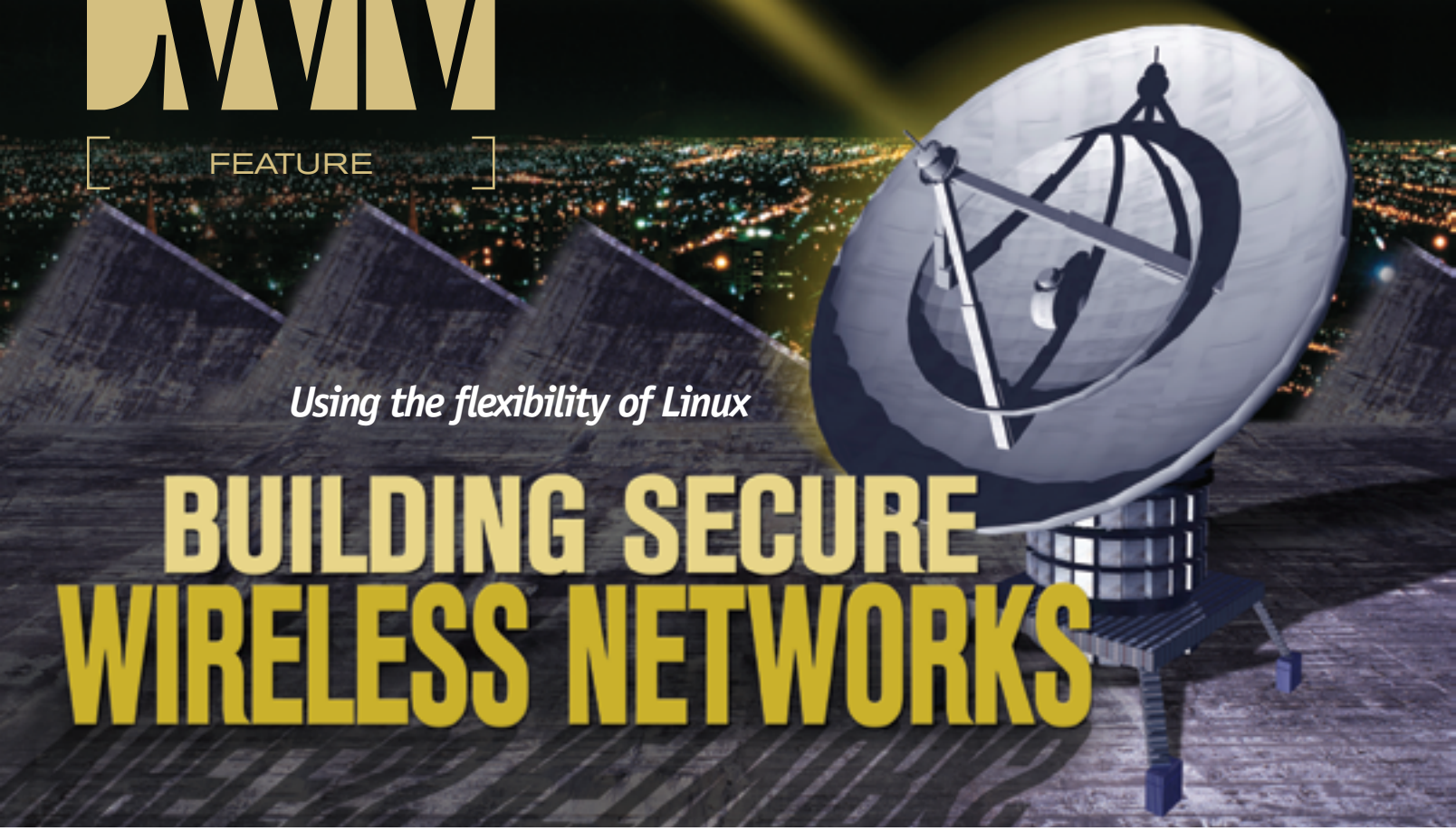

Using the flexibility of Linux

BUILDING SECURE WIRELESS NETWORKS

BY ANDREW A. VLADIMIROV, KONSTANTIN V. GAVRILENKO, AND ANDREI A. MIKHAILOVSKY

While many paranoid system administrators and users still consider any WLAN to be a gaping hole, these networks can be successfully secured against snooping and unauthorized access with a little thought and effort. Fortunately for us, Linux provides some flexibility when it comes to choosing a wireless safeguard. While it's the ultimate wireless attacker's platform, it's also the optimal system to rely on when deploying a hardened WLAN. This is not surprising, if you consider that network attack and defense are two sides of the same coin.

This article describes the security issues facing the modern 802.11 networks and the solutions available to mitigate these problems using the Linux platform.

Main Security Problems

The number one wireless security problem in the real world is the ignorance of the users and system administrators. We have wardriven for several years in different parts of the world collecting statistics about all discovered WLANs. Unfortunately, the percent of completely open WLANs (roughly 70% of all found networks) still remains the same. It doesn't matter how good the industry-provided safeguards are; they're entirely useless if not turned on and properly configured, and that's what we see on every corner of every street we pass by. Some of the open access points we saw were clearly Linux HostAP-based, so Linux users are not spared and can be just as security-ignorant as well. In fact, there are several types of ignorance that make WLANs an easy prey for attackers on the streets or obnoxious neighbors:

- Complete lack of knowledge of Layer 1/radio frequency operation. Not knowing how far the signal can spread from the intended coverage zone and how far away a prepared attacker can pick it up and abuse it is probably the main reason for leaving all these completely unprotected WLANs around.
- The lack of understanding about Layer 2 wireless security – there are still people who believe that WEP, closed ESSIDs, and even MAC address filtering are sufficient to stop the attackers cold (no, this is not a joke).

NEED HEAD SHOT

ABOUT ANDREW A. VLADIMIROV

Dr. Andrew A. Vladimirov is a graduate of Kings College London and the University of Bristol. Andrew is a researcher with a wide area of interests ranging from cryptography and network security to bioinformatics and neuroscience. He published his first scientific paper at the age of 13 and is one of the cofounders of Arhont Ltd. Furthermore, Dr. Vladimirov was one of the first UK IT professionals to obtain the CWNA certification and is currently in charge of the wireless consultancy division within Arhont Ltd.

andrew@arhont.com

- System administrators who are clueless about so-called "rogue" wireless devices being installed on their (not obviously wireless) networks by unruly users or even serious crackers using such appliances for out-of-bound backdoor access.

While everything mentioned above is related to the low level of user education and network mismanagement, there are unfortunately a few purely technical problems related to 802.11 security. First of all, 802.11 management frames are still not authenticated. The 802.11 "I" task group, assigned with improving wireless security, tried to implement 802.11 certain frames authentication but did not succeed. Thus, any 802.11 WLAN can be easily DoS'ed by flooding with spoofed deauthentication or deassociation frames. Such floods are more than a mere nuisance since they can be used as an integral part of the man-in-the-middle and even social engineering attacks. The only thing you can do is install wireless IDS that will detect the flood, spot the attacker physically, and scare them away.

Second, the equipment supporting the recently adopted 802.11i wireless security standard practically implemented by the Wireless Protected Access (WPA) Industry Certification still suffers from vendor-interoperability problems, despite WPA version 1 being a part of the Wi-Fi Certification now. This presents a serious challenge for multivendor wireless networks, such as public hot spots relying on users bringing their own cards. Finally, the 802.11i wireless security standard is actually more like a set of standards, and some of these standards have well-known weaknesses, e.g., lack of mutual authentication in EAP-MD5. Besides, even when the standard design is solid, there are always bad implementations that nullify the advantages it presents.

Why and How Crackers Exploit WLANs

Knowing your enemy is an absolute requirement of proper network protection, and penetration testing should always be your first line of defense. It's highly suggested that as a system administrator or wireless community guru you spend some time trying to exploit your own WLAN. If you are an IT security professional, it's always good to participate in ethical wardriving to see what's really happening on the "wireless front lines" despite many "armchair expert" opinions. This is why we wrote *Wi-Foo: The Secrets of Wireless Hacking*. Since the final manuscript was submitted, nothing has changed when it comes to wireless attacker motivation and type. People still attack WLANs seeking fully anonymous access (no ISP logs) to hide their tracks, looking for backdoor out-of-bound access to corporate networks (no egress filtering would help and IDS sensors can be circumvented), and free bandwidth. However, a variety of new public domain attack tools have appeared, notably Hotspotter, aircrack, and wep_lab. These and many other tools can be found at our site (www.wi-foo.com), which probably has the largest categorized collection of wireless security–related open source tools and is updated on a regular basis. Hotspotter allows successful man-in-the-middle attacks against unpatched Windows boxes, exploiting a flaw in Windows Profiles. Even the WPA-protected networks are vulnerable.

Aircrack optimizes cracking WEP, achieving a much higher efficiency than AirSnort, used casually for this task, and implements WEP'ed packets reinjection to accelerate WEP cracking in a way that's similar to OpenBSD, Wnet's reinj tool. WEPPlus, a proprietary Proxim's solution to the FMS attack against WEP now replaced by TKIP in WPA-certified Proxim/Orinoco products, is also vulnerable to aircrack's novel statistical attack. Wep_labs is another optimized WEP cracking tool and its latest version, posted to Packetstorm two days before this article was written, has been successfully ported to MS Windows. This puts the last nail into the coffin of WEP. Those still relying on it as the main WLAN defense measure should immediately switch to TKIP or higher-layer defenses. WEP cracking is now as easy as it gets, and even a Netstumbler kiddie with XP Home Edition has a reasonable chance of getting your key.

However, WPA version 1 is also not without security problems. We have mentioned the lack of mutual authentication with EAP-MD5, the first EAP type to be employed by 802.1x that is still widely in use, since any 802.1x implementation would most likely support it. Setting up HostAP plus accepting any authentication credentials on a Linux host and forcing the clients to associate with such a rogue AP is dead easy. Cisco EAP-LEAP is also flawed or, better to say, MS-CHAP uses it. The attack against EAP-LEAP (implemented by Asleap-imp) was first unleashed by Joshua Wright at Defcon 11. Since then more tools that use it, such as THC-Leapcrack, were released. TKIP is vulnerable to offline dictionary attacks, at least in the SOHO preshared key (PSK) mode. A research paper describing these attacks in detail is available at http://wifinetnews.com/archives/002452.html. There is also a lot of hype regarding the use of the WPA version 1 hash message authentication code (HMAC) implementation as a vector for DoS attacks. However, launching such attacks in practical terms has been far from easy and we have never encountered them in the real world. Please refer to Table 1 for a comparison of various wireless encryption schemes.

Secure Wireless Networks Design and Deployment Using Linux

Despite everything said above, WPAv1 (TKIP+802.1x+MIC hash or TKIP+PSK+MIC hash for SOHO mode) is far more secure than WEP, and WPAv2 (CCMP+802.1x+AES-based hash) is supposed to be even harder to crack than WPAv1. Here we'll describe how to implement these countermeasures to build a secure Linux wireless network that includes both Linux client hosts and Linux-based, custom-built access points. Many commercial access points, for example, those produced by Belkin and Netgear, are built on Linux anyway. We will extensively

ABOUT KONSTANTIN V. GAVRILENKO

Konstantin V. Gavrilenko has over 12 years' experience in IT and security and together with the other two authors is a cofounder of Arhont Ltd. His writing draws primarily from his real-world knowledge and experience in security consultancy and infrastructure development for a wide range of clients. His main areas of interest lie in security in general and, more specifically, in firewalling, cryptography, VPNs and IDS. Konstantin holds a first class BSc honors degree in management science from DeMontfort University and an MSc in Management from Lancaster University.
k.gavrilenko@arhont.com

use HostAP, open source software that can be downloaded from http://hostap.epitest.fi, for running and securing Linux clients and access points. Another common tool related to securing wireless networks is Xsupplicant (www.open1x.org), which provides Linux client-side support for the 802.1x port-based authentication standard. Figure 1 shows the 802.1x authentication mechanism.

HostAP

Jouni Malinen's HostAP is split into four parts: hostap-driver, hostapd, hostap-utils, and wpa-supplicant. The driver part is responsible for providing a flexible interface to the hardware and firmware functions of your wireless card. HostAP has initially been developed to support Intersil Prism chipset cards, but has now been extended to support other wireless chipsets such as Orinoco (alas, not in an Access Point mode). Hostapd daemon enables us to use a Prism chipset wireless card in Access Point mode (Master mode) with support for IEEE 802.1x and dynamic WEP rekeying, RADIUS Accounting, RADIUS-based ACL for IEEE 802.11 authentication, minimal IAPP (IEEE 802.11f), WPAv1, and IEEE 802.11i/RSN/WPAv2. Hostap utilities provide extended capabilities to your wireless interface and include diagnostic and debugging utilities, firmware update tools, and various wireless scripting interfaces.

ABOUT ANDREI A. MIKHAILOVSKY

Andrei A. Mikhailovsky first became enticed by Unix flavors back in school. He cultivated and expanded his knowledge into networking aspects of IT while obtaining his bachelors degree from The University of Kent at Canterbury. On accomplishing his MBA, he cofounded an Information Security company Arhont. Andrei's technical interests include user authentication mechanisms, database and directory services, wireless networking security, and systems integration.
andrei@arhont.com

The wpa-supplicant allows clients to utilize WLANs that support WPA-PSK (SOHO) and WPA Enterprise authentication methods.

hostapd

Many people wish to use their Prism2 cards as a functional and secure access point. This task is very easy to accomplish with hostapd. Download and compile the latest version of hostapd and copy hostapd.conf and hostapd binary to your preferred location. Now you need to edit the hostpad.conf configuration file to specify the exact functionality of your Linux-based AP. The hostapd is very flexible and extensive; it allows you to control every aspect and security function of the AP. On multiple occasions we've found HostAP-based access points to be more stable and controllable than the industry-standard expensive APs. We'll briefly outline how to configure hostapd to support 802.1x, WPA-PSK, and WPA Enterprise level user/device authentication, and re-keying schemes.

hostapd and 802.1x Authentication

If your equipment is outdated or an implementation of WPA is not feasible for your organization for some bizarre reason, frequent 802.1x-based WEP key rotation is one of the few choices left to secure your WLAN. To support dynamic WEP rekeying using hostpad, you should have the following configuration options enabled in hostapd.conf:

```
ssid=Arhont-x
macaddr_acl=1
accept_mac_file=/etc/hostapd.accept
deny_mac_file=/etc/hostapd.deny
ieee8021x=1
wep_key_len_broadcast=13
wep_key_len_unicast=13
wep_rekey_period=900
own_ip_addr=192.168.111.22
nas_identifier=hostap.arhont.com
auth_server_addr=192.168.111.101
auth_server_port=1812
auth_server_shared_secret=Very-Secret_
KEY
acct_server_addr=192.168.111.101
acct_server_port=1813
acct_server_shared_secret=Very-Secret_
KEY
```

Adjust the following settings of your specific network setup: ssid, own_ip_addr, nas_identifier, auth_server_addr, auth_server_shared_ secret, acct_server_addr, and acct_server_shared_secret. The next step is to create /etc/hostapd.accept and /etc/hostapd.deny files, which will have a list of MAC addresses of wireless cards that are allowed to connect to your AP. Once the configuration files are ready, launch hostapd in the following manner:

```
hostapd /etc/hostapd.conf
```

where /etc/hostapd.conf is the location of the hostapd configuration file. Don't forget that you'll also need a working RADIUS server. The FreeRADIUS server is an excellent open source solution. You can download it from www.freeradius.org. Check out the freeradius mailing list and FAQ if you have any difficulties with the RADIUS implementation.

hostapd and WPA-PSK

WPA-PSK is a replacement for static WEP on SOHO environment networks. To achieve WPA authentication using the Pre-Shared Key authentication, enable the following options in the hostapd.conf file:

```
ssid=Arhont-X
macaddr_acl=1
accept_mac_file=/etc/hostapd.accept
deny_mac_file=/etc/hostapd.deny
auth_algs=1
own_ip_addr=192.168.111.22
wpa=1
wpa_passphrase=secret-password-blah
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP CCMP
```

As with the previous example, adjust the settings to represent your network requirements. Unlike 802.1x and WPA Enterprise authentication means, with WPA-PSK there is no need to specify RADIUS server details. Once the configuration files are ready to be deployed, run hostpad the same way you would with the 802.1x setup.

Congratulations, you now have a working hostapd with WPA-PSK support. However, don't forget to select a very strong PSK, taking into account its vulnerability to bruteforcing.

hostapd and WPA Enterprise

To enable the enterprise grade WLAN encryption, consider using WPA-EAP

authentication. The following settings in hostapd.conf are required to enable this mode:

```
ssid=Arhont-x
macaddr_acl=1
accept_mac_file=/etc/hostapd.accept
deny_mac_file=/etc/hostapd.deny
ieee8021x=1
own_ip_addr=192.168.111.22
nas_identifier=hostap.arhont.com
auth_server_addr=192.168.111.101
auth_server_port=1812
auth_server_shared_secret=Very-Secret_
KEY
acct_server_addr=192.168.111.101
acct_server_port=1813
acct_server_shared_secret=Very-Secret_
KEY
wpa=1
wpa_key_mgmt=WPA-EAP
wpa_pairwise=TKIP CCMP
wpa_group_rekey=300
wpa_gmk_rekey=6400
```

As with dynamic WEP using 802.1x, WPA-EAP requires the use of a RADIUS server to authenticate mobile users. Once the hostapd is restarted, to take effect of the modified hostapd.conf file you should have a perfectly working Linux AP with WPA-EAP authentication means.

wpa_supplicant

We've dealt with the server side of setting up a Linux AP with various authentication schemes; now it's time to discuss a secure setup for the client side. Once the wpa-supplicant is downloaded (http://hostap.epitest.fi) and compiled (refer to the README file on how to create a .config file and compile the tool), you should edit the wpa_supplicant.conf configuration file. The default version of this file has already been provided for your convenience with a description of all the necessary fields that you might need to enable in order to participate in the WPA-protected WLAN. For instance, to have client-side support for the WLAN that authenticates its clients against the RADIUS server with EAP-TLS support, the following should be enabled:

```
network={
        ssid="Arhont-w"
        proto=WPA
        key_mgmt=WPA-EAP
        pairwise=CCMP TKIP
        group=CCMP TKIP
        eap=TLS
        identity="client_name@domain"
        ca_cert="/etc/ssl/certs/cacert.
pem"
        client_cert="/etc/ssl/certs/cli-
ent-cert.pem"
        private_key="/etc/ssl/certs/cli-
ent-priv.pem"
        private_key_passwd="client-se-
cret-password"
        priority=1
}
```

In case you don't need the WPA enterprise–level authentication and you simply want to enable the WPA-PSK support, the following setup should be reflected in the wpa_supplicant.conf file:

```
network={
       ssid="Arhont-w"
       psk="very secret PSK passphrase"
       priority=5
}
```

Once the configuration file is ready, you can launch the wpa-supplicant utility to associate and authenticate to the desired wireless network. It can be done the following way:

```
wpa_supplicant -iwlan0 -c/etc/wpa_sup-
plicant.conf -Dhostap -B
```

This should run wpa_supplicant in daemon mode using the hostap driver on a wlan0 interface with a configuration file located in /etc/wpa_supplicant.conf. You should get the following output from the iwconfig and iwlist commands once authentication is successful.

```
wlan0     IEEE 802.11b  ESSID:"Arhont-w"
          Mode:Managed
Frequency:2.462GHz  Access Point: 00:XX:
XX:XX:XX:XX
          Bit Rate:11Mb/s   Tx-Power:50
dBm   Sensitivity=0/3
          Retry:off   RTS thr:off   Frag-
ment thr:off
          Encryption key:61CC-3D80-
78CF-33D4-294F-B24F-C7C6-C6B8
       Security mode:restricted
          Power Management:off
          Link Quality=26/94  Signal
level=-69 dBm  Noise level=-95 dBm
          Rx invalid nwid:0  Rx invalid
crypt:0  Rx invalid frag:0
          Tx excessive retries:0  In-
valid misc:0   Missed beacon:0

ath0      3 key sizes : 40, 104, 128bits
          4 keys available :
```

	Cisco LEAP with TKIP	EAP-TLS with TKI	EAP-PEAP with TKIP	IPsec-based VPN
Key length (in bits)	128	128	128	168/128, 192, 256
Encryption algorithm	RC4	RC4	RC4	DES/3DES/AES
Packet integrity	CRC-32/MIC	CRC-32/MIC	CRC-32/MIC	MD5-HMAC/ SHA-HMAC
Device authentication	No	Certificate	No	PSK/certificate
User authentication	Username/password	Certificate	Username/password	Username/ password or Certificate
Certificate requirements	None	RADIUS server/ WLAN client	RADIUS server	Optional
Additional hardware	No	Certificate server	Certificate server	IPsec Concentrator
Per-user keying	Yes	Yes	Yes	Yes
Protocol support	Any	Any	Any	IP unicast / any with L2TP
Open standard	No	Yes	IETF draft RFC	Yes

TABLE 1 COMPARISON OF DIFFERENT WI-FI PROTECTION MECHANISMS

```
        [1]: E619-D524-557B-21A3-7B48-
6E26-DB68-2272 (128 bits)
        [2]: 006E-D5E5-6EBC-F41B-A9EC-
8906-74E6-DA7D (128 bits)
        [3]: off
        [4]: off
         Current Transmit Key: [1]
         Security mode:restricted
```

Xsupplicant

In a way the configuration of xsupplicant is quite similar to wpa-supplicant. To make the setup work, you'll need an AP with 802.1x support, a RADIUS server, and a set of certificates. The clients should download and compile the xsupplicant tool and edit the xsupplicant.conf file that has various configuration options to be implemented by xsupplicant. Unfortunately, the scope of this article doesn't allow us to go into the details of configuring and debugging 802.1x authentication schematics. More information on this topic can be easily Googled. If you prefer a hard copy of systematic reading material, our book *Wi-Foo: The Secrets of Wireless Hacking* is a hands-on guide to wireless security and hacking.

Once the configuration of xsupplicant is ready and configured for your WLAN, issue the following command to authenticate and get the per-session–based dynamic WEP key.

```
xsupplicant -i ath0 -c /etc/xsupplicant.
conf
```

If all goes well, you should have a similar output to iwconfig command:

```
ath0  IEEE 802.11g  ESSID:"Arhont-x"
      Mode:Managed  Frequency:2.462GHz
Access Point: 00:XX:XX:XX:XX:XX
      Bit Rate:36Mb/s   Tx-Power:50 dBm
Sensitivity=0/3
      Retry:off   RTS thr:off   Fragment
thr:off
      Encryption key:A3D0-FF7F-AD85-
E6AB-1808-38A8-90   Security mode:
restricted
      Power Management:off
      Link Quality=28/94  Signal level=-
67 dBm  Noise level=-95 dBm
      Rx invalid nwid:0  Rx invalid
crypt:0  Rx invalid frag:0
      Tx excessive retries:0  Invalid
misc:0   Missed beacon:0
```

As you can see, the xsupplicant successfully authenticated to AP with 802.1x support and received a pair of keys that is used to encrypt the unicast and broadcast traffic. By issuing the iwlist wlan0 key command you can get the list of keys that has been assigned to you by the AP.

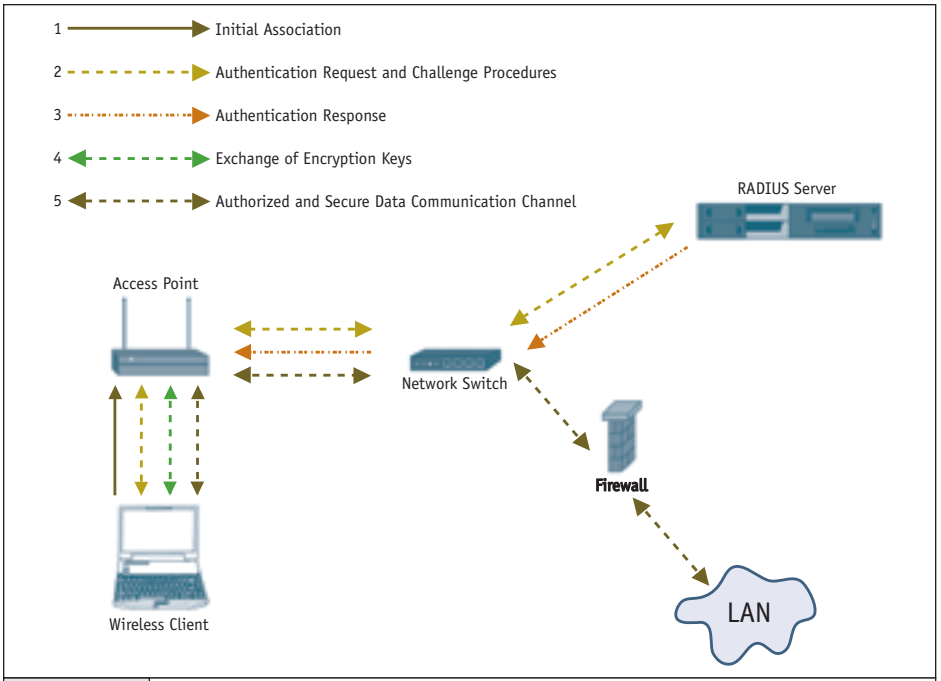

FIGURE 1 802.1X AUTHENTICATION MECHANISM

```
ath0      3 key sizes : 40, 104, 128bits
          4 keys available :
        [1]: A3D0-FF7F-AD85-E6AB-1808-
38A8-90 (104 bits)
        [2]: CCD1-7D97-A2D3-9B4A-CAA1-
DE7E-A6 (104 bits)
        [3]: off
        [4]: off
          Current Transmit Key: [1]
          Security mode:restricted
```

You can control reauthentication and rekeying time intervals from an access point side. If WEP and not TKIP or CCMP is used, we suggest rotating the key every 10–15 minutes.

Wireless Intrusion Detection (wIDS), Higher Layer Defenses, and Secure Wireless Gateways

Apart from implementing 802.11i-based countermeasures, there are more things you can do to secure your Linux-based wireless network. One is detecting attacks against your WLAN. This can be done by adding another wireless PCMCIA or PCI card to your Linux-based access point or building a specialized wIDS box, perhaps using a Soekris board (www.soekris.com) or a Linux PDA. This card will have to stay in the RFMON mode with a selected wIDS tool (or set of tools) running to analyze the traffic it picks. The defense part at www.wi-foo.com lists the currently available open source wIDS tools. Most of them are signature-based and easy and straightforward to configure. However, probably the best option for implementing now is to use Kismet to monitor your WLAN. Kismet detects an extensive list of suspicious wireless events, including Netstumbler kids and floods with various 802.11 management frames. It will also show you rogue access points and other WLANs in the area, as well as certain types of non-802.11 traffic using the same frequency range with Wi-Fi LANs.

When a suspicious event is detected, a siren sounds and information about the event flashes at the bottom of the screen. To see the info about recent suspicious events on your WLAN in a separate ncurses panel, press "w". If you're deploying a large WLAN, you can gain a great advantage from Kismet's client/server structure, with multiple clients installed along the network reporting the events to a centralized server.

On the server side, you can easily integrate Kismet with Snort, providing intrusion detection on all network layers. Open kismet.conf file, scroll toward the #fifo=/tmp/kismet_dump, uncomment this line, save the configuration file, and start kismet_server. Once started, Kismet will lock the /tmp/kismet_dump file until it's picked up by Snort. Now, let's start Snort. Configure it to your liking, but add an additional -r /tmp/kismet_dump switch when you run it, so it will read data from the FIFO feed of Kismet. You can further install and run ACID for pleasant and colorful IDS log viewing.

Another thing to consider is deploying higher-layer defenses instead of or with 802.11i (if the security requirements of the network are high or you're truly paranoid). Imagine a long-range point-to-point wireless link. Using IPSec as implemented by Linux OpenSwan or KAME suites to secure such link provides more flexibility than using WPA, since you have a great choice of (symmetric, assymmetric, and hash) ciphers and IPSec modes. You won't need the RADIUS server for the link a la WPA Enterprise and will achieve a higher level of security than provided by WPA SOHO.

Make sure that the IPSec key distribution over such a link is mutually authenticated (Diffie-Hellman) to avoid cracker_jack-style wireless man-in-the-middle attacks. If you consider IPSec too difficult to use or unnecessary, modern Linux PPTP with MPPE implementations are reasonably secure. Of course, in such cases you are limited to 128-bit RC4 and static PSK to encrypt wireless data.

If you want to connect a limited resources device such as a Linux PDA or mobile phone without 802.11i/WPA support, SSH port forwarding can be an appropriate and easy choice that is highly interoperable and does not put a large burden on the available device resources. Make sure that SSHv2 is running and there are no vulnerabilities in the sshd daemons used, since anyone can try to launch an attack against your link and daemons. There are many extensive sources that describe the practical use of IPSec, PPTP, and other VPN protocols such as cIPE and SSH port forwarding on Linux so we're not going to compete with them here.

Finally, it makes sense to separate your wireless and wired networks with a secure gateway. *802.11 Security* by Bruce Potter and Bob Fleck (O'Reilly) goes to great lengths explaining how to build such gateways using stateful filtering and port/protocol forwarding with Linux Netfilter. The gateway must be as hardened as it can get: we strongly suggest using security-oriented distros such as Astaro or Immunix and implementing kernel-level security (OpenWall, Grsecurity, St Jude, etc.) alongside the standard Linux-hardening practices. Due to the flexibility of the OS, such a gateway can also serve as an 802.11i-secured access point, wireless traffic load-balancer, wIDS/IDS sensor, VPN concentrator, and RADIUS server. Combine all these properties in a commercial, proprietary, closed-source solution and you'll get a $100,000 product. With Linux, the opportunities are there and are only limited by your imagination, desire, and time.