A Practical, Targeted, and Stealthy Attack Against WPA Enterprise Authentication

Aldo Cassola William Robertson Engin Kirda Guevara Noubir
Northeastern University
College of Computer and Information Science
{*acassola,wkr,ek,noubir*}*@ccs.neu.edu*

Abstract

Wireless networking technologies have fundamentally changed the way we compute, allowing ubiquitous, anytime, any-where access to information. At the same time, wireless technologies come with the security cost that adversaries may receive signals and engage in unauthorized communication even when not physically close to a network. Because of the utmost importance of wireless security, many standards have been developed that are in wide use to secure sensitive wireless networks; one such popular standard is WPA Enterprise.

In this paper, we present a novel, highly practical, and targeted variant of a wireless evil twin attack against WPA Enterprise networks. We show significant design deficiencies in wireless management user interfaces for commodity operating systems, and also highlight the practical importance of the weak binding between wireless network SSIDs and authentication server certificates. We describe a prototype implementation of the attack, and discuss countermeasures that should be adopted. Our user experiments with 17 technically-sophisticated users show that the attack is stealthy and effective in practice. None of the victims were able to detect the attack.

1 Introduction

Only a decade ago, gaining unauthorized access to a local network in an organization typically required physical access to the wired network. Today, the situation has dramatically changed. Wireless networks are ubiquitous, and allow users to have any-time, any-where access to information. At the same time, this convenience comes at a cost with respect to security: Wireless signals are not physically confined to the perimeter of an organization, but may be received by adversaries from very long distances. Therefore, although wireless networks have become an indispensable technology, they serve to increase the attack surface of an organization, and can potentially allow attackers to gain access to sensitive information over the æther.

Since their popularization and wide deployment, wireless networks have had a long history of security vulnerabilities. The initial effort to provide data confidentiality and client authentication – namely, Wired Equivalent Privacy (WEP) – suffered a widely publicized series of progressively more efficient attacks that necessitated WEP's replacement by the WPA family of security protocols [6]. Today, WPA Enterprise is widely used to protect large enterprise wireless networks against unauthorized access. Trust in the security of WPA Enterprise stems from the use of proven security protocols for authentication (e.g., SSL/TLS), and the centralization of client authentication and authorization on well-protected authentication servers (e.g., RADIUS). However, while the individual protocols that comprise the WPA Enterprise security suite are well-tested and widely regarded as being secure, the composition of different features, components, and protocols has received less scrutiny.

In this paper, we present a novel, stealthy, and effective variant of the evil twin attack [4, 22, 15] against WPA Enterprise networks. The key insight behind our attack is that the combination of cross-layer capabilities such as stealthy jamming using software radios, the inadequacy of wireless user interface mechanisms in popular commodity operating systems, and the insecure trust model used in wireless authentication makes real-world end-to-end attacks against wireless network authentication feasible. Our experiments with 17 technically-sophisticated users show that the attack is highly effective in practice and very difficult – if not impossible – for victims to detect.

Our targeted attack consists of four main phases. In the first phase, the attacker uses a software-defined radio to target the victim's specific device, without impacting other users of the network. To accomplish this, the attacker uses targeted, stealthy reactive jamming techniques to deny access to the legitimate wireless network the victim would like

to connect to. While recent work in the wireless community has made significant progress in developing mitigation techniques against jamming [44, 52, 8, 45, 49, 39, 12, 29], most of these techniques are still not part of wireless standards or deployed systems. Through reactive jamming, the victim is first disassociated from the legitimate network, and her probe requests are partially jammed to prevent legitimate access points from receiving them. We then send spoofed probe responses from a rogue wireless network. The combination of high-gain antennas and software radios makes such attacks possible from locations hundreds of meters distant from the targeted network.

In the second phase, the attacker takes advantage of predominantly unstudied and inadequate security mechanisms in popular commodity operating systems for disambiguating similar wireless network SSIDs. For example, the attacker can use character substitutions or invisible characters to create visually similar SSIDs – e.g., "`Enterprise`" vs. "`Enterprise␣`" – to trick users into connecting to a rogue network that is under the control of the attacker. This phase is an extension of the evil twin attack, where an attacker deploys a rogue access point that spoofs a legitimate wireless network.

In the third phase, the attacker presents a legitimate-appearing public key certificate, which can often be obtained for a cost of less than 200 USD. Since the certificates used in WPA Enterprise are not strongly bound to the network SSID, the victim's device that connects to the rogue network setup by the attacker has no basis for enforcing strict verification of certificates in popular commodity operating systems.

The fourth phase of the attack leverages the fact that WPA Enterprise deployments rely on the MSCHAPv2 [53] protocol for authentication, which has vulnerabilities that have been well documented [41]. MSCHAPv2 was initially designed for wired networks, and despite its use of outdated DES encryption, design flaws, and the availability of multiple automated cracking tools [50, 11, 37], MSCHAPv2 continues to enjoy wide usage, with nearly every major operating system and wireless infrastructure device supporting it. This can be partially explained by the fact that MSCHAPv2 is believed to be sufficiently secure when tunneled within an SSL/TLS session. However, the lack of a verifiable chain of trust from a CA to the network SSID allows an attacker to impersonate a trusted wireless network to capture victim authentication credential hashes.

In the final part of the attack, the attacker recovers plaintext authentication credentials by leveraging parallel password cracking techniques.

In summary, this paper makes the following contributions:

- We present a practical, end-to-end, stealthy, and targeted evil twin attack against WPA Enterprise networks. The attack leverages novel, specific weaknesses in the human-computer interfaces of commodity operating systems for managing wireless connections that have not heretofore been discussed in literature.
- We are the first to demonstrate a significant weakness that exists in modern wireless authentication systems today – namely, that authentication server certificates are not strongly bound to network SSIDs. Using this fact, an attacker can use selective jamming techniques to trick unsuspecting users into connecting to a rogue access point without receiving an invalid certificate warning. Note that certificates are widely believed to be the most effective form of protection against evil twin attacks by practitioners [47].
- We describe a prototype implementation of the attack, present experiments with real users that demonstrate that the attack is feasible and effective in practice, analyze its cost, and discuss countermeasures that should be adopted.

The paper is structured as follows. First, we present relevant background on WPA Enterprise in Section 2. The descriptions of the basic attack and further optimizations are given in Section 3. The prototype implementation of our attack is described in Section 4. Section 5 evaluates the effectiveness and cost of the attack. Section 6 discusses possible countermeasures against our attack. Finally, we present related work in Section 7 and briefly conclude in Section 8.

2 WPA Enterprise Background

In this section, we discuss relevant background information on WPA Enterprise. In particular, we focus on the WPA Enterprise authentication procedure, as well as implementation behavior when no known networks are available or when a new wireless network profile is created at the client. The reader is referred to the WPA Enterprise standard for further details [26].

2.1 Devices and Authentication

The WPA Enterprise authentication procedure involves several distinct devices: a *client*, an *access point*, and an *authentication server*. The client is a device with a 802.11-compliant network interface that requests access to the network. To connect to a network, the client communicates with an access point, which serves as a point of entry to one or more wireless networks. The authentication server is used to authenticate users of the wireless network, and typically runs a network authentication protocol such as RADIUS.

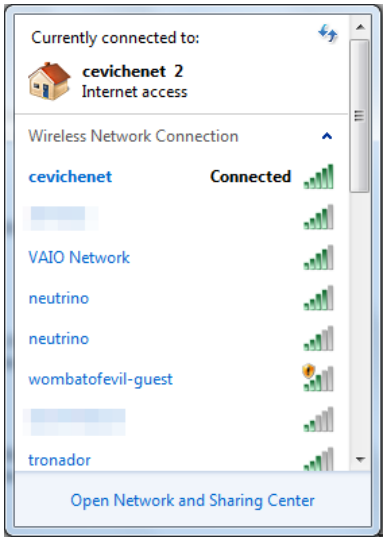

Figure 1. Wireless network list for Microsoft Windows-based operating systems. Here, there are two seemingly identical entries for the SSID neutrino**.**

WPA Enterprise authentication proceeds in three distinct phases: Discovery, Key Exchange, and Authentication. We note that PEAP [33] and MSCHAPv2 [53] are the most popular methods used to perform the Key Exchange and Authentication phases. With PEAP, the client initiates a TLS tunnel with the authentication server. During tunnel establishment, the authentication server presents its certificate, which ideally has been signed by a trusted certification authority (CA) that the client can verify and serves as the means of authenticating the network to the client. Once the TLS session has been established, MSCHAPv2 is then used to authenticate the client. Even though MSCHAPv2 has known flaws, the authentication procedure is commonly thought to be secure since it is encapsulated by the TLS session.

2.2 GUI Implementations

Network SSID lists on modern operating systems show only printable characters, with no way for the user to distinguish between identifiers that merely look similar. In addition, WPA Enterprise support differs considerably between platforms. Every client presents the user a different set of options when creating a new wireless network profile, some fields may be set automatically depending on user choice, and certificate management ranges from restrictive to permissive. We describe relevant behavior of common WPA clients below.

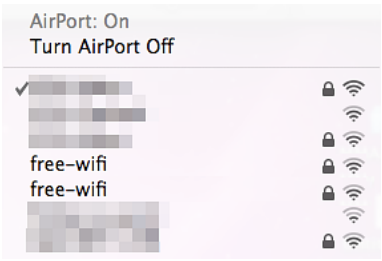

Figure 2. Wireless network list for Mac OS X. Again, there are two seemingly identical entries for the SSID free-wifi**.**

Windows (XP and above) Figure 1 shows the wireless network selection list for Windows-based operating systems. Note that Windows displays network names with no visual aid to distinguish similar SSIDs. Instead, the system shows seemingly identical networks as separate entries in the list of available networks. For instance, in Figure 1, neutrino is displayed twice.

When a Windows client receives an authentication server certificate during the authentication phase, a summary of the certificate's fields appears for verification. If the user accepts the certificate, the corresponding CA certificate can be used to verify the identity of the server for this network under "Trusted Root Certification Authorities." The user may also specify that the network should be authenticated only when the certificate's common name field matches a particular name. By default in Windows 7, the server name field is set to the value seen in the certificate, if accepted.

Apple Figure 2 shows the wireless network list for Mac OS X. Like Windows-based systems, Mac OS X also shows available SSIDs without visual aids. When creating a new network profile, the system selects most of the network parameters automatically – e.g., the use of PEAP. Once the authentication server presents its certificate, the client will present the user a summary of the certificate's fields for inspection. Mac OS X provides a visual aid in the form of quotation marks to delimit the SSID of the chosen network. In contrast to Windows clients, however, it is unclear how to restrict connections to a specific server name for a new network profile from the OS X GUI. Like Mac OS X, iOS uses quotation marks to visually delimit network SSIDs in the username and password dialog.

GNU/Linux Family GNU/Linux systems offer a variety of WPA client configuration interfaces, NetworkManager and wpa_gui being the most common. In both cases, the new network configuration dialogs are similar, and none offers visual aids to distinguish similarly-named networks. We note that text-based configuration tools allow checking

for the correct authentication server name, and to distinguish similar SSIDs by displaying them within quotation marks. However, text-based configuration remains an advanced task out of reach for the common user.

3 Targeted, Stealthy Evil Twin Attacks

In this section, we sketch an overview of our advanced, stealthy evil twin attack. The goal is to subvert the WPA Enterprise authentication described in Section 2, such that an attacker tricks a victim client into unwittingly authenticating to a rogue network with their credentials for a real, trusted network. The capture of these credentials allows the attacker to then authenticate to the trusted network with the privileges of the victim client. We note that this attack, when correctly performed, is completely transparent to the victim – that is, the victim will be entirely unaware that their network authentication credentials have been leaked.

In the following, we first detail the threat model we assume for the attack. Then, we present an exposition of the details of the attack, including several variations that increase both its effectiveness and stealthiness.

3.1 Threat Model

The attack we present in this work makes a number of realistic assumptions regarding the configuration of the wireless network and victim clients, as well as the capabilities of the attacker. We enumerate these assumptions here.

1. The wireless network uses WPA Enterprise for authentication, and uses MSCHAPv2 to authenticate clients over RADIUS. This is a common configuration; for instance, it is the default on recent versions of Windows.

2. The attacker can successfully communicate with a target wireless network, and transmit with sufficient power to successfully jam legitimate clients of the network. Section 5 shows the distances satisfying these requirements.

3. The attacker has sufficient resources to mount the attack. We demonstrate in Section 5 that the attack is feasible on common, high-end servers.

4. The victim clients run one of several commodity operating systems, including: Windows XP or later; Apple's Mac OS X or iOS; or, GNU/Linux with common GUI-based configuration tools.

3.2 Attack Description

In the following description of the targeted evil twin attack, let C be a legitimate client of the victim network N that uses an access point AP_N advertising SSID S_N. Let R_N be the victim network's authentication server that has been set up to perform PEAP with MSCHAPv2 for authentication with a certificate signed by certification authority CA_N, and can access the user database for N. In this scenario, C has a stored profile for N that includes its SSID, authentication credentials, authentication method, certification authority, and certificate fields to verify. The attacker A sets up an access point AP_A in range of the client C, using the same channel as AP_N. AP_A uses an authentication server R_A with a certificate similar to R_N.

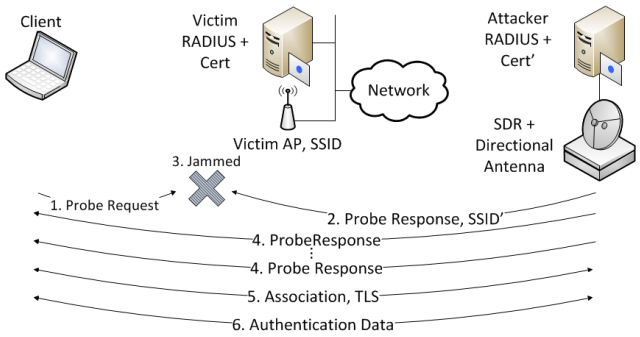

Figure 3. In the attack, an attacker leverages reactive jamming techniques to coerce a victim client to authenticate to a rogue network that appears identical to a target wireless network. A hash of the victim's authentication credentials is then captured. The hash is cracked using a high-performance password cracker (not shown). Once the plaintext password has been recovered, the attacker uses it to authenticate to the target network with the privileges of the victim client.

The attack is illustrated in Figure 3, and proceeds as follows.

1. AP_A sends Disassociation frames to C, forcing C to reassociate. These frames can be spoofed easily to appear being issued by AP_N using standard tools such as aircrack-ng [2].

2. C sends Probe Request frames to scan for active access points in order to reassociate with a suitable wireless network.

3. AP_A reacts to every Probe Request frame from C by jamming it and broadcasting Probe Response to C with SSID S_A. This has the effect of preventing reception of C's Probe Request frames at AP_N, allowing AP_A to impersonate AP_N.

4. C connects to S_A advertised by AP_A, thereby creating

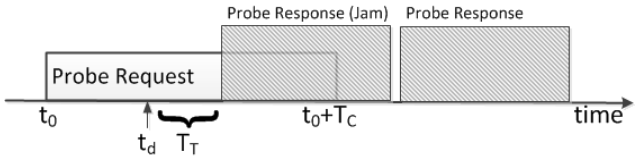

Figure 4. An example of reactive jamming, where an attacker conditionally jams a target based upon the contents of a wireless frame. In this example, the attacker detects that a target is issuing a Probe Request **frame. Before the target finishes transmitting the frame, the attacker issues a** Probe Response **frame to jam the target. Note that because the attacker can conditionally apply this jamming technique, the attack can be precisely targeted at a particular client and type of transmission.**

a new network entry in the client's wireless network configuration database. In effect, the attacker abuses the user's trust in the legitimate SSID S_N.

5. R_A presents C with a certificate signed by a trusted CA to avoid arousing suspicion.[1]

6. Normal authentication proceeds, such that C discloses a hash of his authentication credentials to R_A.

7. A transfers the captured hash to cloud-based password cracker to recover the plaintext authentication credentials.

In the following sections, we elaborate upon each of the steps of the attack outlined here. In particular, we describe the jamming strategy required to coerce C to communicate with AP_A. We then provide a procedure for choosing a suitable value for S_A and the properties necessary for R_A's TLS certificate. Finally, we outline a procedure for efficiently recovering authentication credential plaintext.

3.3 Reactive Jamming

Jamming a radio signal is often described as a physical operation where the jammer's transmitter outputs energy into the medium to disrupt communication without further knowledge of the data it is trying to disrupt. In the context of our attack, however, the jammer needs additional higher-layer knowledge of the victim's transmitted data in order to react properly. For the purposes of our work, a reactive jammer is a device that is capable of A) examining higher-level network protocol data contained in wireless frames, and B) conditionally jamming a target based on intercepted data *prior to that frame's end of transmission*. A depiction of this technique is shown in Figure 4.

In our attack, the access point AP_A operates by waiting for Probe Request frames from a client. Once a frame is detected, but before transmission ends, AP_A sends a continuous train of Probe Response frames that do not obey SIFS, DIFS, or PIFS timings in the 802.11 standard and lasts long enough to ensure that C receives it.[2] If the original frame is jammed before it reaches AP_N, there will be no response from it, removing its entry from the SSID list at the client C.

Successful jamming requires that the attacker is able to respond quickly to transmissions from C. Let t_0 be the transmission time of C's Probe Request frames, T_C the time required to transmit one Probe Request frame, T_T the radio turnaround time of AP_A, T_b the transmission time of 1 bit, and t_d the time at which AP_A detects a Probe Request frame from C. To successfully jam the Probe Request frame, T_T must satisfy the inequality

$$t_d + T_T < t_0 + T_C - T_b. \quad (1)$$

For instance, with a rate of 1.0 Mbps and frame size of $\approx$ 600 bits, $T_C < 600\mu s$.

Even though this technique only works for a fixed channel, Section 3.7 discusses the case where other channels must be jammed. Elimination of competing access points in the client's SSID list serves as the first step of the attack.

3.4 SSID Selection

As alluded to in Section 2.2, wireless network configuration interfaces provided by common operating systems can render maliciously-named networks indistinguishable from legitimate networks to the average user. In the simplest case, all that is needed to exploit this shortcoming is to use an SSID with a trailing or leading non-printable character in its name – e.g., "`Enterprise`" vs. "`Enterprise␣`". However, as in the case of browser URL spoofing, substitution of similar glyphs can produce spoofed network names. The standard defines SSIDs as an arbitrary string of up to 32 characters, and leaves the interpretation open to the implementation [26]. Given the abundance of glyphs available, especially in extended character sets, this implies that the set of possible spoofed SSIDs available for use by an attacker is potentially large.

When the victim client receives a Probe Response frame with a spoofed SSID, a new network profile will be

[1] We discuss issues surrounding obtaining trustworthy certificates later in the section.

[2] SIFS, DIFS, and PIFS are inter-frame spaces of different magnitudes. Shorter inter-frame spacing gives a transmitter greater transmission priority on a wireless channel. By necessity, reactive jammers ignore such inter-frame spacing standards.

created. None of the security settings for the legitimate network will apply to the new profile. Creation of a new network profile is a critical step of the attack, and it illustrates how subtleties in user interface designs can enable serious attacks despite the use of theoretically strong authentication protocols.

3.5 Authentication Server Certificates

Once a new profile exists in the client for the spoofed network, the attacker's authentication server must present a valid certificate to the client. To accomplish this task, the attacker can register a domain that appears similar to a trusted domain, or one that is related to wireless networking in some way (e.g., in our experiments, we registered the domain `openinfrastructures.net`). Since many wireless security products ship with self-signed certificates, users may be conditioned to accept any certificate that has a semblance of validity. We test this conjecture in Section 5 with user experiments, and show that it holds in practice.

The attacker uses the same CA used by R_N to sign her own certificate and bind it to a name in the registered domain. For instance, a deployment that uses VeriSign to sign its certificates can be targeted by an attacker registering a domain, and signing her own certificate with VeriSign as well.

Choosing a certificate may even be made simpler if a deployment chooses to use its own internal CA to sign certificates. If this is the case, then any CA in the client's trust store that is willing to sign certificates – e.g. VeriSign – is enough to prevent the operating system from reporting an error to the user.

Once the user accepts the attacker's certificate for the new network profile, the TLS tunnel between the victim client and the attacker's authentication server is established, and authentication commences. Crafting a reasonable certificate constitutes the final step in giving the attacker access to the less secure authentication protocol – e.g., MSCHAPv2 – that is encapsulated by TLS.

3.6 Recovering Authentication Credentials

Once the client starts MSCHAPv2 authentication, the attacker can record every frame exchanged, or even act as a full man-in-the-middle, forwarding frames between the client and the victim network. In both cases, the attacker can intercept the victim client's authentication credentials.

Breaking MSCHAPv2 is well documented [41], and tools such as asleap [50], mschapv2acc [11], and John the Ripper [37] can recover the plaintext secret given the conversation handshake. The speedup over brute-force search claimed by exploiting MSCHAPv2 weaknesses is 2^{16}, and this can be improved in practice by distributing the search over a pool of high-speed computation nodes. For instance, GPU computing platforms such as CUDA [35] allow the attacker to massively parallelize the key search.

To summarize the classic attack against NTLM, given a key $K = K_1||K_2||K_3$, the attacker must first obtain the values for either K_1 or K_2; obtaining K_3 is trivial, and only provides 2 bytes. Cracking DES once yields one of K_1 or K_2, giving a total of 9 out of the 16 bytes of the NTLM hash. The last 7 bytes of the hash can be obtained by running a dictionary attack against NTLM. In Section 5, we show how we were able to crack real-world passwords in a short period of time, and also empirically evaluate the computational effort required to completely execute this step of our attack.

3.7 Optimizations

While the basic procedure outlined above contains the essential details of the attack against WPA Enterprise, a number of optimizations are useful to consider when mounting this attack in the real-world. We consider possible improvements in the following.

The basic version of the attack might require the attacker to be in close proximity to the victim. Depending on the target, this may not be possible without arousing suspicion in the victim's activities. However, inexpensive directional antennas can make a remote attack possible from great distances. We examine the performance of one such antenna in Section 5.

The version of the attack presented above can only target a single wireless channel, and is limited in practice as wireless deployments move towards higher access point density; this has the effect of making more channels available to clients. For these scenarios, the attacker could use an additional jamming device for each of the available channels, using the same principle of operation as the basic attack. Another strategy would involve jamming the non-overlapping channels not targeted by the reactive jammer, essentially forcing the victim client to use the desired channel covered by the reactive jammer.

In order to convince more security-minded users to accept the rogue certificate, other social engineering techniques are possible. Consider an attacker AP that contains its own certificate and the one the legitimate authentication server presents. In the first iteration, AP_A uses the legitimate certificate and advertises S'_N, where S'_N is a visually equivalent variation of S_N. The user can inspect this certificate and since it will be the one expected, he will accept it. TLS session establishment will fail, and AP_A switches SSID to S''_N, another visually similar variation of S_N, creating yet another network. At this point the user may be more likely to accept the rogue certificate, as he has already

inspected and approved one.

Using arrays of GPUs to parallelize the computation can shorten the time needed to crack DES. Buying and managing such an array can be costly, but our evaluation in Section 5 demonstrates that current offerings such as Amazon EC2 [3] can greatly reduce these costs.

4 Attack Prototype Design and Implementation

In the current section, we describe our implementation of the advanced, targeted evil twin attack. Our prototype, shown in Figure 5, uses a desktop computer running Gentoo Linux for coordination. The machine contains an Intel Core 2 Quad Q9650 3GHz processor, 4GB of RAM, and an NVIDIA GeForce 9800GT graphics card with GTX280 GPUs. The reactive jammer is implemented using two USRP2 software-defined radio boards from Ettus Research [17], each with one RFX2400 daughterboard. A Buffalo WZR-HP-300NH wireless router provides the rogue wireless network, and two HyperGain HG2419G 2.5GHz 19dBi parabolic grid antennas are used to increase the range of the attack.

The reactive jamming component is a GNURadio-based software-defined radio (SDR) [20], running an 802.11b module developed by BBN [5] on our desktop host. The two USRP2 boards and daughterboards, one for reception and one for transmission, connect to the host through Gigabit Ethernet adapters. The USRP2s are connected to the directional antennas.

The reactive jammer is provided with the target client and access point MAC addresses, and the desired spoofed SSID. With these parameters, the jammer builds a Probe Response frame that is then passed to the modulation blocks for 802.11. The returning data is the train of signals to repeat on the sender, which is stored in memory. Once the receiver thread starts, the frame decoding function fills the received data buffer from the incoming signals, checking for the target client MAC address. When the desired Probe Request frame is encountered, the jammer transmits the stored Probe Response train.

For our rogue authentication server, we use the "Wireless Pwnage Edition" [51] of FreeRADIUS [18], a patch that maintains a challenge and response authentication log. Also connected to the host is the Buffalo router running the OpenWrt 10.03 firmware [38]. The router connects to the desktop computer, and is configured for WPA Enterprise using our desktop as a RADIUS server. We registered the domain `openinfrastructures.net`, and generated a certificate for `radius.openinfrastructures.net` signed with a well-known certification authority trusted by all common operating systems.

We explored four separate approaches for recovering plaintext from captured authentication hashes. We evaluated the use of a 9800GT card, a GTX280 card, a Tesla S870 cluster with 8 GPU boards, and an Amazon EC2 Cluster GPU Quadruple Extra Large Instance with two M2060 Fermi CPUs. We evaluate the relative impact of each approach on the efficiency of our attack in Section 5.

5 Evaluation

In this section, we report on an evaluation of our attack prototype. First, we establish bounds on the reaction time of our jammer, and the computational overhead required to recover authentication credential plaintexts. We evaluate the overall effectiveness of our prototype against a production WPA Enterprise-protected wireless network of our own. We quantify the range of our prototype in a separate experiment conducted using high-gain antennas in an urban environment. We report on the real-world effectiveness of the attack with user experiments with 17 technically-sophisticated participants. Finally, we present an economic, worst-case analysis of the attack in terms of the cost of the hardware and the software required to guarantee the success of the attack in practice against a particular victim.

5.1 Ethical Considerations

All of the experiments described in this section only target devices under our control, or ones for which we have obtained prior consent for testing. The experiments were performed in a wired environment whenever possible. However, to demonstrate the effectiveness of the attack, it was also necessary to perform it over the air. Note that fine-grained control of the attack is possible due to the nature of our jammer implementation that reacts only to specific MAC addresses. For test devices that we did not control, we obtained prior consent from the targeted users, and debriefed them after the experiments.

5.2 Jamming Speed

Recall from Section 3.3 that for the reactive jammer to successfully block transmission of the victim client's Probe Request frames, the radio turnaround time of the jammer must satisfy Eq. (1). This turnaround time effectively determines the maximum reaction speed to detected signals. Therefore, we performed the following experiment to measure the reaction time of our jammer.

In this experiment, we RF-wired a USB wireless dongle with an external antenna adapter to our jammer through a pair of 30dB attenuators. The output of the dongle and jammer transmitter is displayed on our signal analyzer.

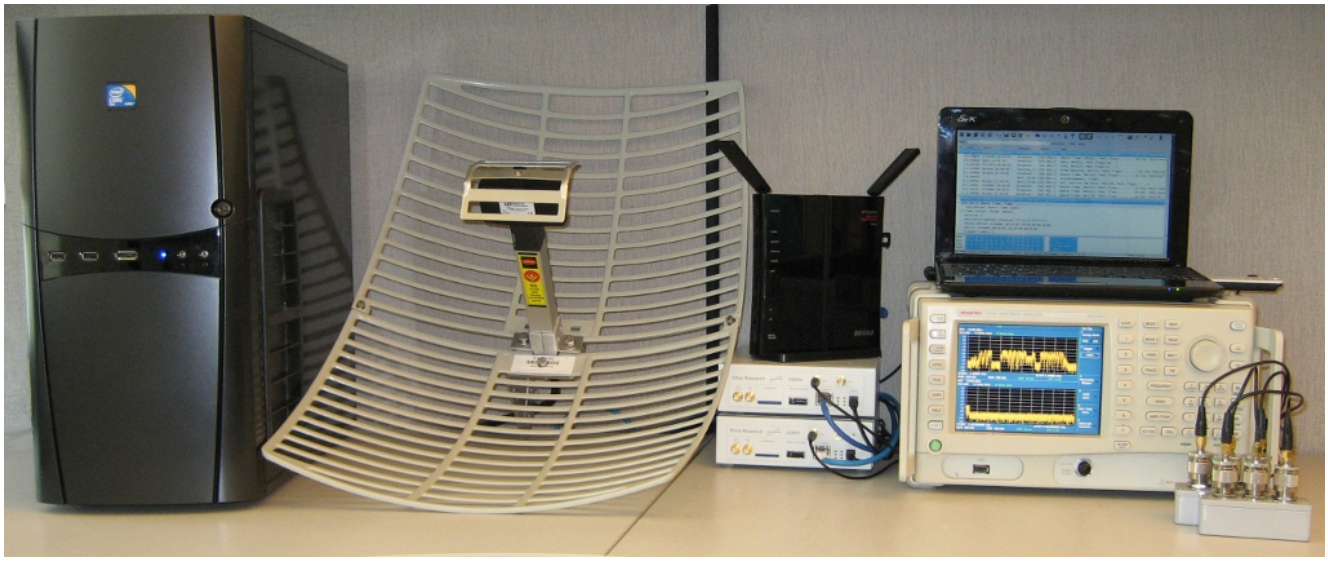

Figure 5. Overview of hardware used in the prototype implementation of the attack. From left: desktop, antenna, router, USRP2 software-defined radio, victim client, signal analyzer (for evaluation), and attenuators.

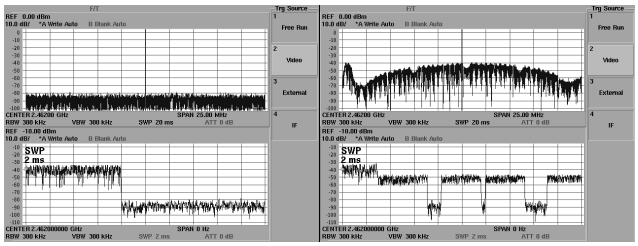

Figure 6. A demonstration of the reaction time of our jammer. On the left, the signal analyzer shows an uninterrupted $800\mu s$ frame transmitted by a wireless client. On the right, the signal analyzer shows a Probe Response **frame jammed at $300\mu s$. The horizontal grid width denotes $200\mu s$ divisions.**

Figure 6 illustrates the reaction time of our jammer. On the left panel, a single Probe Request frame lasting $800\mu s$ is shown while no jammer is active. On the right panel, the same Probe Request frame is interrupted by the jammer $300\mu s$ after transmission starts, establishing an upper bound on the reactive jamming capability of our prototype.

Because 802.11g network management packets are sent at the lowest rate (1 Mbps) to ensure delivery [26], our prototype's minimum response time of $300\mu s$ means that our reactive jammer is able to jam 802.11g management packets at around byte 38 using our software-radio implementation. Additionally, we note that while this reaction time is too long to allow for reliable jamming of 802.11n-based wireless networks, hardware is available that would render the attack possible in that context as well.

5.3 Plaintext Recovery

As we mention in Section 4, we explored several options for recovering authentication credential plaintexts. In particular, we deployed GPU-optimized password crackers on a low-end NVIDIA 9800GT card, a medium-end GTX280 card, a Tesla S870 cluster with 8 GPU boards, a Tesla C1060 cluster with 8 nodes, and an Amazon EC2 Cluster GPU Quadruple Extra Large Instance with two M2060 Fermi GPUs. To empirically demonstrate the required computational overhead to recover authentication credential plaintexts, we measured MD4 hashing speed by performing a full dictionary search on each device.

Figure 7 presents a hashing performance comparison between different hardware and thread configurations. To generate this plot, we ran an exhaustive dictionary search over a space of 8 alphanumeric characters with increasing threads per CUDA block to a maximum of 512 blocks. The dictionary build times range from a little over a week to approximately 13 hours.

As expected, the number of hashes computed per second initially increases with increasing parallelism, but eventually levels out or decreases due to thread contention. We note that for the 9800GT, the drop in performance at 512 threads is due to the configuration being too large for the card memory. We also note that a more careful implemen-

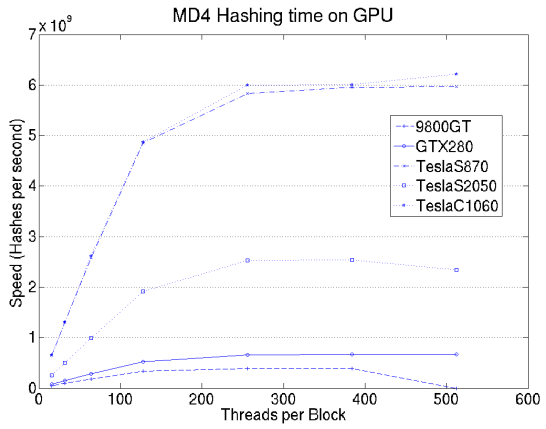

Figure 7. GPU hashing speed as a function of threads per CUDA block for dictionary generation. In particular, we compare the relative efficiency of four different hardware implementations.

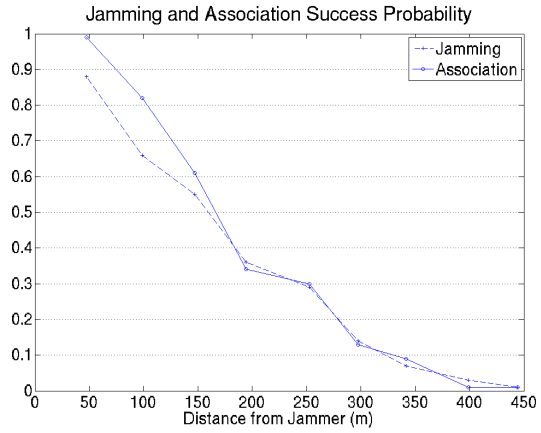

Figure 8. Jamming and association probability over 1,000 trials as a function of distance. Note that attackers are not limited to a maximum of 1,000 trials, and that each trial takes approximately one second to execute.

tation of the distributed cracker might improve performance at scale.

Once a full dictionary is generated, plaintext recovery requires at least one DES encryption per entry in the dictionary. An efficient GPU implementation [1] would allow the recovery to be fully-parallelized. However, the speed gain factor of the above implementation against a single CPU is around 10, meaning that a host with multiple cores is a reasonable substitute for the GPU implementation.

5.4 Attack Range

To quantify the maximum distance from which our attack could be mounted by our prototype, we deployed our high-gain directional parabolic antenna on the 16^{th} floor of a building in a large U.S. city. The antenna was connected to a computer running Kismet through a TL-WN722N USB wireless dongle. We recorded the MAC addresses of a number of access points, and queried these addresses using the Skyhook Wireless Positioning Service [43] to give an approximate distance from the placement of the antenna. The results show that our 19dBi antenna is able to communicate with networks 800m in distance.

To further quantify the performance of our prototype, we measured the jamming and association success probability of our system. We placed our prototype on the 4^{th} floor of the same building as in the previous experiment to limit its range, and co-located an access point for a target wireless network. Then, we varied the position of a test client at locations 50 meters apart in line-of-sight of the transmitter. As a test client, we used a GNU/Linux laptop containing an Atheros-based wireless interface and the factory antenna. For each position, we ran 1,000 network scans. Figure 9 sketches the relative position of all the components for the experiment.

For the chosen channel, we consider a single jamming attempt to be successful if the scan reveals only our system's MAC address. Failure to see our system in the scan, or seeing other transmitters in the channel, indicates a failed jamming attempt. Similarly, we consider an association to be successful if the test client associates with our rogue access point. If, on the other hand, the client connects to the legitimate wireless network, then we consider the association attempt to be a failure. The results for both tests are shown in Figure 8.

The results indicate that our prototype is able to jam and force client associations with close to 100% probability at ranges under 100m. As expected, success probabilities decrease with increased range; however, they do remain non-negligible up to 400m. We note that although these probabilities might indicate that the attack becomes ineffective at larger distances, the attackers are not constrained to 1,000 trials. Indeed, an attacker could quite feasibly perform hundreds of thousands or millions of trials in relatively short periods of time. Though this strategy comes with a corresponding increase in the risk of detection, it is not necessarily the case that the true risk is greater since one would have to be looking for signs of the attack in the first place.

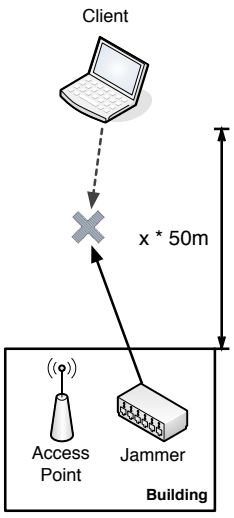

Figure 9. Experimental setup for jamming tests. The access point and jammer are co-located in this experiment, while the client distance from the AP and jammer is varied at 50m increments. The jamming and association success probabilities against the client are recorded over 1,000 trials at each distance.

5.5 User Experiments

Similar to work conducted by Jakobsson et al. [27, 28], we believe that realistic experiments are the only way to reliably estimate success rates of attacks in the real world.

In order to assess the feasibility of the attack we describe in this paper, we performed user experiments with 17 volunteer computer science graduate students that we recruited in our department. All participants were technically-sophisticated – that is, the participants were expert Internet and knowledgeable wireless network users. We set up the attack prototype in the lab with small Antenova B4844-01 antennas to limit range. The system was modified to stop jamming and answering requests for clients already seen and captured.

Before the experiments, we informed all participants that we would be capturing traffic, but without revealing the concrete attack that we were launching. The participants did not know that we were performing a security experiment. Furthermore, we reassured our participants that we would not be accessing any personal information (e.g., email contents, Facebook messages, etc.). Also, we anonymized captures when cracking such that no user could be mapped to a password , and only processed information automatically (i.e., we did not manually look at it).

Password	Time to find (s)
1	27
2	7,510

Table 1. Password search time, ordered by elapsed time.

For each participant, we assigned a series of common, innocuous tasks such as browsing the web, sending email through a web interface, and solving CAPTCHAs after authenticating with the university wireless network that we were targeting. Establishing a connection to the wireless network we were targeting was, therefore, not an end in itself, but was instead the means of accomplishing another unrelated goal. This was to ensure that the users were not aware of the attack that we were launching so that we could determine how effective the attack would be in a realistic, real-world setting. When available, the participants performed the tasks using their personal computers. Otherwise, we provided an Asus 1005PE netbook running Ubuntu 11.10.

After the completion of each user experiment, we debriefed the participant on the real purpose of the tests, asked if they had noticed any suspicious behavior during the experiment, and asked them to complete a self-assessment of their technical abilities. Figure 10 shows the distribution of scores reported by the users. Note that most participants were technically-sophisticated. Hence, the results we report in this section demonstrate that the attack works effectively even against very knowledgeable users.

Our prototype was able to successfully perform a man-in-the-middle attack against all of the 17 users who participated in our user experiments, including capturing the MSCHAPv2 conversation. All users authenticated with the wireless network that we were targeting, and only one reported seeing suspicious activity (even so, she still provided her password to the system).

To determine how long it would take for us to crack a password to gain illegal access to the network, we performed a brute-force search of 8 character alphanumeric passwords over the anonymized captures using John the Ripper on a 24 Xeon-CPU server at 2.4GHz in our lab. We parallelized the search to use one candidate password on each CPU. Table 1 shows that we were able to successfully crack a password after about 30 seconds, and then another one after about two hours. Hence, our attack is feasible in practice and can be used to gain unauthorized access to WPA Enterprise networks.

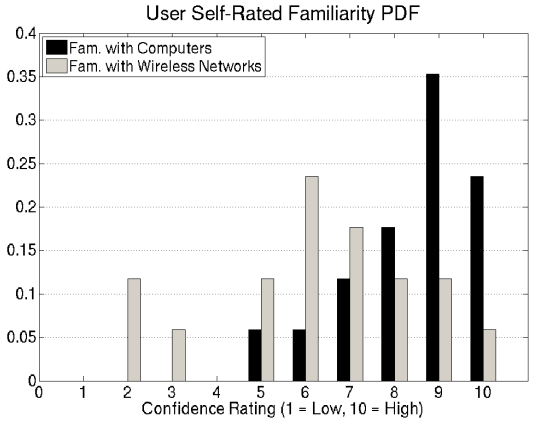

Figure 10. Results of a self-assessment of familiarity with computers and wireless networking performed by participants in our user study. Users tended to view themselves as technically sophisticated with respect to computing in general, while familiarity with wireless networking exhibits a bimodal distribution. In the latter case, most viewed themselves as sophisticated, while a smaller group classified themselves as less competent.

Component	Cost (USD)
1 Desktop Core 2 Quad 4GB RAM	580.00
2 USRP2 boards @ $1500 ea.	3,000.00
2 RFX2400 boards @ $275 ea.	550.00
1 Buffalo WZR-HP-300NH AP	66.00
2 Parabolic grid ant. @ $47.99 ea.	95.98
1 Standard TLS certificate	178.47
8 Tesla C10 computing devices @1,050 ea.	8,400
Total	**$12,870.45**

Table 2. A breakdown of the cost for each component of our attack prototype.

5.6 Economic Analysis

We split the estimated cost of building a system capable of launching our attack into two parts. First, we quantify the cost of the hardware we used, shown in Table 2. The total cost is dominated by the USRP2 SDR boards, at $3,000 USD, which is unsurprising given the relatively specialized nature of the hardware. Nevertheless, the final cost of $4,470.45 USD is well within the reach of many motivated attackers.

Next, we analyze the cost of the plaintext recovery components, including the MD4 dictionary generator to map strings to hashes, and the DES-cracking implementation if the attacker would like to have guaranteed success in cracking a specific victim's password. Agosta et al. [1] and Guneysu et al. [24] examine the use of equivalent GPU and FPGA clusters respectively for performing key searches in the $10,000 USD range. A cluster of 48 NVIDIA GTX260-216 boards completes a single DES in 18 days, while COPACOBANA does so in 12. Taking the former as an upper bound for the time required to completely explore the possible space of keys, the budget for such a plaintext recovery component in under 20 days comes to less than $15,000 USD when using the Amazon EC2 GPU extra-large instance for dictionary generation.

From the above cost analysis, our attack can be performed for under $20,000 USD if a specific user is targeted rather than when the attacker is interested in gaining general access to a WPA Enterprise network as we demonstrated in our user experiments. We believe that this renders the attack feasible for a wide range of attackers who are interested in launching targeted attacks – e.g., criminal gangs, well-funded corporations, and nation-states. Note that for those entities with significant resources – e.g., the security agency of a country – the attack could be performed in a far more efficiently than above by investing in faster resources.

6 Countermeasures

Preventing our attack requires coordination by defenders on multiple layers. Several countermeasures are required to mitigate such attacks. In this section, we present and discuss some possible mitigation strategies.

A central component of our attack concerns the spoofing of network SSIDs in wireless network user interfaces. Implementations that can provide visual cues to the user that a rogue SSID is distinct from a similar-appearing trusted SSID could potentially neutralize or warn against the attack. For example, SSIDs might be displayed between visible delimiters, or visible placeholders might be displayed for non-printable characters. More sophisticated approaches might check for SSID similarity, similar to phishing heuristics integrated into modern browsers, and alert the user in such cases. Note, however, that educating users to take notice of security-relevant information is a notoriously difficult problem, especially in a wireless setting where most users are not aware of the risks. For example, simple session-hijacking tools such as Firesheep have received much attention because of their simplicity and effectiveness in wireless environments.

Unfortunately, flaws in MSCHAPv2 and the wide availability of distributed computational resources make it feasible to recover plaintext user credentials. Attacks against it are well-known [41] and implemented in a variety of tools. Perhaps worst of all, these attacks have negligible parallelization overhead. Our work supports the fact that

more secure alternatives to MSCHAPv2 need to be deployed in WPA Enterprise networks. Even small changes to MSCHAP, such as using a single AES encryption for the client response, could make the attack we describe infeasible. Also, using client certificates for authentication would avoid relying on password-based authentication protocols all-together.

In this work, we show that the lack of a strong binding between the SSID and the authentication server certificate is a significant security problem. Tying the wireless network SSID to the authentication server certificate gives the victim a chance to detect the attack and determine that she is actually contacting a rogue access point. WPA clients should also require certificate authority selection and common name verification to protect from SSID spoofing attacks.

Allowing users to configure their own network profiles to be able to connect to an organization's wireless network may be convenient from a management and organizational point of view. However, this flexibility creates the potential for attacks such as ours. For deployments using WPA Enterprise, we believe that it might be better for a central authority to distribute wireless profiles to clients, and disallow dynamic profile creation.

While our attack makes use of directional antennas to increase the range of the attack, physical security techniques (e.g., secure pairing) have recently been proposed to mitigate rogue device attacks [9, 32]. Work in this vein, although mainly focused on short range handsets, could potentially be adapted to newer 802.11n multi-antenna wireless clients and WPA Enterprise networks. This could severely limit the range of the attack, and force the jammer to be co-located with the victim clients, reducing the problem to one of physical security.

7 Related Work

Our attack against WPA Enterprise incorporates a combination of cross-layer vulnerabilities, from the physical communication medium to the human-computer interface. In this section, we place the attack in the context of related work.

General wireless attacks Protection against impersonation and man-in-the-middle attacks in wireless networks has garnered interest in the networking community in recent years. Techniques include providing visual feedback to the user such as light or sound for secure pairing [40, 23], using correlated motion [10], using ambient signals around the transmitter to authenticate it [32, 4, 22], and special packet coding techniques to detect jamming [21]. Anti-jamming systems have also been studied for decades, since the introduction of spread spectrum communication [42] to newer forms that allow key establishment for spread spectrum techniques in hostile environments [45]. While such techniques provide useful building blocks for securing wireless networks, they are limited in applicability for the existing WPA Enterprise standard and its deployment constraints.

Evil twin protection Most evil twin attacks in the literature contemplate the impersonation of unsecured networks and do not include jammers, while our attack relies on a composition of cross-layer techniques to exploit weaknesses in UI design, authentication protocols, and the physical layer, achieving stealthy and targeted attacks. Proposals in the literature against evil twins fall into several categories, none of which is enough to prevent our attack. Secure device pairing [40, 32, 23] seeks to use properties in or around the devices that will be communicating to establish their identity. These approaches require line-of-sight access to the access point, or assume the attacker can't co-locate with the device. Although our attack is capable of attacking long-range targets, it can also be placed close to an AP or client. Dedicated hardware could render the attack less conspicuous, and could allow for deployment with plug devices or smaller, making its physical presence hard to detect.

Protocols relying on trust-on-first-use assume that the first time a network is configured, the environment is secure. Our attack, however, exploits this assumption by creating a new network to trust as needed and, as such, trust-on-first-use approaches and similar are vulnerable [4, 22].

Wireless intrusion prevention systems (WIPS) capture packets and search for attack patterns [13]. When one is found, the WIPS alerts a supervisor and may terminate the overheard exchanges. However, our attack involves posing as a legitimate access point to the client and can even involve spoofing a jammed access point hardware address. Therefore, from the perspective of a WIPS, our attack looks no different from a normal association.

Device fingerprinting assumes the evil twin hardware differs from the victim deployment and sends malformed probe packets to elicit responses from the twin, which will be compared against a device table [7]. This approach requires the evil twin to respond to packets from the probe. Our targeted jamming and response approach requires the probe to know the attack target beforehand, while our use of directional antennas forces the probe to co-locate with the victim, limiting its usefulness. In addition, an attacker using the same hardware as the victim is undetectable under this approach.

Password-based authentication A substantial body of work in the security literature has studied attacks against password-based authentication, from the popular John the Ripper [37] that relies of lists of common words, to time-

space trade-offs approaches culminating in the well-known rainbow table [25, 36] to probabilistic approaches such as Markov models [34] and context-free grammars [48] derived from public password lists. In the WPA world, wpacracker is a recent commercial effort focused on bringing cloud resources to bear on the problem of cracking WPA2-PSK passwords [31]. We note, however, that in wpacracker's target challenges are exchanged between two nodes with a shared secret. Our work, however, is an end-to-end attack against WPA Enterprise networks. The most significant similarity between the two efforts is their use of cloud computing to parallelize the plaintext recovery process.

While our attack relies upon a robust password cracking component to successfully recover WPA Enterprise passphrases, it is agnostic and, therefore, orthogonal to the underlying technique used. In our current prototype, we make use of parallel DES cracking techniques on GPUs and can use cloud computing nodes.

User interface attacks The attack we describe leverages vulnerabilities in user interfaces that fail to convey important security-relevant information to the user. Attacks in this vein have been known since the early days of multi-user computing, where mechanisms such as secure attention sequences – e.g., the now-infamous CTRL-ALT-DELETE – were introduced to establish a trusted path between the user and the operating system.

The particular vulnerabilities exposed by our work bear resemblance to a number of attacks that have been launched against the web browser, where the URL plays a similar role to a wireless network SSID in that users make trust decisions based upon the reputation of a particular domain or network name. In particular, homograph attacks [19] have been used to mount phishing attacks against users that expect to visit a trusted domain by tricking them into visiting a site with a similar-appearing domain name by exploiting similarities between glyphs in a character set – e.g., paypa1.com vs. paypal.com – or across character sets [46]. Our attack uses similar techniques, although an important difference in the context of wireless SSIDs is the general lack of delimiters, allowing for the use of invisible and non-printable characters.

Subverting the SSL/TLS PKI infrastructure that is relied upon by HTTPS to verify the authenticity of web servers is an important class of web security attacks that mirrors – to some extent – our user interface attacks in the context of WPA Enterprise. SSL/TLS has recently suffered a number of issues, such as the questionable trustworthiness of some certificate authorities [14, 16] that has led to the issuance of malicious, but correctly signed, certificates. Another interesting class of attack involves browser URL spoofing by, for example, creating SSL certificates that spoof trusted domains in vulnerable browsers by injecting null bytes in the certificate common name field [30].

8 Conclusions

This paper presents a novel wireless attack against WPA Enterprise networks. The key insight behind our attack is that the combination of cross-layer capabilities such as stealthy jamming using software radios, the inadequacy of wireless user interface mechanisms in popular commodity operating systems, and the insecure trust model used in wireless authentication makes end-to-end attacks against wireless network authentication feasible in practice. Our user experiments demonstrate that the attack is highly effective in practice and very difficult for victims to detect. We are the first to show significant deficiencies in wireless management user interfaces for commodity operating systems, and also the first to highlight the weak binding between wireless network SSIDs and authentication server certificates. We described a prototype implementation of the attack, analyzed its effectiveness and cost, and discussed countermeasures that should be adopted.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0915985.

References

[1] G. Agosta, A. Barenghi, F. D. Santis, and G. Pelosi. Record Setting Software Implementation of DES Using CUDA. In *Proceedings of the International Conference on Information Technology: New Generations*, pages 748–755. IEEE Computer Society, 2010.

[2] Aircrack. Aircrack-ng. `http://www.aircrack-ng.org/`, 2012.

[3] Amazon. Amazon Elastic Cloud Computing (EC2). `http://aws.amazon.com/ec2/`, 2012.

[4] K. Bauer, H. Gonzales, and D. McCoy. Mitigating Evil Twin Attacks in 802.11. In *Proceedings of the Performance, Computing and Communications Conference*, pages 513–516, December 2008.

[5] BBN Technologies. 802.11b Transmitter/Receiver. `https://www.cgran.org/wiki/BBN80211`, 2008.

[6] A. Bittau, M. Handley, and J. Lackey. The Final Nail in WEP's Coffin. *Security and Privacy, IEEE Symposium on*, 0:386–400, 2006.

[7] S. Bratus, C. Cornelius, D. Kotz, and D. l. Peebles. Active behavioral fingerprinting of wireless devices. In *Proceedings of the first ACM conference on Wireless network security*, WiSec '08, pages 56–61, New York, NY, USA, 2008. ACM.

[8] M. Cagalj, S. Capkun, and J.-P. Hubaux. Wormhole-Based Antijamming Techniques in Sensor Networks. *IEEE Transactions on Mobile Computing (TMC)*, 2007.

[9] L. Cai, K. Zeng, H. Chen, and P. Mohapatra. Good Neighbor: Ad hoc Pairing of Nearby Wireless Devices by Multiple Antennas. In *Network and Distributed System Security Symposium (NDSS)*, 2011.

[10] C. Castelluccia and P. Mutaf. Shake them up!: a movement-based pairing protocol for cpu-constrained devices. In *MobiSys*, pages 51–64, 2005.

[11] B. Charles. mschapv2acc, a proof of concept of MS-CHAP-V2 auditing and cracking tool. `http://code.google.com/p/mschapv2acc/`, 2010.

[12] J. Chiang and Y.-C. Hu. Cross-layer jamming detection and mitigation in wireless broadcast networks. In *Proceedings of MobiCom*, 2007.

[13] Cisco. Cisco Adaptive Wireless IPS Software. `http://www.cisco.com/en/US/products/ps9817`, 2011.

[14] Comodo, Inc. Comodo Report of Incident. `https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`, 2011.

[15] J. Damsgaard, M. A. Parikh, and B. Rao. Wireless commons perils in the common good. *Communications of the ACM*, 49(2):104–109, February 2006.

[16] Electronic Frontier Foundation. A Post Mortem on the Iranian DigiNotar Attack. `https://www.eff.org/deeplinks/2011/09/post-mortem-iranian-diginotar-attack`, 2011.

[17] Ettus Research. Universal Software Radio Peripheral. `http://www.ettus.com/`, 2012.

[18] FreeRADIUS Project. FreeRADIUS RADIUS Server. `http://freeradius.org/`, 2012.

[19] E. Gabrilovich and A. Gontmakher. The Homograph Attack. *Communications of the ACM*, 45(2), February 2002.

[20] GNU Radio Project. GNU Radio. `http://gnuradio.org`, 2012.

[21] S. Gollakota, N. Ahmed, N. Zeldovich, and D. Katabi. Secure In-Band Wireless Pairing. In *Proceedings of the USENIX Security Symposium*, 2011.

[22] H. Gonzales, K. Bauer, J. Lindqvist, D. McCoy, and D. Sicker. Practical Defenses for Evil Twin Attacks in 802.11. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–6, December 2010.

[23] M. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and Clear: Human-Verifiable Authentication Based on Audio. In *Proceedings of the IEEE Internation Conference on Distributed Computing Systems*, page 10, 2006.

[24] T. Guneysu, T. Kasper, M. Novotny, C. Paar, and A. Rupp. Cryptanalysis with COPACOBANA. *IEEE Transactions on Computers*, 57:1498–1513, 2008.

[25] M. Hellman. A Cryptanalytic Time-Memory Trade-Off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[26] IEEE. Standards for Wireless, Local, and Metropolitan Area Networks. `http://standards.ieee.org/findstds/standard/802.1X-2010.html`, `http://standards.ieee.org/about/get/802/802.11.html`, 2012.

[27] M. Jakobsson, P. Finn, and N. Johnson. Why and How to Perform Fraud Experiments. *Security & Privacy, IEEE*, 6(2):66–68, March-April 2008.

[28] M. Jakobsson and J. Ratkiewicz. Designing ethical phishing experiments: a study of (ROT13) rOnl query features. In *15th International Conference on World Wide Web (WWW)*, 2006.

[29] Y. Liu, P. Ning, H. Dai, and A. Liu. Randomized differential dsss: jamming-resistant wireless broadcast communication. In *Proceedings of the 29th conference on Information communications*, pages 695–703, Piscataway, NJ, USA, 2010. IEEE Press.

[30] M. Marlinspike. sslstrip. `http://www.thoughtcrime.org/software/sslstrip/`, 2012.

[31] M. Marlinspike. wpacracker. `https://www.wpacracker.com/`, 2012.

[32] S. Mathur, R. Miller, A. Varshavsky, W. Trappe, and N. Madayam. ProxiMate: Proximity-based Secure Pairing using Ambient Wireless Signals. In *International Conference on Mobile Systems, Applications, and Services (MOBISYS)*, 2011.

[33] Microsoft, Inc. Extensible Authentication Protocol. `http://technet.microsoft.com/en-us/network/bb643147`, 2011.

[34] A. Narayanan and V. Shmatikov. Fast Dictionary Attacks on Passwords using Time-Space Tradeoff. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, Alexandria, VA USA, November 2005. ACM.

[35] nVidia. Compute Unified Device Architecture. `http://developer.nvidia.com/object/cuda.html`, 2012.

[36] P. Oeschlin. Making a Faster Cryptanalytic Time-Memory Trade-Off. In *Proceedings of Advances in Cryptology (CRYPTO)*, Lecture Notes in Computer Science. Springer, 2003.

[37] Openwall Project. John the Ripper. `http://www.openwall.com/john/`, 2012.

[38] OpenWRT Project. OpenWRT. `http://www.openwrt.org`.

[39] K. Pelechrinis, I. Broustis, S. V. Krishnamurthy, and C. Gkantsidis. Ares: an anti-jamming reinforcement system for 802.11 networks. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 181–192, New York, NY, USA, 2009. ACM.

[40] V. Roth, W. Polak, E. Rieffel, and T. Turner. Simple and effective defense against evil twin access points. In *Proceedings of the ACM Conference on Wireless Network Security*, pages 220–235, New York, NY, USA, 2008. ACM.

[41] B. Schneier, Mudge, and D. Wagner. Cryptanalysis of Microsoft's PPTP Autentication Extensions (MSCHAP-v2). `http://www.schneier.com/paper-pptpv2.pdf`, 1999.

[42] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread Spectrum Communications Handbook*. McGraw-Hill, 2001.

[43] Skyhook, Inc. Skyhook Wireless Positioning System. `http://skyhookwireless.com/developers/`, 2012.

[44] M. Strasser, B. Danev, and S. Capkun. Detection of Reactive Jamming in Sensor Networks. *ACM Transactions on Sensor Networks (TOSN)*, 2010.

[45] M. Strasser, C. Popper, S. Capkun, and M. Cagalj. Jamming-resistant Key Establishment using Uncoordinated Frequency Hopping. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2008.

[46] The Shmoo Group. IDN Homograph Attack. `http://www.shmoo.com/idn`, 2005.

[47] WatchGuard. Anatomy of a Wireless "Evil Twin" Attack. `http://www.watchguard.com/infocenter/editorial/27079.asp`, 2011.

[48] M. Weir, S. Aggarwal, B. de Medeiros, and B. Glodek. Password Cracking Using Probabilistic Context-Free Grammars. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA USA, May 2009. IEEE Computer Society.

[49] M. Wilhelm, I. Martinovic, J. B. Schmitt, and V. Lenders. Short paper: reactive jamming in wireless networks: how realistic is the threat? In *Proceedings of the fourth ACM conference on Wireless network security*, WiSec '11, pages 47–52, New York, NY, USA, 2011. ACM.

[50] J. Wright. ASLEAP, Exploiting CISCO LEAP. `http://www.willhackforsushi.com/Asleap.html`, 2008.

[51] J. Wright. FreeRADIUS, Wireless Pwnage Edition. `http://www.willhackforsushi.com/7`, 2010.

[52] W. Xu, W. Trappe, and Y. Zhang. Defending wireless sensor networks from radio interference through channel adaptation. *ACM Transactions on Sensor Networks*, 4:18:1–18:34, September 2008.

[53] G. Zorn. Microsoft PPP CHAP Extensions, Version 2. `http://tools.ietf.org/html/rfc2759`, 2000.