Crossroads
The ACM Student Magazine

WiFi Exposed

by ***Andrea Bittau***

Introduction

Over the past few years, IEEE 802.11 wireless networks have become increasingly widely deployed. Wireless LANs can be found in coffee shops, airports, hospitals, and all major institutes. However, as for conventional wired networks, the spread of such networks may have been faster than the diffusion of security knowledge about them. As a consequence, 802.11 is the new playground for many hackers, who are attracted to the environment by virtue of its anonymity. Attacks may be traced back to the wireless network, but the intruder could have been anyone driving by within the radius of the network, making it hard, if not impossible, for him/her to be traced. Securing wireless networks is a hard task, because the standard solutions do not work effectively in guaranteeing privacy and authentication, as this article shows; as a consequence, many wireless networks are left open.

This article is structured as follows: initially, an overview of the 802.11 protocol is presented. This is followed by an analysis of the steps involved in connection to and use of such a network, first in the absence of encryption and then taking into account WEP. Attacks for these different scenarios are presented and analyzed, leading to the conclusion that WEP is unsuitable as the sole security measure for such links. Finally, attacks on wired networks that are connected to a wireless LAN are analyzed.

The article concludes that existing standards for wireless security as applied to the most widely used wireless standard, 802.11, are inadequate in several ways, can be attacked using publicly available tools, and lead to a false sense of security. Some advice about mitigation of threats is offered throughout the article, but the most effective solution is awareness of potential attacks and the maximization of the amount of time and effort needed to break into the network by using defence in depth.

IEEE 802.11 Overview

IEEE 802.11 [13] is the standard for the family of wireless network protocols. It deals with the **Medium Access Control** (MAC, OSI Layer 2) and Physical Layer (OSI Layer 1). One of the most used protocols in this family is 802.11b, which operates at the 2.4GHz unlicensed frequency band and has a maximum bandwidth of 11Mbit/s. 802.11g is a recent and rapidly spreading development that is backwards-compatible with 802.11b, but has a headline data rate of 54Mbit/s. In order for multiple networks to share the same medium, thus having more than one wireless network in the same physical place, there are different communication channels that may be used, each with a different frequency band. Channels in 802.11b/g vary from 1 to 14 (2,412-2,484GHz) but have legal constraints on which subset of channels may be used; for example, channel 14 is used only in Japan. A final, though less prevalent variant is 802.11a, which operates around 5GHz and also has a maximum bandwidth of 54Mb/s.

Overall,there are three different logical levels that must be considered within the protocol. At the physical level we need to know if sending is allowed at this point in time, if our data was received, and so on. This is dealt with the use of **control frames**. We also need to have the ability to advertise our presence, possibly to associate to a peer, to authenticate, and so on. A family of **management frames** takes care of these issues. The last class of frames are **data frames** that deal with upper layer traffic, which is actual data to be transmitted over the wireless network.

Operation Modes

There are two modes of operation in 802.11 networks: **Ad-hoc** (infrastructure-less, DCF) and **Infrastructure** (managed via access points in PCF mode). The ad-hoc mode is mainly used for temporary networks, in which peers communicate with each other directly. In infrastructure mode, peers associate with and communicate via a single **access point** (AP, also called a **base station**) at any point in time; they no longer communicate directly. A network may have multiple APs and clients may **roam** among them, effectively switching the AP with which they are associated. An obvious advantage of this approach is that peers need only to be in range of the APs and not of each other, simplifying the further connection to the Internet.

In AP-based networks, the concept of authentication and association needs to be considered. Before a client may communicate with an AP, it must first authenticate itself and must then send an association request. There are two modes of authentication: **Open System** and **Shared Key**. As the name suggests, open system is a null authentication algorithm in which anyone may associate. Shared key authentication, on the other hand, is achieved by using the **Wired Equivalent Privacy** (WEP) algorithm, which requires prior agreement of a shared secret WEP key. After a successful authentication request and association response, the client is in the associated state and may start sending data.

Nodes may additionally encrypt data frames using the WEP algorithm. It is important to notice that only data frames, and, specifically, only the actual data section, is encrypted if the WEP option is enabled.

The above two uses of WEP are somewhat independent. Thus, for example, a system may be open, but use WEP in data transmission — in other words, a peer may associate with the AP but will not be able to do much, as it will not be able to encrypt or decrypt data.

Network Identification and Discovery

Stations must advertise themselves periodically in order to inform mobile hosts of their presence. Networks must also have a naming strategy so distinction between different logical networks may be achieved when more than one is present in the same geographical location. This is done via the Service Set Identity (SSID), which becomes the name of the network. Two modes exist for network discovery: **passive** and **active** scanning. The first mode implies listening to the network for **beacon** frames (a subtype of management frames), which advertise networks. The second mode implies **probing** the network for a specific SSID, in effect broadcasting to determine whether a specific network is present. Finally, note that networks may be allocated to different channels; thus it is important to select the same channel as that on which the network operates.

The next section focuses on how to determine all parameters needed for a successful connection, and how to perform possible attacks once connected.

Simple Scenario: No WEP

Administrators often simply leave 802.11 networks open, largely for two reasons: ignorance, and difficulty in key management. WEP requires distribution of a key, which is, at best, tedious in the first instance, and which causes further problems each time it must be renewed. This is a particular problem in environment in which users change often. The "solution" to this management problem, adopted by many, is to leave the network open, without the use of WEP. However, many wireless network cards may be put in **rfmon** mode or monitor mode, which makes the card firmware pass all the data it receives to the operating system. This effectively makes **eavesdropping** (intercepting data) trivial. The only constraint is that only one channel at a time may be monitored, though this is only a minor inconvenience because some cards support hardware channel hopping, which makes it feasible to monitor more than one channel.

It is worthwhile noting at this point that some 802.11 network cards also allow sending all types of frames and not only data packets. Normally, the firmware deals with management and control frames, so the ability manually to send any type of frame becomes useful in active attacks. Abaddon's Airjack [**6**] is a driver for Linux that enables the sending of raw data on a wireless network. This is crucial for injection attacks, which are discussed later in the article.

Associating With the Network

The first step is to physically locate the network, which usually involves listening for traffic and channel hopping. This can be done with tools such as dragorn's Kismet [**5**]. Analyzing beacon packets gives enough information (at least at the MAC layer) about the network. The information is always in clear-text, as WEP only encrypts the data portion of data frames even if it is used, thus leaving management frames in clear-text. Beacon frames contain the channel of the network (useful in order to avoid the need for channel hopping), the SSID of the network (needed for association), the supported rates of the network (useful to distinguish 802.11g and 802.11b networks), and a flag indicating whether WEP is being used. Some APs can disable beacon broadcasting. In such cases, it may be enough to wait for probe request packets from clients, as they contain the same information as beacon packets.

The next step involves associating with the AP. If WEP is not used, this merely involves setting the SSID. Some APs allow only certain MAC addresses (physical addresses) for association. **Spoofing** (faking) MAC addresses is again trivial, being simply a matter of waiting for authorized clients to associate and stealing their MAC address for use in future.

IP Address and Router Discovery

Once associated with the AP, connection, so far as the mobile node is concerned, is very much like being physically connected to the hub of a wired LAN. The next step in becoming active is IP address discovery, which sometimes involves using only the **Dynamic Host Configuration Protocol** (DHCP). If so, an IP address is assigned and the **Domain Name Service** (DNS) information and default gateway (router) are obtained. Alternatively, one may eavesdrop on the network until an IP address is recovered via IP or **Address Resolution Protocol** (ARP) traffic. Other interesting ideas include sending ICMP echo requests to the broadcast address or multicast addresses, as routers reply to such traffic quite frequently, or simply trying some of the common private IP classes: 10.0.0.0/8, 172.16.0.0/12 or 192.168.0.0/16. Tools such as THC-RuT [11] may be used to perform IP discovery in various ways (e.g., via ARP).

Once an IP address is found, gateways tend to have the same network portion of the address, but end with either 1 or 254.

IP and ARP, or any traffic encapsulated in IEEE 802.2 **Logical Link Control** (LLC) may now be sent. In most cases, there are no firewall restrictions because the mobile node is, at this point, viewed as being inside the local network, obviating the need for a border firewall.

Once access has been obtained, it may be abused. Because most wireless networks are connected to the Internet, and because it is possible to associate with wireless networks fraudulently, it is clearly possible for a rogue mobile node to launch anonymous attacks on machines elsewhere in the system.

Advice

There is little that can be done to prevent someone associating with a wireless network in the absence of proper encryption and authentication. Furthermore, it is argued later in this article that the simple WEP scheme based on shared-key authentication is also inadequate. Instead, therefore, authentication must be achieved at a higher layer by, for example, forcing clients to login a **Virtual Private Network** (VPN). The network may then be configured such that all non-VPN traffic is simply ignored or re-routed to some informational web server. Clients that have associated with an AP but which are not authenticated will not be able to do much other than flood the network and perform **Denial Of Service** (DoS) attacks. The attacker would have been able to conduct a DoS attack on the network even without the need to associate, by sending signals at a high radio power on the correct frequency (physical layer DoS). Hence, allowing anyone to associate is unlikely to cause greater harm to the network provided that a proper VPN is built on top of the wireless network and all other traffic is ignored. Obviously the network security now relies on how secure the VPN software is.

Another issue to avoid is having the network spread too much in physical size outside the site boundaries where it is not needed. Administrators should regulate transmission power on the access points and potentially select appropriate antennas in order to make sure that the radius of the network is only as large (and not much larger) than the radius of the site on which the network is intended to operate.

Recovering WEP Keys

Wired Equivalent Privacy (WEP)

The fact that WEP is insecure is quite well known — contrary to considerations in other forms of cryptography, a 128 bit key will require weekly changes if even a moderate degree of security is desired. Fortunately, the details of how practically to exploit the vulnerabilities of WEP are somewhat less widely known (though easily locatable for those so minded). There are two main branches of WEP attacks: the **weak IV** attacks (dealing with key recovery) and **injection attacks** (sending traffic without knowing the key). The most practical way to break WEP is to use a hybrid of both methods. This requires certain skills which is why, according to many, these attacks remain somewhat theoretical. However, this view is somewhat short-sighted: the current nonexistence of a fully automated tool to recover WEP keys with minimal effort does not mean that the vulnerabilities in WEP are nonexistent or negligible.

The IEEE 802.11 standard specifies WEP with a 64 bit key, though most current hardware also supports 128 bit keys. WEP is an implementation of the **RC4 stream cipher**, which is an encryption algorithm developed by Ronald Rivest [9]. It may be split into two main steps:

- **Key Setup Algorithm** (KSA): establishes a 256 byte state array, which is key-dependent.
- **Pseudo-Random Generation Algorithm** (PRGA): creates a pseudo-random stream based on the state array, without using the key explicitly.

The overall process consists of setting up RC4 with a 64 bit **seed** (key) and generating a pseudo-random stream of the same length as the clear-text. This stream is then XOR-ed with the clear-text to produce cipher-text. XOR is a symmetric operation -- XOR-ing the cipher-text with the same stream allows retrieval of the clear-text.

The term "64bit seed" is used instead of "key" to differentiate between the WEP secret key and the actual seed that is passed to the RC4 algorithm. The secret WEP key is 40 bits long, to which is prepended the 24 bit **Initialization Vector** (IV), making up the full 64 bit seed. The IV is a 24 bit number generated by the sender, that should be unique and re-used as little as possible. However, many implementations currently use a simple incremental counter to generate IVs.

In order to decrypt the data successfully and be able to detect errors, the data portion in WEP packets contains some additional fields. The expansion of the data field in WEP packets may be seen in **Figure 1**.

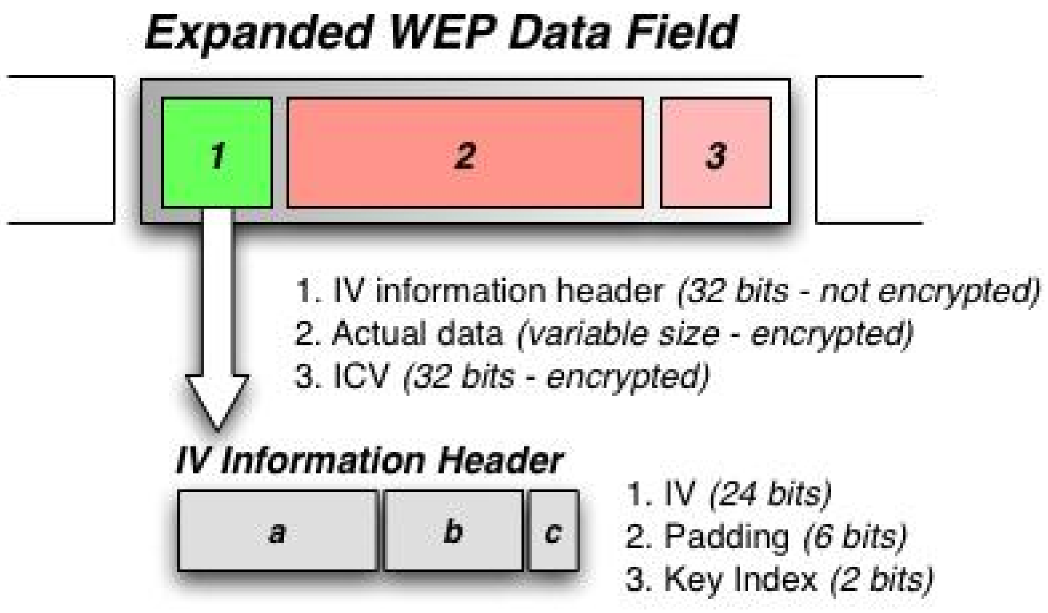
Expanded WEP Data Field
1
2
3
1. IV information header (32 bits - not encrypted)
2. Actual data (variable size - encrypted)
3. ICV (32 bits - encrypted)
IV Information Header
a
b
c
1. IV (24 bits)
2. Padding (6 bits)
3. Key Index (2 bits)

Figure 1: WEP data field.

The IV information is expanded further. The first (left to right) 24 bits are the actual IV used in the encryption. The next six bits are for padding, leaving the last two for the **Key index**. WEP may have up to four (2^2) different keys in use within a single network. The key index indicates which key is being used. The ICV is the CRC32 algorithm run over the clear-text data. Thus, when a packet is decrypted, the CRC32 checksum of the clear-text is calculated and matched with the ICV in order to detect possible errors.

Brute-Force and Pass-Phrases

The secret key is only 40 bits long. On an average modern PC, it takes around a month to search the entire key-space testing possibilities (brute-force). Distributing the work makes this even easier. Also, if Murphy's law does not apply, on average the key will be recovered in half the time. Brute-forcing 104 bit keys is infeasible. In practice, keys are not entered as hex digits all the time, as other mechanisms may be provided instead.

Remembering a 40 bit hex key may be impractical. Many vendors have implemented ways of transforming a pass-phrase into the equivalent hex key. Such algorithms are not standard. For example, by default Microsoft Windows XP maps the ASCII value of the pass-phrase to its equivalent hex value ('A' becomes 0x41, an so on). Furthermore, the pass-phrase has to be exactly either five (40 bit WEP) or thirteen characters long (104 bit WEP). Brute-forcing alphanumeric, five character pass-phrases, takes very little time, and some networks seem to be using them. Other vendors may take some hash of the pass-phrase, although this is still vulnerable to dictionary attacks and possibly even the hash function itself, or its implementation [7]. For example, an attacker may iteratively hash dictionary words and use the result as a possible key, until the correct one is found.

Weak IV Attack

This attack was formalized by Fluhrer, Mantin, and Shamir (FMS) [1] although, chronologically, Wagner [12] first noticed this vulnerability in 1995 (before 802.11 was published). It consists in eavesdropping for packets that use **weak IVs**. A weak IV is a particular IV that will set up the key in a specific way that might give information about a key byte. A single weak IV packet will give about 5% probability of guessing the correct value for a particular key byte. This makes it a statistical attack, as we need to build a table of the most probable candidates for each key byte. FMS recommend collecting about 60 weak IVs per key byte. The attack is also incremental — we need to know the current key byte before we can recover the next (the first key byte may be recovered on its own).

A good implementation of this attack is h1kari's bsd-airtools [2]. There are also classes of IVs that recover the correct key byte with 13% probability, however I am currently unaware of any public tools and documentation that deal with them properly. These IVs are mentioned in [10] although are not thoroughly explained. Some tools attempt to recognize such IVs, but, due to logic errors, never actually find them. However, I have personally analyzed these IVs, and they do occur in practice.

The constraints of this attack are that it is essential to wait for packets that use particular IVs to be transmitted. The frequency of weak IVs may be quite low. Some modern implementations filter out the most obvious weak IVs, but having a single "old" client is enough for it to be a threat in the network, as it will probably transmit using weak IVs without skipping them. Also, not all classes of weak IVs are documented. Instead of waiting for traffic, we may **replay** traffic on the network. For example, we may intercept traffic that seems like ARP requests (identifiable by the length and if the destination is a broadcast) and re-inject it on the network. Hopefully, we will get an ARP reply that will most likely use a different IV. Since the attack is incremental, we may brute-force key bytes for which we do not have enough weak IVs, trading CPU power for eavesdropping time.

Injection Attacks

In contrast to the weak IV attack, injection attacks mainly exploit the PRGA part of the RC4 algorithm. In fact, they do not aim to recover the WEP key, but instead a PRGA stream, which will be used to inject arbitrary traffic without knowing the actual WEP key. In WEP, the PRGA stream acts as a **one-time pad**. The stream is XOR-ed with the clear-text to produce the cipher-text. Knowing that stream will allow us to recover the clear-text (XOR cipher-text with stream). Each PRGA stream is constant and unique to the seed (IV and WEP key pair) that did the key setup (KSA). One reason why WEP introduced IVs is to avoid having a single PRGA stream for the network (the seed is constant as there is no variable IV). Instead, each IV will generate a different PRGA stream. If a PRGA stream is recovered for a particular IV, arbitrary traffic may be encrypted and sent to the network using that IV.

The difficulty in this attack is determining the state of the PRGA. This relies on one knowing the clear-text so that the attacker may XOR it with the cipher-text eavesdropped and recover the pseudo random number used. Consider a simple scenario in which shared key authentication is used. During the challenge/response procedure, the AP sends a clear-text message that the client must encrypt. Eavesdropping this clear-text and cipher-text pair allows one to determine the pseudo random number associated with the IV, key pair used by the client in the response. As argued above, keys themselves change relatively rarely so the security of this algorithm is based entirely on the IV, which is only a 24 bit quantity. Although WEP recommends changing the IV with every frame, this is not always implemented, and, in any case, the IV space is sufficiently small that reuse is practically inevitable.

Anton's tool WEPWedgie [8] can be used in this type of attack. Another interesting feature of this tool is that it tries to establish a two-way communication in the network (note that usually transmission is only permitted in this attack). It does this via a **helper** host on the Internet, assuming the wireless network routes to the Internet, and by sending data with the source IP address of that host. The responses may therefore be monitored from the network on that helper host. The concept is particularly useful for bypassing firewalls which may be installed at the border of the network and not internally, where the wireless network operates as well.

This attack is also possible when open system authentication is used, but this requires more advanced techniques. If the different PRGAs (for the various IVs) are recovered, an **IV dictionary** may slowly be built and decryption of packets that use those particular IVs is possible. Injection attacks are key independent, so having a longer and more complex key does not help. Also, IV filtering, which helps in protecting from weak IV attacks, makes injection attacks easier (e.g., the IV dictionary will be smaller). This is a classic example of where one patch introduces another hole. If shared-key authentication is used, another speedup in building the IV dictionary may be possible. The attacker can force clients to de-associate via DoS attacks. When the client re-authenticates, hopefully a different IV will be used thus adding to the dictionary.

Advice

No matter how WEP is used, it is fundamentally flawed. Some "patches" do make life slightly harder for attackers, but the base protocol is based on a cryptographically weak approach, and typical realizations of this in hardware and firmware weaken this further. The danger of using WEP is that is conveys a false sense of security to the uninitiated.

Thus any real solution to the problem of security must eliminate the problem at its base rather than attempting to circumvent problems in a fundamentally broken protocol. A safe (and quite realistic) assumption is that anyone may read WEP encrypted wireless traffic. Users should, therefore, rely on application layer encryption such as SSL (for web, pop3, etc.), PGP (email messages), SSH (remote login and also provides SSH tunneling for other protocols), and others.

A cleaner solution is, as previously mentioned, the use of a VPN. Most VPN solutions provide encryption mechanisms that are highly secure. For example, OpenVPN [**14**] provides certificate-based public key encryption (making it hard to spoof identity). IPSec [**4**] is another widely used solution for encrypted traffic.

Generic LAN Attacks Applied to Wireless Networks

Man in the Middle Attacks

Some people may think that the worst that can happen is that the intruder might use the Internet or damage the wireless network itself, leaving the wired LAN unaffected, as if it was independent. With the ability to send ARP packets and being able to eavesdrop, **man in the middle** attacks become very practical. A specific example relies on modifying the **ARP caches** of the various machines. This may be done by forging fake ARP replies or requests. Once the ARP cache is modified, the host will send data to the attacker's MAC address, thinking it is the location of the wanted IP. This technique is known as **ARP spoofing**.

The most obvious reason for performing these attacks is to make possible the eavesdropping of a wired LAN. Consider a user, Alice, on the wired LAN, wanting to communicate to Bob on the Internet, via a wired router R (**Figure 2**). Also consider a malicious user Eve on the wireless LAN bridged to Alice's wired LAN. Eve will try to **poison** Alice's ARP cache, mapping R's IP address to Eve's MAC address. The result being that traffic meant for R will be directed to Eve, which will be routed through to the wireless LAN. Once the traffic reaches Eve, she will monitor it, changing it if needed, and send it back to R, which eventually routes it through the Internet to Bob. In order to get the traffic going from Bob to Alice, the ARP cache of R must be poisoned in a similar way (mapping Alice's IP to Eve's MAC address).

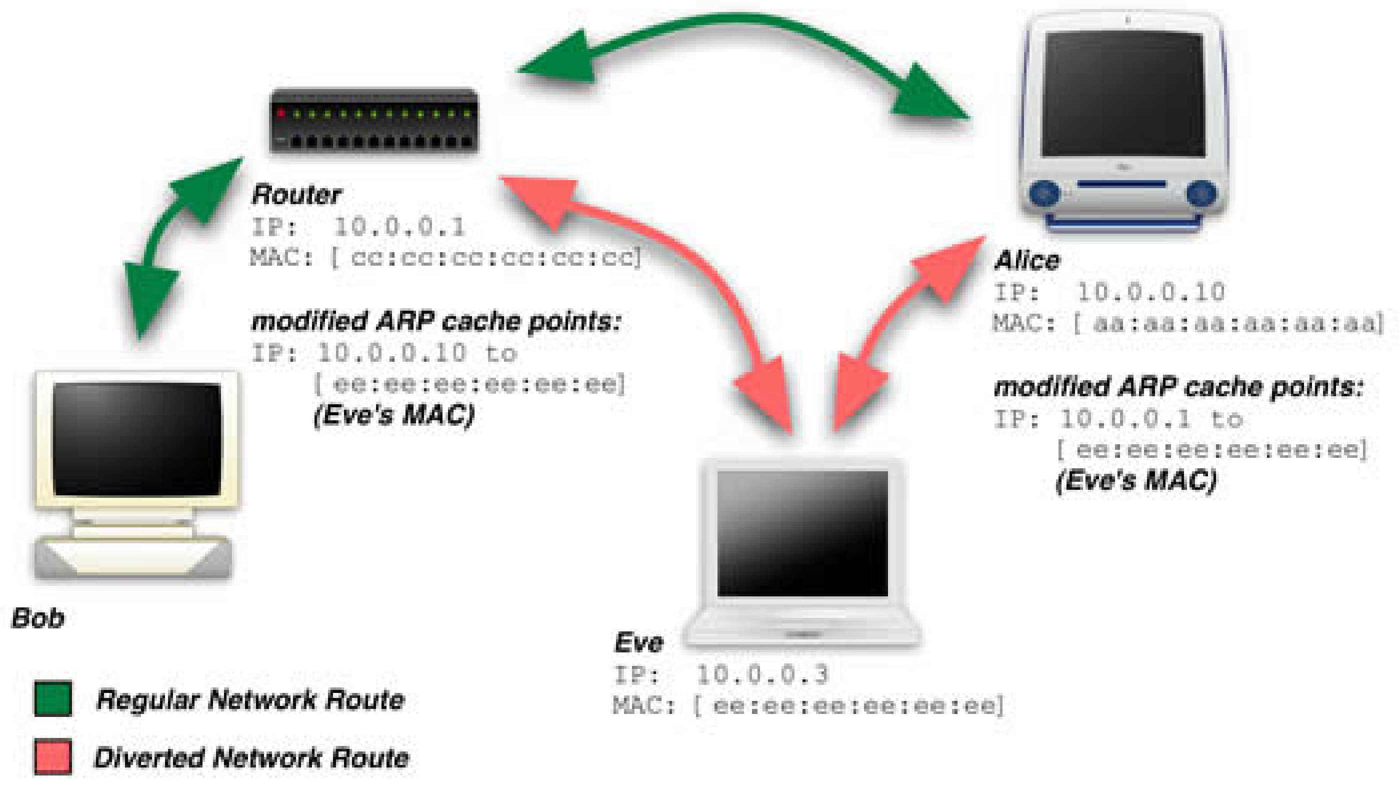

Figure 2: Man in the middle attach via ARP spoofing.

Another equivalent attack is impersonating actual servers. The attacker may redirect all traffic meant for a web server to him/herself to intercept login passwords. Although many wireless networks are open, they require further authentication via a web page, say, before traffic may be routed on to the Internet. Sometimes they use HTTPS but with self-signed certificates. Many users blindly accept certificates per session, so recovering passwords in these cases may be done with some imagination (fake certificates and public keys). Also, after successful authentication, state may be maintained via the IP and MAC address pair. The attacker may impersonate an authenticated user (e.g., by using his IP and MAC address) and reroute the original user to a non-existent host, effectively ending up in a DoS attack against the real user. Impersonating the local Domain Name Service (DNS) server might be another interesting attack as redirection of clients at a hostname level is achievable.

DNS Tunneling

This last section discusses a possible way in which un-authenticated users may route to the Internet in particular circumstances. Some setups, mainly found in hot-spots and large companies, authenticate users at a later stage with a different mechanism, rather than relying on 802.11. Consider a fully open wireless network with DHCP, where, as soon as you try to visit any web page, you are redirected to some login page where authentication details must be provided. Prior to authentication, all Internet traffic will be denied by the local gateway. The idea of this attack is to exploit the ability to send DNS queries through the network's local DNS server in order to communicate with a party (i.e., server) that is under our control, and potentially, through this server, freely access the Internet. Messages destined to or coming from this server are encoded in DNS queries and replies.

DHCP normally gives the IP address of a DNS server (or one may be found also by scanning the network). These DNS servers are sometimes fully functional (i.e., any Internet hostname may be resolved into its correct IP address). This means that data is sent and received from the authoritative name-server of a particular domain-name. Therefore a **tunnel** may be formed by **encapsulating** the data within the queries, and receiving the data from the replies. The tunnel endpoint will be the authoritative name-server of the domain-name that is looked up in the queries.

Let us suppose the attacker owns domain.com and its authoritative name-server ns. domain.com. If he/she wants to send the message "foo" to ns.domain.com, he/she may encode it in a host lookup query such as 666F6F.domain.com. In this particular example, the encoding used was the ASCII hex value of the string "foo." If, however, ns.domain.com wants to send the attacker some data, there is no way it can contact him/her unless he/she sends it another request (reception will have to be based on **polling**). He/she may, for example, send a host-lookup for getdata.domain.com. If the reply is an IP address, then there is no data. Conversely, if the reply is a CNAME to some other host like 626172.domain.com, then he/she knows he/she received the data unit 626172, which corresponds to "bar" according to the simple ASCII encoding scheme. This example is simplified for clarity, as there are obvious ways to make the encoding scheme better (data compression and use of other DNS records such as TXT). Also DNS caching must be avoided (low TTL values and unique lookups) in order to receive correct data all the time.

Notice that this is different from sending data that appears to be similar to DNS queries on the Internet, such as encapsulating traffic in UDP packets with a source port of 53 and trying to bypass an insecurely set up firewall. In DNS tunneling, the attacker is tunneling above the application layer, and all the traffic is directed to the local DNS server of the wireless network. The quality of service will be poor, but it is a proof of concept. A tool that claims to realize this is nstx by Heinz and Oster [3].

Advice

One way to prevent ARP spoofing is using static ARP entries. This is rather inflexible, however. In fact, it is probably only realistic for clients to set a static ARP entry for their default gateway. This implies that all outgoing Internet traffic cannot be re-routed. There are also numerous tools that will alert users when hosts change MAC address; in fact, the attack is highly detectable but harder to prevent.

Making sure un-authenticated users may only resolve internal DNS hostnames and IPs is generally a good idea. Extending the VPN solution proposed above, the network may have two DNS servers. One external to the VPN which may, for example, resolve all hostnames to the IP of a web server that gives more information on how to login to the VPN. The second DNS server will be internal to the VPN and fully functional. This implies only authenticated users will have full DNS functionality.

Conclusions and Possible Solutions

In this article I have presented some of the possible attacks that can be carried out on wireless networks. I have also described how wireless networks could be seen as a potential threat to the wired infrastructure. The vulnerabilities in wireless systems tend to be numerous because of the inherent lack of physical security; being able to monitor and interfere with traffic makes attacks more practical. As presented above, the default encryption mechanism, WEP, contains severe and fundamental flaws. Overreliance on WEP thus leads to a false sense of security.

The attacks described in this article are the most common and only a subset of those possible. Other vulnerabilities exist, especially in WEP, but require the use of greater knowledge and skill in turning them into serious real-world threats. Once associated with the network, various LAN attacks are possible. Combining various attacks gives great flexibility in what can be done, especially with some imagination.

Despite all the security issues currently present, wireless networks are the future; however, people will fear using them if they perceive a substantial threat to their privacy or to sensitive information. It is the administrator's responsibility to make legitimate clients feel safe and confident in the use of a service. Security can never be perfect, especially in large networks, but reliance on mechanisms that are known to be broken is lazy and carries the danger that one's supposedly secure network becomes a playground for those who only know how to download the latest security breaking tool from the web.

Acknowledgments

I would like to thank Dr. Cecilia Mascolo for the patience and time she put into reading and commenting my text. Many thanks go to Dr. Stephen Hailes who made corrections and polished the text. Also, I would like to thank Adam Cimarosti for dealing with the graphics of this article. Finally, a big thank you to my sister Polina who read through the article no matter how obscure the subject was for her.

References

1
Fluhrer, S. Mantin, I., and Shamir, A. "Weaknesses in the Key Scheduling Algorithm of RC4." Lecture Notes in Computer Science, vol. 2259, pp. 1-24. 2001.

2
Hulton, D. bsd-airtools. **http://www.dachb0den.com/projects/bsd-airtools.html**

3
Heinz, F., Oster, J. NSTX. **http://nstx.dereference.de**

4
IP Security Protocol. IETF. **http://www.ietf.org**

5
Kershaw, M. Kismet. **http://www.kismetwireless.net**

6
Lynn, M. Airjack. **http://sourceforge.net/projects/airjack/**

7
Newsham, T. "Cracking WEP Keys, Applying known techniques to WEP Keys." 2001. **www.lava.net/~newsham/wlan/WEP_password_cracker.pdf**

8
Rager, A. WEPWedgie. **http://sourceforge.net/projects/wepwedgie**

9
Rivest, R. The RC4 Encryption Algorithm. 1992.

10
Stubblefield, A. Ioannidis, J., and Rubin, A. "Using the Fluhrer, Mantin, and Shamir Attack to Break WEP." ATT Labs Technical Report, TD4ZCPZZ, Revision 2, August 21, 2001.

11
THC-HuT. The Hackers Choice. **http://www.thc.org**

12
Wagner, D. (sci.crypt) Re: Weak keys in rc4. **http://www.cs.berkeley.edu/**

~daw/my-posts/my-rc4-weak-keys. 1995.

13
Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard 802.11, 1999.

14
Yonan, J. OpenVPN. http://openvpn.sourceforge.net

Biography

Andrea Bittau (a.bittau@cs.ucl.ac.uk) is a second year BSc Computer Science student at University College London. His main interests lie in the field of computer security. His experience includes networking and analysis of vulnerable software. Lately he has been investigating vulnerabilities of 802.11 wireless systems and spent a great deal of time analyzing flaws in the WEP algorithm.