device, a group-of-AP method includes additional devices in the evaluation, e.g., the set of simultaneously-reachable APs.
Client vs. Operator: Either the client tries to detect whether he is connecting to a (potential) evil twin, or the network operator runs a system to detect evil twins within his administrative domain.
Active vs. passive detection: Active methods interact with an AP, e.g., by sending packets, while passive methods just observe traffic. Hence, in general, active methods can be detected, may require cooperation of the AP and may interfere with regular communication while passive methods by design exhibit none of these drawbacks.
Commodity vs. specialized hardware: For certain methods (e.g., radio frequency fingerprinting), specialized hardware (such as a dedicated radio device) is essential, while other methods operate on unmodified commodity hardware such as laptops or smartphones.
With vs. without deviation from standard protocols by AP: Several methods require the standard protocols implemented in 802.11 APs to be modified or extended, e.g., to enable additional cryptographic primitives or to provide additional parameters used for identification.
Single-entity vs. group-based: A single-entity method can be performed by one client device while a group-based method requires anything from one additional collaborating device (e.g., a second laptop/smartphone to perform measurements) up to a large set (e.g., in a crowd-sourced approach).

[3] `http://rfakeap.tuxfamily.org/`
[4] `http://www.aircrack-ng.org/`

Software- vs. hardware-based AP: Since most commonly the evil twin attack is performed by software (e.g., *airbase-ng*), countermeasures could exploit this fact by investigating software-specific characteristics to identify such attacks.
Ad hoc vs. pre-gathered information: Most methods require pre-gathered information (e.g., a fingerprint of an AP), whereas a few provide solutions in environments that have not been recorded before.

In summary, the perfect protection method against the evil twin attack would allow unique remote device fingerprinting for single APs, be passive, operate on unmodified commodity hardware, would not require any protocol modification and can be performed by a single entity. Unfortunately, so far no solution meeting all these requirements exists. Therefore, in the following, we will analyze existing countermeasures with respect to our proposed taxonomy to reveal what trade-offs must be accepted and which directions offer potential for further research.

4. SOLUTIONS

In this section we first describe existing out-of-the-box solutions such as Wi-Fi Protected Access (WPA), 802.1X, Virtual Private Networks (VPN), and web-based authentication to show why they are not appropriate for protecting against evil twin attacks. We then direct our focus on methods proposed by the research community in the past years and thoroughly analyze them.

WPA-Personal (PSK) / 802.11i: For WPA(2), a pre-shared key (PSK) is established to encrypt traffic between client and AP. Such a mechanism can only protect against the evil twin attack if the PSK is concealed from the attacker. Note, however, that in the case of public hotspots the PSK has to be distributed to users by some means or another, e.g., printed on a receipt. Therefore, the attacker can acquire the key in the same way as regular customers would do – and mount the attack unimpeded.

WPA(2)-Enterprise / 802.1X: In a WPA-Enterprise setup, the wireless AP acts as authenticator between a client (called supplicant) and an authentication server using Remote Authentication Dial In User Service (RADIUS) [12] and the Extensible Authentication Protocol (EAP). In the most commonly used standards, EAP-MSCHAPv2 and EAP-TLS, a CA certificate should be used by clients to authenticate the server before submitting credentials. Theoretically, evil twin attacks are rendered impossible by this setup since the attacker cannot trivially fake the authentication server, as it is protected by strong cryptographic means. Nevertheless, this solution has a major weakness in practice: the mechanism has to be carefully set up and maintained by the operator and operators of public hotspots in particular have no incentive to provide such a service. Moreover, the key component of the authentication process, i.e., the validation of the server certificate by the client, is optional. If this is not done carefully by the user (i.e., the certificate check is activated and the user rejects the connection on seeing a certificate warning) it is possible to fake the authentication server, e.g., by harvesting and cracking handshakes [19, 20].

Web-based authentication: Public hotspots deployed in hotels, cafés, or at airports commonly provide web-based authentication. Usually, the first connection initiated by the user is redirected to a *captive portal*, a website hosted by the operator that provides a disclaimer and requests credentials (or credit card information) for accounting. However, the attacker can simply clone such a page and in addition collect information entered by the user into such a web-based form (e.g., credit card number) for further misuse. Note that this task can in fact be automated using, e.g., *airsnarf*[5]. The goal of a web-based authentication at hotspots is to authenticate the user and not the hotspot or its provider. Hence, this method does not provide any security at all for the user with respect to the evil twin attack.

Virtual Private Networks (VPN): Whenever there is a need to connect to the Internet through a potentially untrustworthy operator, VPNs appear on the scene. Nevertheless, the protection provided is not satisfactory, especially in operating systems for mobile devices (e.g., Android, iOS). Besides certificate-based attacks such as those on SSL, it is possible for the attacker to terminate the VPN session (e.g., by dropping management packets) such that the connection falls back to plain mode – typically without an explicit notification to the user.

Since the described solutions fail to protect the user against evil twins, a variety of approaches has been proposed by the research community. In the following we apply our proposed classification regarding the attack (see Section 2) and the countermeasures (see Section 3) to the related work. We focus on approaches that target either the specific detection of the evil twin attack or remote device fingerprinting (which, if possible, can be directly used to detect evil twins). Hence, we disregard approaches that only solve a minor part of these problems such as [7], where only the device driver or [8], where only the device type is fingerprinted and not the unique device (i.e., two devices of the same series get the same fingerprint). Methods for detecting unauthorized access points that do not clone the identities of existing ones (see Section 2) are out of scope of this paper.

An overview on the relevant related work is presented in Table 1. We divide the approaches into three major categories, those requiring *Protocol Modifications*, performing *Hardware Fingerprinting* and *Non-Hardware* based methods. We subcategorize whenever appropriate and reasonable.

4.1 Protocol Modifications

We identified two methods that propose a protection against the evil twin attack by modifying deployed protocols: *Secure Open Wireless Access* (SOWA) [5] and *Simple Wireless Authentication Technique* (EAP-SWAT) [2, 9]. These methods transfer well-known concepts such as SSL or SSH to the 802.11 scenario. In SOWA, SSIDs are unique domain-like strings that are tied to a certificate in a manner similar to SSL/TLS by a Certification Authority (CA). These can authenticate the operator of an AP. EAP-SWAT adapts the trust-on-first-use behavior of SSH connections. If the trust relationship can be ensured for the first connection to an AP, the authenticity of the AP can be guaranteed for all following connections with the help of certificates. Since both methods rely on adapted protocols, they cannot easily be deployed as the drivers and firmware of all the clients *and* APs need to be changed.

4.2 Hardware Fingerprinting

The second class focuses on hardware characteristics to uniquely identify the specific device on which an 802.11 AP

[5] `http://airsnarf.shmoo.com/`

Table 1: Overview on different approaches protecting against the evil twin attack

		Approach	Source	Attack (**R**eplace/**C**oexist/**RE**mote/All*)	Device **F**ingerprint / Evil-Twin-**D**etection	**S**ingle-AP / **G**roup	**A**ctive / **P**assive	**C**ommodity / **S**pecial Hardware	Protocol Adaption (**Y**/**N**)	**S**ingle-Entity / **G**roup / **C**rowd	**C**lient / **O**perator	**A**d hoc / pre-gathered **I**nformation	Accuracy[a]	Dataset Size
Prot.		EAP-SWAT (2008)	[2, 9]	*	D	S	A	C	Y	C	C	I	100%	n.a.
		Secure Open Wireless Access (2011)	[5]	*	D	S	A	C	Y	S	C	I	n.a.	n.a.
Non-Hardware	Evil Hop	RTT-to-DNS-Server[b] (2009)	[10]	C	D	S	A	C	N	S	C	I	n.a.	2
		ETSniffer[b] (Multi-hop-detection) (2010)	[24]	C	D	S	A	C	N	S	C	A	TPR=.99 FPR=.1	n.a.
		WiFiHop[b] (Watermark detection) (2011)	[18]	C	D	S	A	C	N	S	C	A	TPR=.98 FPR=.001	n.a.
		Multiple Signal Detection[b] (2012)	[14]	C	D	G	P	C	N	S	C	A	TPR=.99 FPR=.001	n.a.
	Device	Authentication timing + SVM[c] (2006)	[23]	*	F	S	A	C	N	G	C	I	86%	5
		Active Behavioral Fingerprinting (2008)	[3]	*	F	S	A	C	N	S	C	I	n.a.	5
		Histogram of frame arrival times (2012)	[21]	*	F	S	P	C	N	S	C	I	TPR=.566 FPR=.1	188
	Env.	Centralized hybrid monitoring[c] (2008)	[17]	*	D	S	A	C	Y	S	O	I	n.a.	n.a.
		Context Leashing (2008)	[2, 9]	E	D	G	P	C	N	S	C	I	TPR=.82 FPR=.1	n.a.
Hardware	RFF	Radio Frequency Fingerprinting (2008)	[4, 22]	*	F	S	P	S	N	S	C	I	99%	130
	Clock Skew	Clock Skew[d] (2008)	[13]	*	F	S	P	C	N	S	O	I	n.a.	41
		Clock Skew[d] (2010)	[1]	*	F	S	P	C	N	S	C	I	n.a.	2
		Clock Skew (2012)	[16]	*	F	S	P	C	N	S	C	I	n.a.	388
		Clock Skew+Temperature (2014)	[15]	*	D	G	P	C	N	S/C	C	I	TPR=.9 FPR=.1	12

[a] TPR/FPR: True/False Positive Rate
[b] Evil hop methods assume that the faked AP has a wireless connection to the legitimate AP
[c] Requires additional monitoring device
[d] Requires modified driver

is implemented. Hence, the approaches described here rely on a central authority to collect/manage fingerprints. This constitutes a single point of failure/trust. Within this category, we differentiate between Radio Frequency Fingerprinting (RFF) and clock skew methods.

RFF-based methods aim to identify radiometric signal properties of wireless devices either in the waveform or the modulation domain. We omit a detailed description here and include two representative examples in Table 1. Although such methods can achieve extremely high precision in evil twin detection, they all exhibit a critical limitation: they rely on dedicated specialized hardware for measurement and thus cannot be performed by regular devices such as laptops or smartphones.

Several approaches utilize an unavoidable physical phenomenon called **clock skew** which causes crystal oscillator-based clocks to have tiny yet observable deviations in speed. Clock skew fingerprinting was introduced in the 802.11 domain by Jana and Kasera [13] by extracting Timing Synchronization Function (TSF) timestamps from beacon frames, and improved by Arackaparambil et al. [1]. These methods were rather experimental and not practically feasible due to severe limitations (modified drivers required and fingerprints are not reproducible between different fingerprinting devices). Moreover, the studies were conducted on a small set of devices. Therefore, our own work thoroughly investigated the practicability of TSF clock skew-based fingerprinting. We were the first to propose a lightweight fingerprinter-independent method for measuring TSF clock skews with unmodified commodity hard- and software with arbitrary precision and to analyze its effectiveness for 388 APs 'in the wild' [16]. Our evaluation revealed that TSF clock skews do not provide enough information content to uniquely identify 802.11 APs. We enhanced the methodology by combining clock skew with device-intrinsic temperature-dependency [15]. This led to a method that detects evil twins with high probability if (a) at least 2 different APs are receivable in the same area and (b) this set has been observed for an appropriate duration. Although this approach satisfies most crucial requirements (i.e., it is passive, detects all attack types and does not need any modifications to the AP's deployment) the results were produced in a laboratory environment and lack a large-scale evaluation.

4.3 Non-Hardware based Identification

In the third class we include all approaches that investigate the behavior of devices, certain network properties or the environment of an AP in order to detect an evil twin.

One set of methods, which we call *evil hop detection*, aims to establish whether the presence of an evil twin introduces an additional wireless hop on each route. All these methods assume that the faked AP has a wireless connection to the legitimate AP in order to offer its Internet connection to clients. The goal is to detect this extra hop on the path. In [10] this is achieved by measuring the RTT to a local DNS server, assuming the faked AP is also relaying DNS queries to it. Song et al. [24] modify the client's driver and measure the inter-packet arrival time to reliably distinguish between a one-hop and a two-hop wireless channel. However, a legitimate hotspot connected via a wireless link with the Internet would also be classified as evil twin by this method. Mónica and Riberio propose a method called WiFiHop [18]. They actively induce watermarked packets in order to detect whether these are relayed on a different wireless channel. Note that this approach and the approach of Song et. al work ad hoc, i.e., without access to any pregathered information. The authors of [14] assume that the attacker is running several SSIDs on the same device and try to detect this by correlating received signal strengths. Although all mentioned approaches for evil hop detection claim to achieve a very high detection accuracy in terms of the true positive rate (see Table 1), the practical use of these methods is very limited as they solve only a part of the problem: they only detect the coexistence of an evil twin in situations where the faked AP routes traffic through the legitimate AP (i.e., scenario 2(b) in our attacker model, see Section 2), is connected via a wireless link to the Internet (which is not necessary a sign of a faked AP), or work in the situations where the attacker runs several SSIDs on the same device (which is also not always a sign of a faked AP).

The second group of non-hardware-based approaches attempts to fingerprint behavioral characteristics of the AP. Sieka [23] evaluates precise measurements of timings related to the authentication procedure and trains a Support Vector Machine (SVM) to recognize these fingerprints. The approach achieves an accuracy of 86% but is evaluated only for a non-representative dataset of 5 APs. Moreover, this technique requires a second device for monitoring all authentication sequences. A similar method is evaluated by Neumann et al. [21]. The authors use frame inter-arrival times as a feature for a histogram-based classifier that can be operated by single entities. On a more representative dataset of 188 APs their detection rate drops to 50-60%. Such a rate is insufficient for practical purposes. Bratus et al. [3] propose *active behavioral fingerprinting*, a technique inspired by TCP/IP stack fingerprinting as performed by tools like *nmap*. It utilizes *malformed stimuli-response*, i.e., how devices react to manipulated or fragmented frames such as probe requests. Though these methods may interfere with regular communication, we see remarkable potential in such a method, particularly for identifying characteristics of software-based APs. Still, a thorough evaluation has not been conducted so far.

The last set of approaches does not focus on the AP as an isolated device but rather on its environment, i.e., the group of simultaneously-reachable APs. Gonzales et al. [9] propose *context-leashing*. A context is defined as a set of tuples containing SSID and RSSI for all APs visible from a certain location. An AP is considered suspicious when its context has significantly changed. Recalling our attacker model (Section 2), such a method can only work for the remote clone scenario, i.e., when the attacker sets up the evil twin at a different location from the legitimate AP. In [17], a hybrid framework is introduced. It consists of several components that fulfill different tasks, both centralized and distributed. A frame collector continuously investigates the traffic, searching for anomalies, while a detection module also performs active probing of devices. Note that in this monitoring framework, the evil twin attack is only one possibility that can be detected among various other threats such as improperly configured devices. Moreover, the whole mechanism involves significant additional efforts for network operators and cannot be performed by single clients.

As our analysis reveals, none of the existing approaches explicitly deals with attack scenario 4 (ad hoc clone). Additionally, the methods do not exploit the fact that an evil twin is commonly set up using software tools. We see a remarkable necessity to analyze and detect such a case. It is likely that an attacker will not expend additional effort to replace or clone a legitimate AP with a physical 802.11 hardware device if he is equipped with a piece of software that is capable of doing the same. Besides the lower effort, using software on a mobile device is obviously creating less attention, thereby, reducing the risk for the attacker of being discovered and sentenced. We expect such a tool to leak additional information or to show distinct characteristics compared to hardware access points that enable a thorough detection. To the best of our knowledge, we are the first to propose a method for the explicit detection of software-based evil twin attacks.

5. FILLING THE GAP: DETECTION OF SOFTWARE APS

In this section we propose and experimentally evaluate a novel method for detecting the evil twin attack in the likely scenario that the attacker sets up the fake AP with software running on his mobile device, e.g., a laptop. We focus our evaluation on the *aircrack-ng* suite. This is a set of actively developed and maintained tools which implement a variety of known attacks. It is used for auditing the security of 802.11 networks. *aircrack-ng* contains, among others, a tool called *airbase-ng* that is intended to act as software-based AP aiming at attacking clients in particular. It is able to act as fully operational access point, to manipulate and resend packets, to capture WPA(2) handshakes (see Section 4) and it implements a strategy that we defined as ad hoc clone attack (see Section 2). Finally, running on a variety of different Wi-Fi chipsets and being available as Live-CD[6], *aircrack-ng* can be considered as the ultimate, fully-featured toolbox that attackers could use out-of-the-box to mount all variants of the evil twin attack. To the best of our knowledge, it is the only available software to automate this task. Several other known tools, such as *karmetasploit*[7] or *Katalina*[8] implement additional components to exploit client vulnera-

[6] *aircrack-ng* is also part of the popular penetration testing operating systems Backtrack Linux and Kali Linux

[7] `http://www.wirelessdefence.org/Contents/karmetasploit.htm`

[8] `https://github.com/kussic/Katalina`

bilities or to facilitate the operation of evil twins, but are internally based on *airbase-ng*.

For detecting such an attack, we strive to extrapolate a characteristic that uniquely distinguishes an AP operated with *airbase-ng* from one running on dedicated hardware. Concretely, we exploit the fact, that when software needs to emulate the operational behavior of hardware, it is prone to a loss of accuracy due to processing delays. More specifically, we will explain why *airbase-ng* fails to perfectly imitate a hardware AP regarding the accuracy of Timing Synchronization Function (TSF) timestamps in beacon frames and we will present a method how to detect this.

In an 802.11 network in infrastructure mode, the access point serves as timing master for all associated stations. To this end, all stations maintain a TSF timer with microsecond granularity and a modulus of 2^{64}. The AP periodically (typically every 100ms) sends beacon frames containing a timestamp of its TSF timer and all stations update their local TSF timer to the value in a received beacon whenever this value is later than their local TSF timer. Thus, all timers within a basic service set are synchronized to that of the AP. Since TSF is used for high-precision operations such as synchronized hopping of frequencies or power management, the 802.11 standard specifies very strict accuracy requirements. In detail, an AP should set a beacon frame's timestamp field such that it equals the value of the AP's TSF timer at exactly that moment, when the data symbol containing the first bit of the timestamp is transmitted to the physical layer considering even minuscule delays caused, e.g., by the hardware layout or the antenna [11]. Genuine APs benefit from an optimized combination of hardware and firmware to comply with the specification – this fact enabled the derivation of the TSF timer's clock skew from a sample of timestamps extracted from beacon frames, see [13, 1, 15, 16]. Since *airbase-ng* needs to emulate this functionality to act as access point, we reviewed its source code to reveal if and why a reduction in accuracy is to be expected. In fact, *airbase-ng*, which is written in C, generates beacons within a specific thread. After the beacon interval has expired, it allocates a new packet by writing the required fields directly into the memory. The timestamp is acquired by a *gettimeofday()* system call, which provides timing information in the required microsecond resolution. This timestamp is generated first, then written into the packet byte by byte *at the end* of the crafting process *before* the beacon's delivery is delegated to the driver. Therefore, when the packet is actually sent by the network interface, its timestamp is already outdated compared to the specification and should exhibit a significant delay. This deviation is to some extent unpredictable as it depends, among other factors, on the system's processing delay.

5.1 Experimental Setup

To validate our analysis we performed the following experiments. We used four standard laptops running different distributions of Linux. Details, including the equipped Wi-Fi hardware and the driver used are shown in Table 2.

We set up *airbase-ng* on all laptops as an attacker would do to perform the evil twin attack. Then we sniffed beacon frames from 30 different hardware access points as well as from the four software APs using the Python library *scapy*[9]. From each beacon frame we extracted the TSF timestamp

[9] `http://www.secdev.org/projects/scapy/`

	OS	Wi-Fi chipset	Driver
Host A	Ubuntu 12.04	Atheros5002X	ath5k
Host B	Ubuntu 10.10	Broadcom BCM4312 a/b/g	b43
Host C	Backtrack RC5	Intel Pro/Wireless 3945ABG	iwl3945
Host D	Ubuntu 12.10	Broadcom BCM4313 b/g/n	b43

Table 2: Laptops used for our experiments

t_{TSF} and the corresponding receiving time t_{REC} to calculate the offset $o = t_{TSF} - t_{REC}$. For n received beacons, we obtain a trace $T = \{(t_{REC_1}, o_1), ..., (t_{REC_n}, o_n)\}$.

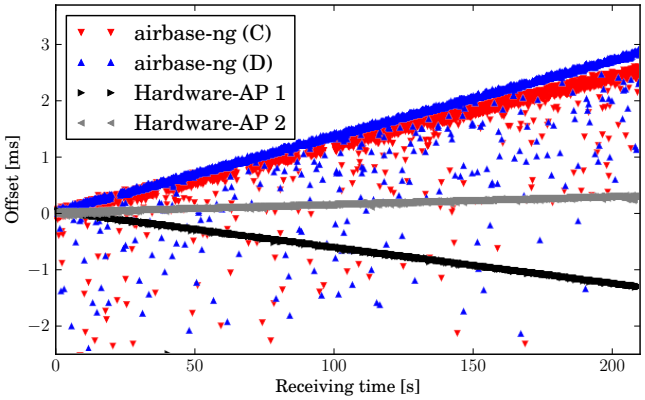

Figure 1: TSF accuracy of *airbase-ng* APs compared to Hardware APs

Figure 1 shows four exemplary traces from two hardware APs and two *airbase-ng* APs. It clearly validates our assumption on the lower TSF accuracy of the software-based AP. While the traces derived from hardware APs form a perfect straight line, those from *airbase-ng* exhibit a significantly higher scattering as well as a large number of outliers – rendering the two types clearly distinguishable. We will now propose and evaluate a method to reliably detect *airbase-ng* access points based on this observation.

5.2 Detection Method

To identify traces produced by *airbase-ng*, we propose to use the *root-mean-square error (RMSE)* of an ordinary least squares regression (LSR) fitted into the (x, y)-points of a trace. Formally, let $P^T_{LSR}(x) = \alpha x + \beta$ be the prediction function obtained by ordinary least squares regression for a trace T. Then, the RMSE of a trace with length $|T| = n$ is given by

$$\mathrm{RMSE}(T) = \sqrt{\frac{\sum_{i=1}^{n} (P^T_{LSR}(x_i) - y_i)^2}{n}}.$$

The RMSE represents the standard deviation of residuals and, therefore, serves as intuitive metric of accuracy. Note that α, the slope of P^T_{LSR}, is an estimation of the subjective clock skew of the access point's TSF timer and the measuring device's system clock (see [15] for further details). While such a skew could be used to fingerprint devices, here we are more interested in identifying behavioral deviations of software vs. hardware. Hence, the RMSE is the preferable metric as it is independent of the slope of the prediction function and thus, by design independent of the device performing the measurement.

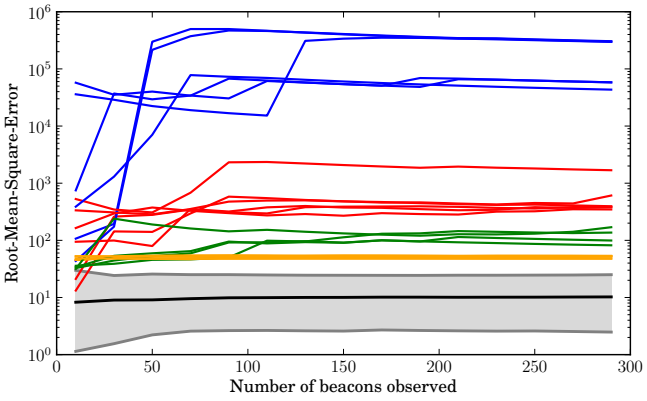

Figure 2: Root-Mean-Square-Error of TSF accuracy (logarithmic scale) for *airbase-ng* hosted by Host A (orange), Host B (green), Host C (red), Host D (blue) and for 30 hardware APs (grey area)

5.3 Evaluation

We gathered traces with a duration of one minute each from the 30 different hardware APs with the four laptops. Additionally, we set up each laptop at a time hosting *airbase-ng* while the remaining three performed the measurement at two different times, resulting in 6 traces per host. From all these traces we calculated the respective RMSE for subsequences of varying lengths (simulating sniffs with a shorter duration). The results are plotted in Figure 2, where the RMSE is scaled logarithmically. The black line represents the mean of all hardware APs, the grey lines the respective minimum and maximum. Thus, the RMSE of all hardware APs fall into the grey area. The colored lines show the values obtained for *airbase-ng*. The overall result is unambiguous: all software-based APs exhibit an RMSE that is larger by several orders of magnitude than all hardware APs with values between $2.65 - 26.45$ (hardware) and $48.06 - 3.44 \times 10^5$ (software) for input data with μs-resolution. The error intensity depends on the laptop hosting *airbase-ng*, the TSF accuracy of Host D and Host B for example differs by a magnitude of 10^3. Nevertheless, in our dataset even the 'worst' hardware AP is still almost twice as accurate as the 'best' software AP regarding this accuracy metric. The second important observation is that the metric stabilizes already for sample sizes of less than 100 beacons. Recalling the fact that beacon frames are typically sent every 100ms, the identification of an AP operated by *airbase-ng* can be done within a few seconds with our approach.

5.4 Discussion

Although the presented results reveal striking potential for identifying evil twins operated by software, some limitations of our evaluation should be noted. First, there is a need for a large-scale evaluation. Therefore, we draw no final conclusions and, concretely, do not extrapolate a threshold value for the RMSE from our data that allows decisive distinction. Second, we used four customary laptops for hosting the software AP. It cannot be excluded that hardware exists, which exhibits a higher precision, or that the sending process of beacon frames cannot be improved to be less prone to system delays. Additionally, it would be interesting to investigate other mobile platforms hosting the access point, e.g., tablets or even smartphones. Third, we limited our experiments to one specific software tool, i.e., *airbase-ng*. Still, we argue that this choice is reasonable since the *aircrack-ng* suite allows a straightforward and reliable setup of evil twins. Due to this fact, several other tools use *airbase-ng* as basis. In future work we plan to extend our analysis to other software tools that could be used to mount evil twin attacks, e.g., the *hostapd* daemon or mobile AP functionality built in operating systems such as Android or iOS. We expect our results to be transferable to all types of software that emulate the high precision construction and sending process of beacon frames.

6. CONCLUSION

The evil twin attack poses a severe security risk for the usage of IEEE 802.11 hotspots. Therefore, there is a great need to equip users with additional tools and methods for the verification of the APs they connect to, in order to make sure that these are authentic and not traps operated by an attacker. In this paper we have proposed a taxonomy for classification of protection mechanisms against this threat and applied it to existing works in the domain.

As we have shown, our taxonomy enables a comprehensive classification of related work in the context of evil twin attacks. Using it, we were able to show relevant differences even of approaches that look very similar at first glance. Therefore, our analysis allows for the first time a thorough comparison of state-of-the-art work. From our findings we are able to draw the following conclusions: although sometimes claimed, no method is able to protect against the evil twin attack for arbitrary clients. The method with the highest accuracy, radio frequency fingerprinting, fails regarding practical applicability (since specialized hardware is required). Protocol modifications, although able to provide sufficient protection, are undesirable in general, as they rely on extensive modifications to deployed systems. Several methods offer remarkable detection rates for evil twin attacks. Nevertheless, in the entire context that we have disclosed in this paper, we have seen that these methods only solve a partial problem, i.e., they only protect against one specific attack scenario (e.g., when the attacker routes packets through the legitimate AP) – and not against the attack as a whole. Much research has been conducted in the field of clock skew based fingerprinting. Recently this field has made significant advances in terms of the achieved accuracy. These methods have potential but have so far been mostly evaluated only in laboratory settings. Hence, the evaluation of their practicability is still an open task.

Finally, we revealed that no method proposed so far explicitly exploits evil twin APs operated by software – although this is the most common attack scenario. We analyzed the popular *airbase-ng* software which is embedded as component in several tools that could be used for this malicious purpose. As we have shown, such software that emulates hardware behavior exhibits a significant timing inaccuracy and, therefore, leaks information that can be used for detection. We proposed a method to reliably identify whether an 802.11 AP is operated by specialized software. As our evaluation has shown, with this method we were able to detect APs operated by *airbase-ng* correctly in all examined cases. Moreover, the observations necessary for this judgement can be gathered by an arbitrary client within seconds.

Acknowledgements

This work has been supported by the Fonds National de la Recherche, Luxembourg and the EU FP7 project CONFINE.

7. REFERENCES

[1] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz. On the Reliability of Wireless Fingerprinting Using Clock Skews. In *Third ACM Conference on Wireless Network Security (WiSec'10)*, 2010.

[2] K. Bauer, H. Gonzales, and D. McCoy. Mitigating Evil Twin Attacks in 802.11. In *1st IEEE International Workshop on Information and Data Assurance (WIDA 2008) in conjunction with the 27th IEEE International Performance Computing and Communications Conference (IPCCC 2008)*, Austin, TX, USA, December 2008.

[3] S. Bratus, C. Cornelius, D. Kotz, and D. Peebles. Active Behavioral Fingerprinting of Wireless Devices. In *Proceedings of the First ACM Conference on Wireless Network Security (WiSec'08)*, 2008.

[4] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *14th ACM International Conference on Mobile Computing and Networking (MobiCom '08)*, 2008.

[5] T. Cross and T. Takahashi. Secure Open Wireless Access. In *Black Hat USA 2011.*

[6] R. Dhamija, J. D. Tygar, and M. Hearst. Why Phishing Works. In *SIGCHI Conference on Human Factors in Computing Systems (CHI '06)*. ACM, 2006.

[7] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker. Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting. In *15th Conference on USENIX Security Symposium (Usenix Sec 2006)*, 2006.

[8] K. Gao, C. L. Corbett, and R. A. Beyah. A passive approach to wireless device fingerprinting. In *40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010)*, 2010.

[9] H. Gonzales, K. Bauer, J. Lindqvist, D. McCoy, and D. Sicker. Practical Defenses for Evil Twin Attacks in 802.11. In *IEEE Globecom Communications and Information Security Symposium (Globecom 2010)*, Miami, FL, December 2010.

[10] H. Han, B. Sheng, C. c. Tan, and S. Lu. A Measurement Based Rogue AP Detection Scheme. In *28th Conference on Computer Communications (INFOCOM 2009)*, 2009.

[11] IEEE Computer Society. *Standard 802.11-2012: IEEE Standard for Information technology – Telecommunications and information exchange between systems, Local and metropolitan area networks – Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* `http://standards.ieee.org/findstds/standard/802.11-2012.html`.

[12] IETF. *RFC 2865: Remote Authentication Dial In User Service (RADIUS)*, June 2000. `http://www.rfc-editor.org/rfc/rfc2865.txt`.

[13] S. Jana and S. K. Kasera. On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews. In *14th ACM International Conference on Mobile Computing and Networking (MobiCom '08)*, 2008.

[14] T. Kim, H. Park, H. Jung, and H. Lee. Online Detection of Fake Access Points Using Received Signal Strengths. In *75th IEEE Vehicular Technology Conference (VTC Spring 2012)*, 2012.

[15] F. Lanze, A. Panchenko, B. Braatz, and T. Engel. Letting the Puss in Boots Sweat: Detecting Fake Access Points using Dependency of Clock Skews on Temperature. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security (AsiaCCS 2014)*, 2014.

[16] F. Lanze, A. Panchenko, B. Braatz, and A. Zinnen. Clock Skew Based Remote Device Fingerprinting Demystified. In *IEEE Global Telecommunications Conference (GLOBECOM 2012)*, 2012.

[17] L. Ma, A. Y. Teymorian, and X. Cheng. A Hybrid Rogue Access Point Protection Framework for Commodity Wi-Fi Networks. In *27th Conference on Computer Communications (INFOCOM 2008)*, 2008.

[18] D. Mónica and C. Ribeiro. WiFiHop - Mitigating the Evil Twin Attack Through Multi-hop Detection. In *16th European Conference on Research in Computer Security (ESORICS'11)*, 2011.

[19] M. R. Moxie Marlinspike, David Hulton. Defeating PPTP VPNs and WPA2 Enterprise with MS-CHAPv2. In *DEFCON'20 Hacking Conference*, 2012.

[20] K. N. N. Asokan, Valtteri Niemi. Man-in-the-Middle in Tunneled Authentication Protocols. http://eprint.iacr.org/2002/163.pdf, 2002.

[21] C. Neumann, O. Heen, and S. Onno. An Empirical Study of Passive 802.11 Device Fingerprinting. In *32nd International Conference on Distributed Computing Systems Workshops (ICDCS 2012 Workshops)*, 2012.

[22] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng. Device fingerprinting to enhance wireless security using nonparametric Bayesian method. In *30th IEEE International Conference on Computer Communications (INFOCOM 2011)*, 2011.

[23] B. Sieka. Active Fingerprinting of 802.11 Device by Timing Analysis. In *3rd IEEE Consumer Communications and Networking Conference (CCNC 2006)*, 2006.

[24] Y. Song, C. Yang, and G. Gu. Who Is Peeping at Your Passwords at Starbucks? - To Catch an Evil Twin Access Point. In *40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010)*, Chicago, IL, USA, 2010.

[25] J. Sunshine, S. Egelman, H. Almuhimedi, N. Atri, and L. F. Cranor. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In *18th USENIX Security Symposium (SSYM '09)*, 2009.