Hal Berghel and Jacob Uecker

WiFi Attack Vectors

The security risks of WiFi connectivity have been established—the reasons for these risks of intrusion are not as well understood.

Most of the comments we received on our recent "Wireless Infidelity" columns (Sept. and Dec. 2004) related to the technical details of breaking 802.11 encryption protocols. While many readers may know that WiFi is vulnerable to hacking, far fewer know why. This column will hopefully fill that gap. To forestall possible initially reactive feedback, we emphasize there is nothing we're going to explain that isn't already understood and put into practice by the hacker and criminal communities. The people in the dark tend to be the law-abiding citizens. Perhaps this column will help level the playing field so that defenders have a better chance of protecting their digital assets against WiFi intrusion.

FMS Attack Vectors

The genesis of the wireless insecurity problem was the 802.11 standard. The vulnerabilities were built into the protocols. Nowhere is this more evident than in the bungled implementation of the RC4 symmetric, stream cipher algorithm in the implementation of Wired Equivalent Privacy (WEP). The WEP implementation of RC4 is flawed in several ways: the Initialization Vector (IV) that is always pre-pended to the key prior to generation of the keystream by the RC4 algorithm is transmitted in cleartext; the IV is relatively small (three bytes), which produces a lot of repetitions as the scant 16.77 million variations are reused to encrypt millions of packets; and some of the IVs are "weak" in the sense they may be used to betray information about the key. It should be noted that even if the implementation of RC4 was corrected, WEP would still be vulnerable to replay attacks, checksum forging, message integrity check forging, and sundry authentication attacks resulting from the fact that both

DAVID GOLDIN

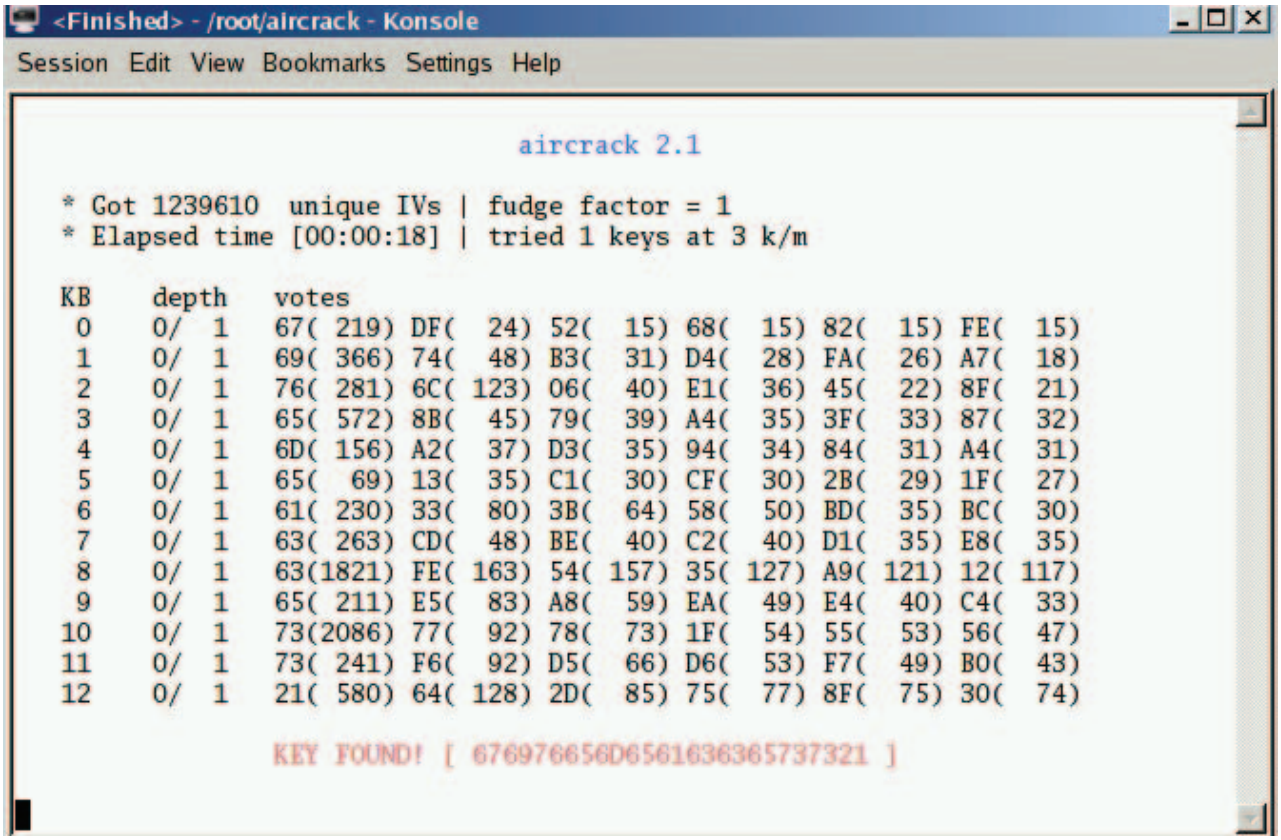


Figure 1. Aircrack Success. 13-byte (104-bit) WEP key found in 18 seconds with 1,239,610 unique IVs—far more than usually required.

the plaintext challenge and cipher text response are broadcast.

Here's an example of IV weakness. Since the first data to be encrypted in a WEP packet is usually the SNAP header (as with IP and ARP packets), the first byte of this header is almost always 0xAA. A weak IV has a format of B+3::ff::X (where B is the byte of the key to be found, ff is the constant 255, and X is irrelevant). Since the IV is transmitted with the packet in plaintext, weak IVs are easy to detect. The key value of B is determined after the B+3rd iteration of the key scheduling algorithm. Given a sufficient amount of traffic, and repeated applications of this strategy, we can recover the entire key. Since the IV is a 3-byte string, there are 2**24 possible values; it has been estimated that about 2% of these are weak. Empirical studies show it only takes a few hundred packets encrypted with weak IVs to crack the encryption. The question isn't whether WEP can be broken, but how long it takes. As a rule of thumb, a few million packets generate enough weak IV traffic to recover 40-bit WEP keys.

We should mention that the IEEE standard for IV selection was ambiguous, so many wireless vendors use sequential IV generators that begin with 00:00:00 and wrap with FF:FF:FF. This is the worst of both worlds. Not only is this procedure guaranteed to generate weak IVs, but it does so predictably. Randomized IV selection is a better idea, but as we shall see also fails to solve the problem. Utilities that break WEP encryption by taking advantage of weak IVs are called "FMS utilities" after the famous article by Fluhrer, Mantin, and Shamir (see "URL Pearls" at the end of this column for more information) that described several classes of weak

IVs (aka "interesting packets"), one of which is the "B+3:FF:X" packet described earlier. Airsnort (airsnort.shmoo.com) is an example of a weak IV encryption cracking utility that targets B+3:FF:X-type packets. Since modern WiFi cards and appliances reduce the percentage of weak IVs that are generated (under the rubric of "WEP+" or "Advanced WEP Encryption"), Airsnort is declining in importance as it takes an unreasonably long time to collect enough packets to break keys.

However, recall that there are several classes of weak IVs. The B+3:FF:X-type is just one. Newer utilities like Aircrack (see www.cr0.net:8040/code/network/) and WepLab (see weplab.sourceforge.net/) use a more sophisticated approach to span a broader spectrum of FMS weak IV classes. Aircrack does not crack live traffic on-the-fly as Airsnort does, but looks for a larger range (five million) of weak IVs. Further, Aircrack is capable of distributing the workload over multiple processors for efficiency. Figure 1 illustrates the key recovery process using Aircrack 2.1.

Beyond FMS Attacks

Like Aircrack, WepLab also uses a broader attack vector than Airsnort. While it includes FMS attacks, it also incorporates brute force, dictionary, and an implementation of the "Korek algorithm." Some estimate that WepLab can break 40-bit WEP keys in 100,000 packets or less and 104-bit WEP in 300,000 packets or less (see the sourceforge Web site).

Other WEP weaknesses that may be exploited include defective key-generation implementations. Replay attacks result when wireless traffic is captured and retransmitted by foreign wireless appliances thereby increasing the volume of traffic. The traffic volume might be increased slightly to lessen the time required to capture sufficient packets to break the WEP key quicker than normal traffic would allow. On the other end of the spectrum, the volume of traffic could be maximized to the point where a denial of service results—much like a ping flood. Aireplay (see www.cr0.net:8040/code/network/aircrack/) is one such tool for relay attacks. It harvests ARP-like packets from captured ("sniffed") Pcap files, and then rebroadcasts them indefinitely to increase the traffic flow.

A variation on this theme is a PRGA injection attack. In WEP, the IV, the key, and the data length value are all used by a pseudo-random number generating algorithm (PRGA) to generate a pseudo-random string exactly the same length as the plaintext data to be encrypted. The ciphertext is the result of XORing this string with the plaintext. If one knew this string, deciphering the text would be trivial because the IV and ciphertext appear in captured packets. Attack vectors that exploit this vulnerability are called PRGA injection attacks. Here's one such strategy.

Suppose I want to connect to an enterprise network through a wireless access point. The emphasis is on the connection itself, and not decryption of packets. What I need is some way of conveying a legitimate packet onto the network. My bottleneck is that I don't have any way of authenticating myself as an authorized user. But if I could just craft a SYN packet with a spoofed IP address to a machine under my control and get the enterprise network to accept it, I could establish a connection. I can use the WAP to help because of its anemic authentication standards.

The PRGA injection works like this. The hacker sniffs WEP challenge/response authentications between the WAP and clients. The challenge is in plaintext. The response is an encrypted version of the challenge. If we XOR the challenge with the response, the result is the PRGA. Armed with the PRGA and four bytes of control information, one may craft packets at will. Add to that a spoofed, known acceptable IP address for protection, and one can move through the wireless fabric like Code Red through a straw. The only caveat is that the crafted packets must reuse the same IV. So long as IV reuse is accepted by the wireless appliances, this is not a liability. Enter WEPWedgie (see sourceforge.net/projects/wepwedgie/).

WEPWedgie is an automated tool for PRGA injection. It consists of two modules: prgasnarf, which collects WEP

challenge/response authentication; and wepwedgie, which crafts the packets (for example, the SYN packet described two paragraphs earlier). The actual injection is accomplished by a sister product called Airjack (see sourceforge.net/projects/airjack). Were one so inclined, WEPWedgie can also be modified to generate a plethora of such crafted packets to create a broadcast flood.

One remaining attack vector of note is the dictionary attack. In WEP, a passphrase is used to generate the four WEP keys. The packet key is the result of appending one of the four WEP keys to the IV. In plain terms, if one knows the passphrase and the algorithm used to generate the WEP keys, and the IV that is broadcast in every encrypted packet in plaintext, one can re-create the packet keys. Many popular WEP WiFi products use the Neesus Datacom key-generation algorithm for the creation of the WEP keys from the passphrase. A dictionary attack results from running the Neesus algorithm against a list of common passphrases. The result is a file of WEP keys that may then be compared against the transmitted data for a match. The details of how to optimize the match is beyond the scope of this column, but suffice it to say that the utility WEPAttack (wepattack.sourceforge.net) is designed for just this purpose.

The question isn't whether WEP can be broken, but how long it takes.

But There's WPA, You Say

Given the swiftness with which the hacker community compromised WEP, the Wi-Fi Alliance made damage control a top priority. In doing so, they faced two fundamental issues: they needed to develop a fix immediately if not sooner; and they had to stay within the vendor's capabilities to implement it as quickly as possible even on legacy hardware. So, by mid-2003, the Wi-Fi Protected Access (WPA) standard was implemented and marketed by the more aggressive vendors.

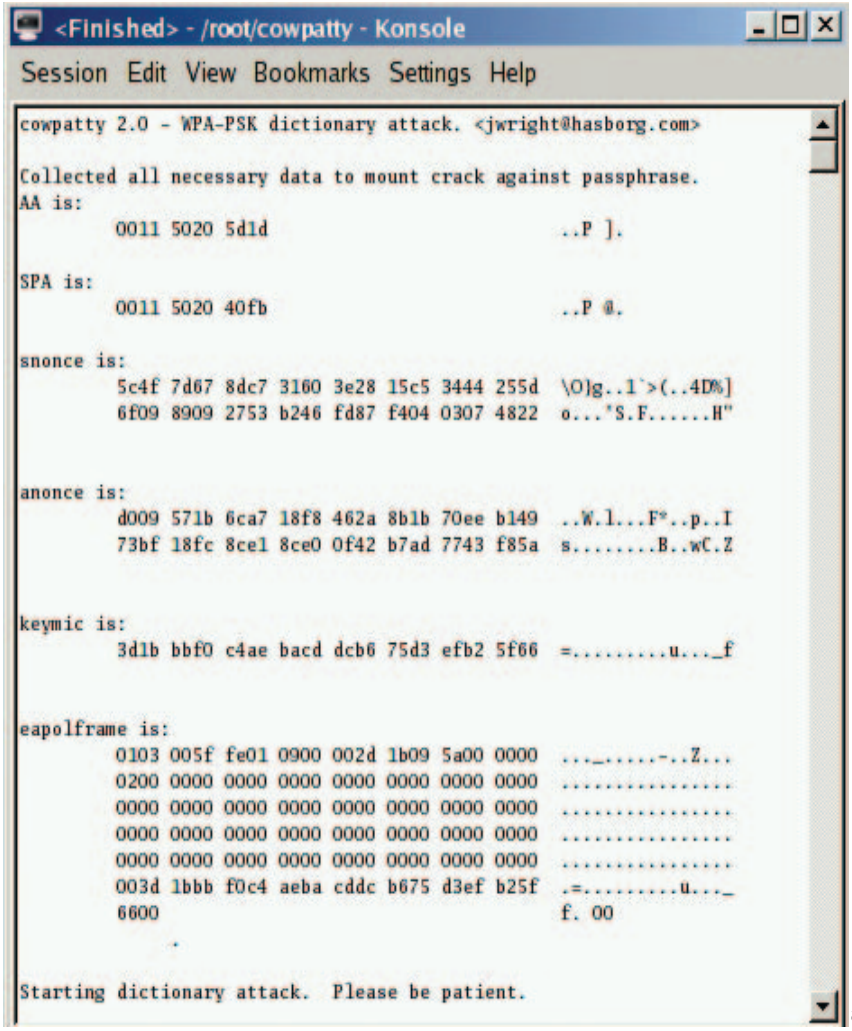
WPA sought to overcome the critical deficiencies of WEP with several changes. We list six here along with the attack vectors mentioned previously that they mitigate against:

1. The minimum key length was increased from 40 to 256 bits (dictionary attacks);
2. The IV length was doubled (FMS attacks);
3. IV sequencing was enforced (replay attacks);
4. Key rotation was embedded automatically (FMS, PRGA, and dictionary attacks);
5. Mutual authentication was built in so that both the WAP and station had to prove to each other that they were legitimate (spoofing); and
6. Packet tampering detection was built in with a Message Integrity Code (PRGA Injection)

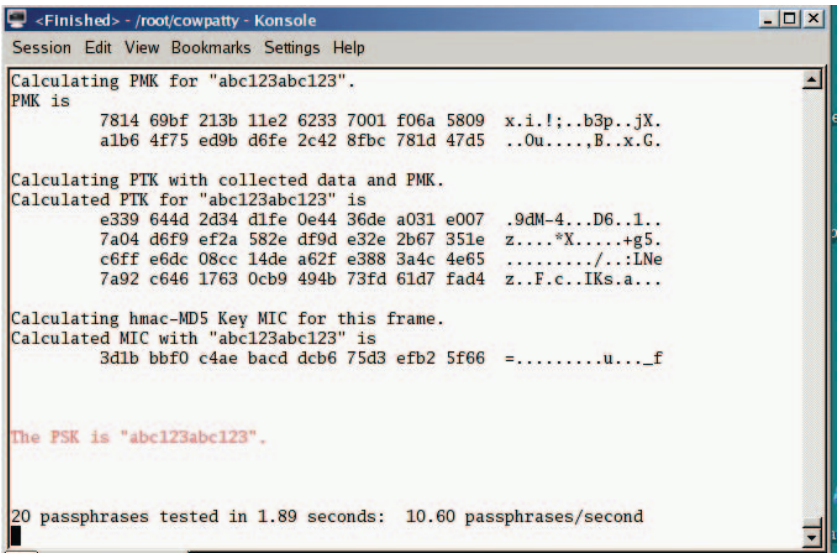
There were other changes as well. Life was good for a few months.

WPA is an improvement to be sure. However, an early vulnerability arose from the looseness in the way the pre-shared keys (PSK) were used. The WPA-PSK implementation was meant as a surrogate for authentication servers (aka Radius servers), which are uncommon in the SOHO market. As with WEP, users simply enter the same passphrase on WAP and client, and the authentication is transparent. Unfortunately, passphrases less than approximately 20 characters gave rise to a new WiFi attack vector: hash comparison attacks.

When WPA-PSK is implemented, the passphrase that is entered both on the client and WAP side is run through a series of computations to produce a set



a.



b.

Figure 2a. coWPAtty finds all the data necessary to find the passphrase in four packets.
Figure 2b. Using coWPAtty to find the WPA pre-shared key.

of keys. It is these keys that are used to encrypt traffic and verify its integrity. When users enter a passphrase, they are directly converted to hexadecimal (and by extension, binary). The entropy of the bytes is quite small. In other words, the values are anything but random. So to add some entropy to the passphrase, it is hashed along with some other session variables like SSID and SSID length. This hashing operation is done 4,096 times to derive what is called the pairwise master key or PMK. This PMK is not yet the key that is used in the encryption. The PMK is actually used to generate the pairwise transient key, (PTK). It is the hacker's job to find this PTK. If that can be found, it can be reverse engineered back to the PSK. Once that is found, the hacker can join the wireless network.

One of the newest WPA-PSK cracking utilities is called coWPAtty (see new.remote-exploit.org/index.php/Codes_main). It goes through the process of finding the PTK for every word or phrase in a dictionary and checking to see if that PTK generates the correct Message Integrity Check (MIC) value for a given packet. To do this, it gathers the four-way handshake that constitutes a WPA authentication sequence. From this series of packets, it first finds the SSID of the network that is

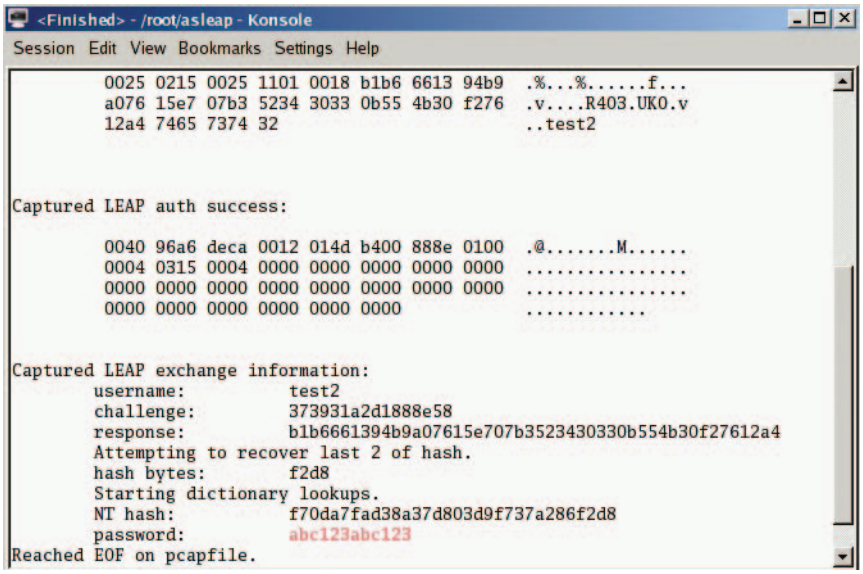


Figure 3. Using asleap to find the LEAP password.

needed in the hashing algorithm to find the PMK. To find the PTK, the "nonces" (random values) and MAC addresses that are used in the four-way handshake are needed. Once they are found (see Figure 2a), they are used with the PMK to find a PTK value. Finally, this value can be used to find the MIC of a packet. If the calculated MIC matches the MIC given in the packet, the correct passphrase has been found. If not, the process is repeated with the next dictionary word/phrase. Using coWPAtty, we were able to break WPA in less than a minute (see Figure 2b). Part of the reason for this speed is only authentication frames are necessary. And if the four handshaking packets proved difficult to come by, other WiFi tools exist that force de-authentication, so the legitimate user has to re-associate and re-authenticate on demand! Now the hackers have all the packets they need.

The solution to this problem is long, complex passphrases. According the Moskowitz (see wifinetnews.com/archives/002452.html), after 20 characters the passphrase would begin to be difficult to break. Some authors recommend random passphrases twice that length. But how likely do you think it is that the typical SOHO user has shared secrets that look like ikd8Jue*#^&hfda;lnvc74793-KDie40I#$(*$d? Remember that even if the length is satisfactory, the strength of the passphrase is undercut if it consists of dictionary words—and hackers seem to think of everything when they build their dictionaries. So don't even think about pA$$w0rd. One final thought. Look for an abundant supply of WPA cracking utilities to appear in the next 12 months. The best protection against them will include selecting AES instead of RC4 if that is available on your WPA-compliant appliance and to use very long, complex passwords (WPA supports passwords from 8 to 63 characters, so 63 should be your target).

Leap of Faith

We'll get to the "L" in a moment. First, EAP stands for

WiFi will continue to be more vulnerable to attack than hardwired LANs as long as electromagnetic radiation fails to obey property lines.

extensible authentication protocol. 802.1x provides a structure for allowing layer 2 access to a network. This is done using three parties: the supplicant, authenticator, and the authentication server. The supplicant is the device that wishes to have access on the network. In the wireless world, it's usually a client computer wishing to connect to the wireless network. The authenticator is the device that allows or denies the actual access to the network—the wireless access point. Any access requests from the supplicants are sent to the authentication server through the authenticator to see if they can be allowed on the network. The authentication server will determine whether the supplicant should be allowed access to the network and inform the authenticator of the decision. Depending on the decision, access can either be granted or denied. The client can also authenticate the authentication server, which helps thwart man-in-the-middle attacks. So under 802.1x, both end points should in principle be confident of the legitimacy of each other.

What 802.1x doesn't specify (deliberately) is the mechanism as to how the authentication server will determine whether the supplicant should be allowed on the network. This is where EAP comes in. EAP is provided to create a channel through which the supplicant and the authentication server can exchange their creden-

URL Pearls

The theory behind FMS-type attacks is described in the classic article that started the world of WEP cracking: "Weaknesses in the Key Scheduling Algorithm of RC4" by Scott Fluhrer, Itsik Mantin, and Adi Shamir, which is abundantly available via Web search. For a quick hop, see www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf.

Airsnort is available from The Schmoo Group (airsnort.shmoo.com/). AirCrack (www.cro.net:8040/code/network/) is the latest and greatest WEP cracking package. Weplab (weplab.sourceforge.net/) combines brute force, dictionary attacks, and statistical methods to find the WEP key. These products support the new KoreK methodology, which can be seen in program chopper (www.netstumbler.org/showthread.php?t=11878&page=2; you must register to get the download). A packet-by-packet decryption technique has also been created and implemented in the program chopchop (www.netstumbler.org/showthread.php?t=12489; you must register to get the download).

WEPwedgie (sourceforge.net/projects/wepwedgie/) allows traffic generation on an encrypted wireless network through either the Internet or a wireless client.

WPA crackers will gain popularity and influence as WPA grabs more of the market but for now, here are some tools: WPA Cracker (www.tinypeap.com/html/wpa_cracker.html), and coWPAtty (new.remote-exploit.org/index.php/Codes_main).

LEAP crackers: leap (packetstormsecurity.nl/0310-exploits/leap.tgz), anwrap (www.securiteam.com/tools/6O00P2060I.html), THC-LEAPcracker (www.thc.org/download.php?t=r&f=thc-leapcracker-0.1.tar.gz), and asleap (asleap.sourceforge.net).

To create a dictionary for dictionary attacks, John the Ripper is state of the art (www.openwall.com/john/).

For further reading on wireless insecurities, check out Wi-Foo (www.wi-foo.com). A book that was just released as this column went to press is *Network Security Tools* by Nitesh Dhanjani and Justin Clarke; the later chapters provide some useful technical information on WiFi hacking.

For anyone seriously interested in this topic, the best resource is the hands-on SANS course on Auditing Wireless Networks (www.sans.org) written, and occasionally taught, by Joshua Wright. Attendees have the opportunity to work with many of the tools and techniques mentioned here.

tials. These credentials are determined by the various EAP types—one of which is the lightweight version—hence the 'L'. It's up to the EAP type to provide the security between the two parties, this is where the attacks happen. Let's take a look at the case of LEAP.

Although LEAP is only available on Cisco or some Linksys (as they are part of Cisco now) access points (that would be the authenticator part), it has become the most popular EAP type. Cisco developed the technology and allowed wireless card vendors access to the technology. As a result, nearly all wireless client cards support LEAP (that's the supplicant side). Unfortunately, it is also fundamentally flawed due to its usage of MS-CHAPv2. This algorithm, and specifically the way it is implemented in EAP/LEAP, allows an attacker to perform an offline attack to determine the password. When the usage of EAP-LEAP has been agreed upon, the authentication server sends the supplicant (by way of the authenticator) a nonce, or challenge text. Specifically, it is an 8-byte random stream the supplicant must encrypt. To encrypt the challenge text, the password is hashed using an NT hash and split up to generate three separate keys. The first key consists of the first seven bytes of the hashed password, the second key is the second seven bytes of the hashed password, and the third key is the final two bytes followed by five NULL values. These three keys are each used to encrypt the 8-byte challenge text. The three 8-byte results are then concatenated into one 24-byte value. This value is sent back to the authentication server for verification. Since EAP/LEAP supports mutual authentication, the process is repeated in the opposite direction to authenticate the authentication server with the supplicant.

The key to breaking EAP/LEAP is the fact that NT hashing does not use "salt." That means that the same plaintext value will always hash to the same hashed value. So an attacker can hash a dictionary of plaintext passwords and store the corresponding hash values. If the password is one of the dictionary words, the hashes will match. Since the third hashed value that is used as a key to encrypt the 8-byte challenge consists of five null values, there is really only 2**16 different values that the key could be. With so few possibilities, it is possible to find the two bytes in less than a second. At this point, the last two NT hashed bytes of the password have been recovered. Using the precompiled dictionary, the attacker finds all hashed passwords in which the last two bytes match what has been found. This usually limits the possible passwords to a number that can be brute forced against the authentication server. Now the attacker can achieve access to the wireless network.

There are a number of utilities that can perform this attack. The most famous of which is asleap (see asleap.sourceforge.net), developed by Joshua Wright. Figure 3 shows the output of asleap when discovering the LEAP password of a user. Notice the challenge/response values, hash lookup bytes, and corresponding dictionary hashes are prominently displayed.

Conclusion

We leave where we began with the hope that this treatment of WiFi attack vectors will simultaneously encourage you to deploy wireless technology sensibly, be cognizant of the inherent risks, and minimize your vulnerability by taking advantage of the security protections available. WiFi will continue to be more vulnerable to attack than hardwired LANs as long as electromagnetic radiation fails to obey property lines. C

Hal Berghel (www.berghel.net) is associate dean of the Howard R. Hughes College of Engineering at the University of Nevada, Las Vegas and Erskine Fellow at the University of Canterbury. He is also the director of the Center for Cybermedia Research and co-director of the National Identity Theft and Financial Fraud Research and Operations Center.

Jacob Uecker (jacob@juecker.net)is a research assistant at the University of Nevada at Las Vegas Center for Cybermedia Research and the National Identity Theft and Financial Fraud Research and Operations Center.