Hackers: Attacks and Defenses

Ed Skoudis
Senior Security Consultant, Intelguardians Network Intelligence, Howell, New Jersey, U.S.A.

Abstract
Computer attackers continue to hone their techniques, getting ever better at undermining our systems and networks. As the computer technologies we use advance, these attackers find new and nastier ways to achieve their goals—unauthorized system access, theft of sensitive data, and alteration of information. This entry explores some trends in computer attacks and presents tips for securing your systems. To create effective defenses, we need to understand the latest tools and techniques our adversaries are throwing at our networks. With that in mind, we will analyze four areas of computer attack that have received significant attention: wireless LAN attacks, active and passive operating system fingerprinting, worms, and sniffing backdoors.

WIRELESS LAN ATTACKS (WAR DRIVING)

A very large number of companies have deployed wireless LANs, using technology based on the IEEE 802.11b protocol, informally known as Wi-Fi. Wireless LANs offer tremendous benefits from a usability and productivity perspective: A user can access the network from a conference room, while sitting in an associate’s cubicle, or while wandering the halls. Unfortunately, wireless LANs are often one of the least secure methods of accessing an organization’s network. The technology is becoming very inexpensive, with a decent access point and wireless cards for a laptop or PC at very reasonable prices. In addition to affordability, setting up an access point is remarkably simple (i.e., if security is ignored). Most access points can be plugged into the corporate network and configured in a minute by a completely inexperienced user. Because of their low cost and ease of (insecure) use, wireless LANs are in rapid deployment in most networks today, whether upper management or even IT personnel realize or admit it. These wireless LANs are usually completely unsecure because the inexperienced employees setting them up have no idea of or interest in activating security features of their wireless LANs.

In our consulting services, we often meet with Chief Information Officers (CIOs) or Information Security Officers to discuss issues associated with information security. Given the widespread use of wireless LANs, we usually ask these upper-level managers what their organization is doing to secure its wireless infrastructure. We are often given the answer, “We don’t have to worry about it because we haven’t yet deployed a wireless infrastructure.” After hearing that stock answer, we conduct a simple wireless LAN assessment (with the CIO’s permission, of course). We walk down a hall with a wireless card, laptop, and wireless LAN detection software. Almost always we find renegade, completely unsecure wireless networks in use that were set up by employees outside of formal IT roles. The situation is similar to what we saw with Internet technology at the turn of the century. Back then, we would ask corporate officers what their organizations were doing to secure their Internet gateways. They would say that they did not have one, but we would quickly discover that the organization was laced with homegrown Internet.

Network Stumbling, War Driving, and War Walking

Attackers have taken to the streets in their search for convenient ways to gain access to organizations’ wireless networks. By getting within a few hundred yards of a wireless access point, an attacker can detect its presence and, if the access point has not been properly secured, possibly gain access to the target network. The process of searching for wireless access points is known in some circles as network stumbling. Alternatively, using an automobile to drive around town looking for wireless access points is known as war driving. As you might guess, the phrases war walking and even war bikiing have been coined to describe the search for wireless access points using other modes of transportation. I suppose it is only a matter of time before someone attempts war hang-gliding.

When network stumbling, attackers set up a rig consisting of a laptop PC, wireless card, and antenna for discovering wireless access points. Additionally, a global positioning system unit can help record the geographic location of discovered access points for later attack. Numerous software tools are available for this task as well. One of the most popular is NetStumbler (available...
at http://www.netstumbler.com), an easy-to-use GUI-based tool written by Marius Milner. NetStumbler runs on Windows systems, including Win95, 98, and 2000, and a PocketPC version called Mini-Stumbler has been released. For UNIX, several war-driving scripts have been released, with Wi-scan (available at http://www.dis.org) among the most popular.

This wireless LAN discovery process works because most access points respond, indicating their presence and their services set identifier (SSID) to a broadcast request from a wireless card. The SSID acts like a name for the wireless access point so that users can differentiate between different wireless LANs in close proximity. However, the SSID provides no real security. Some users think that a difficult-to-guess SSID will get them extra security. They are wrong. Even if the access point is configured not to respond to a broadcast request for an SSID, the SSIDs are sent in cleartext and can be intercepted.

In a war-driving trip in a taxi in Manhattan, an attacker discovered 455 access points in 1 hour. Some of these access points had their SSIDs set to the name of the company using the access point, gaining the attention of attackers focusing on juicy targets.

After discovering target networks, many attackers will attempt to get an IP address on the network, using the Dynamic Host Configuration Protocol (DHCP). Most wireless LANs freely give out addresses to anyone asking for them. After getting an address via DHCP, the attacker will attempt to access the LAN itself. Some LANs use the Wired Equivalent Privacy (WEP) protocol to provide cryptographic authentication and confidentiality. While WEP greatly improves the security of a wireless LAN, it has some significant vulnerabilities that could allow an attacker to determine an access point’s keys. An attacker can crack WEP keys by gathering a significant amount of traffic (usually over 500 MB) using a tool such as Airsnort (available at airsnort.shmoo.com/).

**Defending against Wireless LAN Attacks**

So, how do you defend against wireless LAN attacks in your environment? There are several levels of security that you could implement for your wireless LAN, ranging from totally unsecure to a strong level of protection. Techniques for securing your wireless LAN include the following:

- **Set the SSID to an obscure value.** As described earlier, SSIDs are not a security feature and should not be treated as such. Setting the SSID to an obscure value adds very little from a security perspective. However, some access points can be configured to prohibit responses to SSID broadcast requests. If your access point offers that capability, you should activate it.

- **Use media access control (MAC) address filtering.** Each wireless card has a unique hardware-level address called the MAC address. A wireless access point can be configured so that it will allow traffic only from specific MAC addresses. While this MAC filtering does improve security a bit, it is important to note that an attacker can spoof wireless card MAC addresses.

- **Use WEP, with periodic rekeying.** While WEP keys can be broken using Airsnort, the technology significantly improves the security of a wireless LAN. Some vendors even support periodic generation of new WEP keys after a given timeout. If an attacker does crack a WEP key, it is likely that they break the old key, while a newer key is in use on the network. If your access points support dynamic rotating of WEP keys, such as Cisco’s Aironet security solution, activate this feature.

- **Use a virtual private network (VPN).** Because SSID, MAC, and even WEP solutions have various vulnerabilities as highlighted earlier, the best method for securing wireless LANs is to use a VPN. VPNs provide end-to-end security without regard to the unsecured wireless network used for transporting the communication. The VPN client encrypts all data sent from the PC before it gets sent into the air. The wireless access point simply collects encrypted streams of bits and forwards them to a VPN gateway before they can get access to the internal network. In this way, the VPN ensures that all data is strongly encrypted and authenticated before entering the internal network.

Of course, before implementing these technical solutions, you should establish specific policies for the use of wireless LANs in your environment. The particular wireless LAN security policies followed by an organization depend heavily on the need for security in that organization. The following list, which I wrote with John Burgess of Predictive Systems, contains recommended security policies that could apply in many organizations. This list can be used as a starting point, and pared down or built up to meet specific needs.

- All wireless access points/base stations connected to the corporate network must be registered and approved by the organization’s computer security team. These access points/base stations are subject to periodic penetration tests and audits. Unregistered access points/base stations on the corporate network are strictly forbidden.

- All wireless network interface cards (i.e., PC cards) used in corporate laptop or desktop computers must be registered with the corporate security team.
- All wireless LAN access must use corporate-approved vendor products and security configurations.
- All computers with wireless LAN devices must utilize a corporate-approved VPN for communication across the wireless link. The VPN will authenticate users and encrypt all network traffic.
- Wireless access points/base stations must be deployed so that all wireless traffic is directed through a VPN device before entering the corporate network. The VPN device should be configured to drop all unauthenticated and unencrypted traffic.

While the policies listed earlier fit the majority of organizations, the policies listed in the following text may or may not fit, depending on the technical level of employees and how detailed an organization’s security policy and guidelines are:

- The wireless SSID provides no security and should not be used as a password. Furthermore, wireless card MAC addresses can be easily gathered and spoofed by an attacker. Therefore, security schemes should not be based solely on filtering wireless MAC addresses because they do not provide adequate protection for most uses.
- WEP keys can be broken. WEP may be used to identify users, but only together with a VPN solution.
- The transmit power for access points/base stations near a building’s perimeter (such as near exterior walls or top floors) should be turned down. Alternatively, wireless systems in these areas could use directional antennas to control signal bleed out of the building.

With these types of policies in place and a suitable VPN solution securing all traffic, the security of an organization’s wireless infrastructure can be vastly increased.

**ACTIVE AND PASSIVE OPERATING SYSTEM FINGERPRINTING**

Once access is gained to a network (through network stumbling, a renegade unsecured modem, or a weakness in an application or firewall), attackers usually attempt to learn about the target environment so they can hone their attacks. In particular, attackers often focus on discovering the operating system (OS) type of their targets. Armed with the OS type, attackers can search for specific vulnerabilities of those operating systems to maximize the effectiveness of their attacks.

To determine OS types across a network, attackers use two techniques: 1) the familiar, time-tested approach called active OS fingerprinting and 2) a technique with new-found popularity, passive OS fingerprinting. We will explore each technique in more detail.

**Active OS Fingerprinting**

The Internet Engineering Task Force (IETF) defines how TCP/IP and related protocols should work. In an ever-growing list of requests for comment (RFCs), this group specifies how systems should respond when specific types of packets are sent to them. For example, if someone sends a TCP SYN packet to a listening port, the IETF says that a SYN ACK packet should be sent in response. While the IETF has done an amazing job of defining how the protocols we use every day should work, it has not thoroughly defined every case of how the protocols should fail. In other words, the RFCs defining TCP/IP do not handle all of the meaningless or perverse cases of packets that can be sent in TCP/IP. For example, what should a system do if it receives a TCP packet with the code bits SYN-FIN-URG-PUSH all set? I presume such a packet means to SYNchronize a new connection, FINish the connection, do this URGently, and PUSH it quickly through the TCP stack. That is nonsense, and a standard response to such a packet has not been devised.

Because there is no standard response to this and other malformed packets, different vendors have built their OSs to respond differently to such bizarre cases. For example, a Cisco router will likely send a different response than a Windows NT server for some of these unexpected packets. By sending a variety of malformed packets to a target system and carefully analyzing the responses, an attacker can determine which OS it is running.

An active OS fingerprinting capability has been built into the Nmap port scanner (available at http://www.insecure.org/nmap). If the OS detection capability is activated, Nmap will send a barrage of unusual packets to the target to see how it responds. Based on this response, Nmap checks a user-customizable database of known signatures to determine the target OS type. Currently, this database houses over 500 known system types.

A more recent addition to the active OS fingerprinting realm is the Xprobe tool by Fyodor Yarochkin and Ofir Arkin. Rather than manipulating the TCP code bit options like Nmap, Xprobe focuses exclusively on the Internet Control Message Protocol (ICMP). ICMP is used to send information associated with an IP-based network, such as ping requests and responses, port unreachable messages, and instructions to quench the rate of packets sent. Xprobe sends between one and four specially crafted ICMP messages to the target system.

Based on a very carefully constructed logic tree on the
sensing side, Xprobe can determine the OS type. Xprobe is stealthier than the Nmap active OS fingerprinting capability because it sends far fewer packets.

**Passive OS Fingerprinting**

While active OS fingerprinting involves sending packets to a target and analyzing the response, passive OS fingerprinting does not send any traffic while determining a target’s OS type. Instead, passive OS fingerprinting tools include a sniffer to gather data from a network. Then, by analyzing the particular packet settings captured from the network and consulting a local database, the tool can determine what OS type sent that traffic. This technique is far stealthier than active OS fingerprinting because the attacker sends no data to the target machine. However, the attacker must be in a position to analyze traffic sent from the target system, such as on the same LAN or on a network where the target frequently sends packets.

One of the best passive OS fingerprinting tools is p0f (available at http://www.stearns.org/p0f/), originally written by Michal Zalewski and now maintained by William Stearns. P0f determines the OS type by analyzing several fields sent in TCP and IP traffic, including the rounded-up initial time-to-live, window size, maximum segment size, don’t fragment flag, window scaling option, and initial packet size. Because different OSs set these initial values to varying levels, p0f can differentiate between 149 different system types.

**Defending against Operating System Fingerprinting**

To minimize the impact an attacker can have using knowledge of your OS types, you should have a defined program for notification, testing, and implementation of system patches. If you keep your systems patched with the latest security fixes, an attacker will be far less likely to compromise your machines even if they know which OS you are running. One or more people in your organization should have assigned tasks of monitoring vendor bulletins and security lists to determine when new patches are released. Furthermore, once patches are identified, they should be thoroughly but quickly tested in a quality assurance environment. After the full functionality of the tested system is verified, the patches should be rolled into production.

While a solid patching process is a must for defending your systems, you may also want to analyze some of the work in progress to defeat active OS fingerprinting. Gaël Roualland and Jean-Marc Saffroy wrote the IP personality patch for Linux systems, available at ippersonality.sourceforge.net/. This tool allows a system administrator to configure a Linux system running kernel version 2.4 so that it will have any response of the administrator’s choosing for Nmap OS detection. Using this patch, you could make your Linux machine look like a Solaris system, a Macintosh, or even an old Windows machine during an Nmap scan. Although you may not want to put such a patch onto your production systems due to potential interference with critical processes, the technique is certainly worth investigating.

To foil passive OS fingerprinting, you may want to consider the use of a proxy-style firewall. Proxy firewalls do not route packets, so all information about the OS type transmitted in the packet headers is destroyed by the proxy. Proxy firewalls accept a connection from a client, and then start a new connection to the server on behalf of that client. All packets on the outside of the firewall will have the OS fingerprints of the firewall itself. Therefore, the OS type of all systems inside the firewall will be masked. Note that this technique does not work for most packet filter firewalls because packet filters route packets and, therefore, transmit the fingerprint information stored in the packet headers.

**RECENT WORM ADVANCES**

A computer worm is a self-replicating computer attack tool that propagates across a network, spreading from vulnerable system to vulnerable system. Because they use one set of victim machines to scan for and exploit new victims, worms spread on an exponential basis. In recent times, we have seen a veritable zoo of computer worms with names like Ramen, L10n, Cheese, Code Red, and Nimda. New worms are being released at a dizzying rate, with a new generation of worm hitting the Internet every 2–6 months. Worm developers are learning lessons from the successes of each generation of worms and expanding upon them in subsequent attacks. With this evolutionary loop, we are rapidly approaching an era of super-worms. Based on advances in worm functions and predictions for the future, we will analyze the characteristics of the coming super-worms that we will likely see in the next 6 months.

**Rapidly Spreading Worms**

Many of the worms have spread fairly quickly throughout the Internet. In July 2001, Code Red was estimated to have spread to 250,000 systems in about 6 hours. Fortunately, recent worms have had rather inefficient targeting mechanisms, a weakness that actually impeded their speeds. By randomly generating addresses and not taking into account the accurate distribution of systems in the Internet address space, these worms often wasted time looking for nonexistent systems or scanning machines that were already conquered.
After Code Red, several articles appeared on the Internet describing more efficient techniques for rapid worm distribution. These entries, by Nicholas C. Weaver and the team of Stuart Staniford, Gary Grim, and Roelof Jonkman, described the hypothetical Warhol and Flash worms, which theoretically could take over all vulnerable systems on the Internet in 15 minutes or even less. Warhol and Flash, which are only mathematical models and not actual worms (yet), are based on the idea of fast-forwarding through an exponential spread. Looking at a graph of infected victims over time for a conventional worm, a hockey-stick pattern appears. Things start out slowly as the initial victims succumb to the worm. Only after a critical mass of victims succumbs to the attack does the worm rapidly spread. Warhol and Flash jump past this initial slow spread by prescanning the Internet for vulnerable systems. Through automated scanning techniques from static machines, an attacker can find 100,000 or more vulnerable systems before ever releasing the worm. The attacker then loads these known vulnerable addresses into the worm. As the worm spreads, the addresses of these prescanned vulnerable systems would be split up among the segments of the worm propagating across the network. By using this initial set of vulnerable systems, an attacker could easily infect 99% of vulnerable systems on the Internet in less than an hour. Such a worm could conquer the Internet before most people have even heard of the problem.

**MultiPlatform Worms**

The vast majority of worms we have seen to date focused on a single platform, often Windows or Linux. For example, Nimda simply ripped apart as many Microsoft products as it could, exploiting Internet Explorer, the IIS Web server, Outlook, and Windows file sharing. While it certainly was challenging, Nimda’s Windows-centric approach actually limited its spread. The security community implemented defenses by focusing on repairing Windows systems.

While single-platform worms can cause trouble, be on the lookout for worms that are far less discriminating from a platform perspective. New worms will contain exploits for Windows, Solaris, Linux, BSD, HP-UX, AIX, and other operating systems, all built into a single worm. Such worms are even more difficult to eradicate because security personnel and system administrators will have to apply patches in a coordinated fashion to many types of machines. The defense job will be more complex and require more time, allowing the worm to cause more damage.

**Morphing and Disguised Worms**

Recent worms have been relatively easy to detect. Once spotted, the computer security community has been able to quickly determine their functionalities. Once a worm has been isolated in the lab, some brilliant folks have been able to rapidly reverse-engineer each worm’s operation to determine how best to defend against it.

In the very near future, we will face new worms that are far stealthier and more difficult to analyze. We will see polymorphic worms, which change their patterns every time they run and spread to a new system. Detection becomes more difficult because the worm essentially recodes itself each time it runs. Additionally, these new worms will encrypt or otherwise obscure much of their own payloads, hiding their functionalities until a later time. Reverse-engineering to determine the worm’s true functions and purpose will become more difficult because investigators will have to extract the crypto keys or overcome the obfuscation mechanisms before they can really figure out what the worm can do. This time lag for the analysis will allow the worm to conquer more systems before adequate defenses are devised.

**Zero-Day Exploit Worms**

The vast majority of worms encountered so far are based on old, off-the-shelf exploits to attack systems. Because they have used old attacks, a patch has been readily available for administrators to fix their machines quickly after infection or to prevent infection in the first place. Using our familiar example, Code Red exploited systems using a flaw in Microsoft’s IIS web server that had been known for over a month and for which a patch had already been published.

In the near future, we are likely going to see a worm that uses brand-new exploits for which no patch exists. Because they are brand new, such attacks are sometimes referred to as *zero-day exploits*. New vulnerabilities are discovered practically every day. Oftentimes, these problems are communicated to a vendor, who releases a patch. Unfortunately, these vulnerabilities are all too easy to discover, and it is only a matter of time before a worm writer discovers a major hole and first devises a worm that exploits it. Only after the worm has propagated across the Internet will the computer security community be capable of analyzing how it spreads so that a patch can be developed.

**More Damaging Attacks**

So far, worms have caused damage by consuming resources and creating nuisances. The worms we have seen to date have not really had a malicious payload. Once they take over hundreds of thousands of systems, they simply continue to spread without actually doing something nasty. Do not get me wrong; fighting Code Red and Nimda consumed much time and many
Worm Defenses

All of the pieces are available for a moderately skilled attacker to create a truly devastating worm. We may soon see rapidly spreading, multi platform, morphing worms using zero-day exploits to conduct very damaging attacks. So, what can you do to get ready? You need to establish both reactive and proactive defenses.

Incident response preparation

From a reactive perspective, your organization must establish a capability for determining when new vulnerabilities are discovered, as well as rapidly testing patches and moving them into production. As described earlier, your security team should subscribe to various security mailing lists, such as Bugtraq (available at http://www.securityfocus.com), to help alert you to such vulnerabilities and the release of patches. Furthermore, you must create an incident response team with the skills and resources necessary to discover and contain a worm attack.

Vigorously patch and harden your systems

From the proactive side, your organization must carefully harden your systems to prevent attacks. For each platform type, your organization should have documentation describing to system administrators how to build the machine to prevent attacks. Furthermore, you should periodically test your systems to ensure they are secure.

Block unnecessary outbound connections

Once a worm takes over a system, it attempts to spread by making outgoing connections to scan for other potential victims. You should help stop worms in their tracks by severely limiting all outgoing connections on your publicly available systems (such as your Web, Domain Name System (DNS), e-mail, and FTP servers). You should use a border router or external firewall to block all outgoing connections from such servers, unless there is a specific business need for outgoing connections. If you do need some outgoing connections, allow them only to those IP addresses that are absolutely critical. For example, your web server needs to send responses to users requesting web pages, of course. But does your web server ever need to initiate connections to the Internet? Likely, the answer is no. So, do yourself and the rest of the Internet a favor by blocking such outgoing connections from your Internet servers.

Nonexecutable system stack can help stop some worms

In addition to overall system hardening, one particular step can help stop many worms. A large number of worms utilize buffer overflow exploits to compromise their victims. By sending more data than the program developer allocated space for a buffer overflow attack allows an attacker to get code entered as user input to run on the target system. Most operating systems can be inoculated against simple stack-based buffer overflow exploits by being configured with nonexecutable system stacks. Keep in mind that nonexecutable stacks can break some programs (so test these fixes before implementing them), and they do not provide a bulletproof shield against all buffer overflow attacks. Still, preventing the execution of code from the stack will stop a huge number of both known and as-yet-undiscovered vulnerabilities in their tracks. Up to 90% of buffer overflows can be prevented using this technique. To create a nonexecutable stack on a Linux system, you can use the free kernel patch at http://www.openwall.com/linux. On a Solaris machine, you can configure the system to stop the execution of code from the stack by adding the following lines to the/etc/system file:

```
set noexec_user_stack = 1
set noexec_user_stack_log = 1
```


SNIFFING BACKDOORS

Once attackers compromise a system, they usually install a backdoor tool to allow them to access the machine repeatedly. A backdoor is a program that lets attackers access the machine on their own terms. Normal users are required to type in a password or use a cryptographic token; attackers use a backdoor to bypass these normal security controls. Traditionally, backdoors have listened on a TCP or UDP port, silently waiting in the background for a connection from the attacker. The
attacker uses a client tool to connect to these backdoor servers on the proper TCP or UDP port to issue commands.

These traditional backdoors can be discovered by looking at the listening ports on a system. From the command prompt of a UNIX or Windows NT/2000/XP machine, a user can type “netstat-na” to see which TCP and UDP ports on the local machine have programs listening on them. Of course, normal usage of a machine will cause some TCP and UDP ports to be listening, such as TCP port 80 for Web servers, TCP port 25 for mail servers, and UDP port 53 for DNS servers. Beyond these expected ports based on specific server types, a suspicious port turned up by the netstat command could indicate a backdoor listener. Alternatively, a system or security administrator could remotely scan the ports of the system, using a port-scanning tool such as Nmap (available at http://www.insecure.org/nmap). If Nmap’s output indicates an unexpected listening port, an attacker may have installed a backdoor.

Because attackers know that we are looking for their illicit backdoors listening on ports, a major trend in the attacker community is to avoid listening ports altogether for backdoors. You may ask, “How can they communicate with their backdoors if they aren’t listening on a port?” To accomplish this, attackers are integrating sniffing technology into their backdoors to create sniffing backdoors. Rather than configuring a process to listen on a port, a sniffing backdoor uses a sniffer to grab traffic from the network. The sniffer then analyzes the traffic to determine which packets are supposed to go to the backdoor. Instead of listening on a port, the sniffer employs pattern matching on the network traffic to determine what to scoop up and pass to the backdoor. The backdoor then executes the commands and sends responses to the attacker. An excellent example of a sniffing backdoor is the Cd00r program written by FX. Cd00r is available at http://www.phenoelit.de/stuff/cd00r.c.

There are two general ways of running a sniffing backdoor, based on the mode used by the sniffer program to gather traffic: the so-called nonpromiscuous and promiscuous modes. A sniffer that puts an Ethernet interface in promiscuous mode gathers all data from the LAN without regard to the actual destination address of the traffic. If the traffic passes by the interface, the Ethernet card in promiscuous mode will suck in the traffic and pass it to the backdoor. Alternatively, a non-promiscuous sniffer gathers traffic destined only for the machine on which the sniffer runs. Because these differences in sniffer types have significant implications on how attackers can use sniffing backdoors, we will explore non-promiscuous and promiscuous backdoors separately in the following text.

**Nonpromiscuous Sniffing Backdoors**

As their name implies, nonpromiscuous sniffing backdoors do not put the Ethernet interface into promiscuous mode. The sniffer sees only traffic going to and from the single machine where the sniffing backdoor is installed. When attackers use a nonpromiscuous sniffing backdoor, they do not have to worry about a system administrator detecting the interface in promiscuous mode.

In operation, the nonpromiscuous backdoor scours the traffic going to the victim machine looking for specific ports or other fields (such as a cryptographically derived value) included in the traffic. When the special traffic is detected, the backdoor wakes up and interacts with the attacker.

**Promiscuous Sniffing Backdoors**

By putting the Ethernet interface into promiscuous mode to gather all traffic from the LAN, promiscuous sniffing backdoors can make an investigation even more difficult. To understand why, consider the scenario shown in Fig. 1. This network uses a tri-homed firewall to separate the DMZ and internal network from the Internet. Suppose an attacker takes over the DNS server on the DMZ and installs a promiscuous sniffing backdoor. Because this backdoor uses a sniffer in promiscuous mode, it can gather all traffic from the LAN. The attacker configures the sniffing backdoor to listen in on all traffic with a destination address of the Web server (not the DNS server) to retrieve commands from the attacker to execute. In our scenario, the attacker does not install a backdoor or any other software on the web server. Only the DNS server is compromised.

Now the attacker formulates packets with commands for the backdoor. These packets are all sent with a destination address of the web server (not the DNS server). The web server does not know what to do with these commands, so it will either discard them or send a RESET or related message to the attacker. However, the DNS server with the sniffing backdoor will see the
commands on the LAN. The sniffer will gather these commands and forward them to the backdoor where they will be executed. To further obfuscate the situation, the attacker can send all responses from the backdoor using the spoofed source address of the web server.

Given this scenario, consider the dilemma faced by the investigator. The system administrator or an intrusion detection system complains that there is suspicious traffic going to and from the web server. The investigator conducts a detailed and thorough analysis of the web server. After a painstaking process to verify the integrity of the applications, operating system programs, and kernel on the web server machine, the investigator determines that this system is intact. Yet backdoor commands continue to be sent to this machine. The investigator would only discover what is really going on by analyzing other systems connected to the LAN, such as the DNS server. The investigative process is significantly slowed down by the promiscuous sniffing backdoor.

**Defending against Sniffing Backdoor Attacks**

It is important to note that the use of a switch on the DMZ network between the web server and DNS server does not eliminate this dilemma. Attackers can use active sniffers to conduct Address Resolution Protocol (ARP) cache poisoning attacks and successfully sniff a switched environment. An active sniffer such as Dsniff (available at http://www.monkey.org) married to a sniffing backdoor can implement this type of attack in a switched environment.

So if a switch does not eliminate this problem, how can you defend against this kind of attack? First, as with most backdoors, system and security administrators must know what is supposed to be running on their systems, especially processes running with root or system-level privileges. Keeping up with this information is not a trivial task, but it is especially important for all publicly available servers such as systems on a DMZ. If a security or system administrator notices a new process running with escalated privileges, the process should be investigated immediately. Tools such as lsif for UNIX (available at ftp://vic.cc.purdue.edu/pub/tools/unix/lsif/) or Inzider for Windows NT/2000 (available at http://ntsecurity.nu/toolbox/inzider/) can help to indicate the files and ports used by any process. Keep in mind that most attackers will not name their backdoors “cd00r” or “backdoor,” but instead will use less obvious names to camouflage their activities. In my experience, attackers like to name their backdoors “SCSI” or “UPS” to prevent a curious system administrator from questioning or shutting off the attackers’ processes.

Also, while switches do not eliminate attacks with sniffers, a switched environment can help to limit an attacker’s options, especially if it is carefully configured. For your DMZs and other critical networks, you should use a switch and hard-code all ARP entries in each host on the LAN. Each system on your LAN has an ARP cache holding information about the IP and MAC addresses of other machines on the LAN. By hard-coding all ARP entries on your sensitive LANs so that they are static, you minimize the possibility of ARP cached poisoning. Additionally, implement port-level security on your switch so that only specific Ethernet MAC addresses can communicate with the switch.

**CONCLUSIONS**

The computer underground and information security research fields remain highly active in refining existing methods and defining completely new ways to attack and compromise computer systems. Advances in our networking infrastructures, especially wireless LANs, are not only giving attackers new avenues into our systems, but they are also often riddled with security vulnerabilities. With this dynamic environment, defending against attacks is certainly a challenge. However, these constantly evolving attacks can be frustrating and exciting at the same time, while certainly providing job security to solid information security practitioners. While we need to work diligently in securing our systems, our reward is a significant intellectual challenge and decent employment in a challenging economy.