Global Information Assurance Certification Paper

Exploiting Internet Explorer via IFRAME

GIAC Certified Incident Handler (GCIH)
Practical Assignment v4

By: Jim Becher

Date Submitted: 23 December 2004

Abstract

This paper is submitted to meet the Practical Assignment (version 4) requirement for achieving the GIAC Certified Incident Handler (GCIH) certification.

In this paper an exploit that takes advantage of a client-side vulnerability in Internet Explorer 6 will be examined in a lab environment. Additional insights based on real-world exposure to use of the exploit are also provided. The intended audience is those responsible for the operation, maintenance, and security of Microsoft Windows workstations that are in the hands of users.

Originally the practical was to cover a vulnerability and exploit for which a patch had not been released. During the planning and preparation of the practical, several virus/worm variants were released based on the vulnerability as well as the patch to address the issue.

Statement of Purpose

When most people think of exploit code, they think of the "compile-and-aim-at-a-vulnerable-service" variety. This typically involves exploiting an operating system service or application server that is listening on a specific port(s). This practical will examine a client-side vulnerability – a security deficiency in the application that a user would use to interact with an application server. In this case, the user application is Internet Explorer 6, and the vulnerability is a buffer overflow in the handling of In-line Floating Frame (IFRAME) attributes.

The intent of the chosen attack is to get an attacker's arbitrary code executed on the target system. The arbitrary code that will be executed will open a backdoor command shell on a high numbered port, allowing the attacker remote access to the target machine. The command shell will operate with the privileges of the currently logged-in user.

The scenario that will be covered requires minimal active involvement from the target. By using a very common web browser and visiting a malicious website, the target system will be exploited. There are scenarios that are even more passive in nature that will be mentioned, but they will not be examined in this paper.

The objectives are to 1) spawn a backdoor shell on the target machine by exploiting a client-side vulnerability in Internet Explorer 6; and 2) demonstrate one likely scenario how this could be carried out by an attacker.

To carry out this attack, the attacker will use e-mail and a hostile web server to compromise the Windows XP target machine. E-mail is used only as the delivery

mechanism to get the URL in front of as many users as possible. It seems like a logical approach given the client-side vulnerability discussed in this paper. The hostile web server would most likely be a machine on the Internet that the attacker has previously compromised. In this case, the hostile web server and the Windows XP target machine are both on the test network. A free web-based e-mail service was used.

The Exploit

Name

The name of the exploit that is going to be used is "InternetExploiter v0.1". The exploit is a malicious piece of HTML with Javascript. The nature of standard HTML and Javascript is that the source code is readily viewable – it is not compiled. The Javascript includes shellcode, which is compiled, and in this case, also encoded. The exploit was authored by Berend-Jan Wever (a.k.a. Skylined) and posted to his website, dated 2 November 2004.

The history of this vulnerability and exploit goes back at least to an announcement on the BugTraq mailing list by Michal Zalewski on 18 October 2004. In his post, he announced that he had released a new tool to generate "razor-sharp shards of HTML." The name of the tool was "mangleme". When Zalewski released the tool, he indicated that he had used the tool to test several popular web browsers. The initial results were that several of the browsers he had tested kept crashing on a regular basis, with the exception of Internet Explorer. In his initial post the list of web browsers that were affected were: Mozilla, Opera, and Lynx. In a follow-up post four days later, Michal Zalewski indicated that he had received reports of crashes on several other browsers, including Microsoft Internet Explorer.

Two days after Zalewski's follow-up post about Internet Explorer also crashing, "ned" (nd@felinemenace.org) posted links to HTML that would cause Internet Explorer to crash. Berend-Jan Wever provided an initial analysis of the vulnerability less than 24 hours later, where he indicated that a working, reliable exploit shouldn't take long to code. The exploit that will be examined, InternetExploiter, is authored by Berend-Jan Wever based on the malicious HTML that "ned" found would cause Internet Explorer to crash.

As far as community references, the Common Vulnerabilities and Exposures dictionary has given this vulnerability a CVE Name of CAN-2004-1050. They describe the vulnerability as a "Heap-based buffer overflow in Internet Explorer 6 allows remote attackers to execute arbitrary code via long (1) SRC or (2) NAME attributes in IFRAME, FRAME, and EMBED elements, as originally discovered using the mangleme utility." SecurityFocus has given this vulnerability a bug ID of 11515, which contains the description "Microsoft Internet Explorer Malformed

IFRAME Remote Buffer Overflow Vulnerability". CERT also released Vulnerability Note VU#842160 regarding this vulnerability.

In response to the vulnerability, Microsoft Security Bulletin MS04-040 was published on 1 December 2004. Microsoft released the patch out of their normal security patch cycle. Except in special cases, Microsoft releases security bulletins/patches on the second Tuesday of each month. However, because of the critical nature of the issue, the fact that an exploit was publicly available, and that the vulnerability was being actively exploited it was probably no surprise that this patch was released out of cycle.

Operating System

In their bulletin, Microsoft indicated that Internet Explorer 6 Service Pack 1 is vulnerable on the following operating systems:

- Microsoft Windows NT Server 4.0 Service Pack 6a
- Microsoft Windows NT 4.0 Terminal Service Edition Service Pack 6
- Microsoft Windows 98
- Microsoft Windows 98 SE
- Microsoft Windows Me
- Microsoft Windows XP Service Pack 1
- Microsoft Windows 2000 Service Pack 3 and Service Pack 4

In addition, Internet Explorer 6 is vulnerable on the following operating system:

- Microsoft Windows XP Service Pack 1 (64-bit)

However, according to Berend-Jan Wever, the vulnerability existed on all Service Packs for Microsoft Windows 2000 and Microsoft Windows XP, with the exception of Microsoft Windows XP Service Pack 2. The author believes that the reason for the discrepancy is due to a Microsoft policy of only listing currently supported operating system and service pack combinations in their bulletins.

Protocols/Services/Applications

The vulnerability does not exist in a particular protocol or service, but in the way the Internet Explorer application handles malformed HTML elements. The exploit code that will be covered preys on one particular HTML element, the IFRAME tag element.

IFRAME is the HTML tag name for Inline Floating Frames. Inline Floating Frames are a somewhat obscure feature that has been supported in Internet Explorer since version 3. Inline Floating Frames allow a webmaster to place a subwindow in an HTML document. The subwindow contains a completely separate document and has scroll bars independent of the HTML document in which they are placed. A normal frame will divide the browser window into subwindows, but an Inline Floating Frame is placed at a specific point inside an HTML document. If a browser supports IFRAME, it displays the document

referenced in the IFRAME tag, retrieving the document from the location defined by the SRC attribute. If the browser does not support IFRAME, it treats the tags as if they were not there, and displays the text between the <IFRAME …> and </IFRAME> tags. Typically, a link to the document referenced in the IFRAME SRC attribute would be included as suitable text between the <IFRAME> and </IFRAME> tags – allowing the user of the browser that does not support IFRAME to click on the link to the referenced document.

Description

The vulnerability is a heap-based buffer overflow in Internet Explorer in the way it handles long attributes for certain HTML elements, such as IFRAME, FRAME, and EMBED. Internet Explorer is vulnerable because it is not performing appropriate sanity checks on the HTML that is provided from the server. IFRAME attributes are included in InternetExploiter that are too long to be stored appropriately in the buffers that are supposed to hold them. It is exploitable, because the attacker is able to control execution after the buffer overflow occurs.

Before addressing how the exploit is taking advantage of the vulnerability, a short explanation of the heap and heap blocks is needed. The heap is an area of memory that is earmarked for the application to use to store data while it is running. The amount of heap that will be needed is not known, so there are methods that an application can use to request and free portions of the heap. These portions are known as heap blocks. Heap blocks that are requested cause the heap to grow upward, toward higher addresses. The heap blocks are 262,144 dwords in size. For each heap block, there is a system-assigned header that is 20 dwords long.

The first step the exploit performs to take advantage of the vulnerability is to prepare space in the heap for execution. The Javascript constructs the data that will be contained in the heap blocks. Each heap block is composed of a no-op sled followed by the shellcode. The length of the no-op sled is determined by taking into account the system-assigned heap block header, the heap block size, and the length of the shellcode. In the case of InternetExploiter, each heap block contains the 20 dwords of heap block header, followed by 261,959 dwords of no-op sled, followed by the 165 dwords of shellcode. The Javascript was not hard-coded with the length of the no-op sled that was required. It alters the length of the no-op sled based on the length of the shellcode. It is assumed that the intent was so that alternative shellcode could be placed in the exploit code with minimal changes required. The Javascript then generates an array of 700 of these heap blocks.

The other step the exploit performs to take advantage of the vulnerability is to supply Internet Explorer with IFRAME attributes that exceed the buffers that are supposed to hold them. The result is that the program execution can be controlled, and is pointed into the heap. The heap contains the 700 heap blocks

of no-op sleds and shellcode. Execution lands inside one of the no-op sleds, and the code slides down to the shellcode. The shellcode is then executed, creating a command shell backdoor on TCP port 28876.

The shellcode that was used in InternetExploiter is also available from Berend-Jan Wever's website. In a personal e-mail exchange, he indicated that the shellcode used in InternetExploiter was the w32_bind_0free_loop shellcode, but that other shellcode could be used.

Signatures Of The Attack

There are signatures of this attack that can be used to detect this particular exploit in an organization's environment – both from a network and host perspective. The caveat to these signatures is that the application is so flexible, so widely used, and tied to other applications, that detection becomes a challenge. Even in the simple case of the InternetExploiter v0.1 exploit code, an attacker could easily alter the transmission channel for the HTML and bypass network detection. Detection at the host is probably going to be more guaranteed as that is where the code is going to execute, regardless of the transmission channel.

At a high level, there are two main indicators that the attack is being used in an organization's environment – 1) IFRAME elements in HTML that have very long SRC and FILE attributes; and 2) a command shell is provided to anyone connecting to a machine on TCP port 28876. The next two sections will discuss how these two indicators can be detected at a technical level from a network and host perspective, as well as some complicating factors to detection.

Network Detection

The day after the exploit code was released, BleedingSnort.com had their first revision of a snort rule to detect the vulnerability. The most recent snort signature from BleedingSnort.com is the rule's third revision, which was published later that same day. The lab experiments were unsuccessful in getting the rule to fire on the traffic generated on the lab network. In an e-mail correspondence with the author of the rule, he indicated that the rule only detects the exploit in its really pure form, so he wasn't sure the rule was all that useful.

As far as a mainstream snort signature, there does not appear to be one available as of 3 December 2004. The online snort signature database, as well as downloading the snort 2.0 signatures, snort 2.1 signatures, snort 2.2 signatures, and CURRENT signatures – revealed no reference to "IFRAME" or "2004-1050".

Internet Security Systems (ISS) released an X-Press Update (XPU) on 9 November 2004 that contained the "HTTP_IE_IFrame_BO" signature. The signature name references HTTP, but there is no indication what network traffic

this signature is applied to. ISS has a closed signature set, so viewing the traffic characteristic matching criteria, as well as the payload matching criteria, is not possible. Firsthand experience with this signature matching traffic indicated a source port of 80/TCP – where it did detect a malicious webpage containing the InternetExploiter v0.1 exploit code as early as 15 November 2004. In the process of investigating, the malicious webpage was obtained using wget – again triggering an alert from ISS.

If an organization is using an IDS that allows them to write and use their own signatures, looking for command shell activity on TCP port 28876 might aid in detecting which machines are actively being compromised. Or if an organization is performing binary capture of network traffic at its borders, they should consider setting up a filter to report on activity on this port on a periodic basis. The IDS alerts or periodic reports would only provide indication of this vulnerability being actively exploited, and not provide visibility in the number of attempts or machines that have been compromised but are not yet being actively exploited.

Detection from the network vantage point is complicated by at least a couple of factors: 1) a webserver can redirect a browser to retrieve a webpage from any TCP port; 2) the exploit code can be delivered via alternative protocols; and 3) some protocols do not lend themselves easily to being monitored.

In the third revision of the BleedingSnort rule, the rule tells snort to examine only data streams that match a source port of $HTTP_PORTS. The $HTTP_PORTS is a variable in the snort.conf file that associates a list of ports to that variable. With snort version 2.2.0, the variable is set by default to only port 80. So the Bleeding Snort rule is only looking on those defined source ports for the malicious HTML. As mentioned earlier, a webserver can redirect a browser to retrieve a webpage from any TCP port. Adding all 65,535 TCP ports to the $HTTPS_PORTS might negatively impact the sensor's performance and cause false positives. The same issue may be faced by ISS, depending on which ports they are monitoring for the malicious HTTP traffic of InternetExploiter. If an organization requires the use of a proxy for web browsing, this is much less of an issue. The IDS would just need to be tuned to the port(s) the proxy server is configured to proxy HTTP traffic on. In addition, the author of the BleedingSnort rule indicated that mostly they have seen it heavily obfuscated with java encoding.

Another complicating factor is that the exploit code can be delivered via alternative protocols. If an organization can positively identify all HTTP traffic traversing their network, it is possible that an attacker still might be able to avoid detection by using another protocol. The Simple Mail Transfer Protocol (SMTP) might be an avenue an attacker would be likely to use. An HTML-based e-mail message containing the malicious HTML or file attachments (Microsoft Word document, password protected zip file, etc.) might be alternative attack vectors. The HTML-based e-mail has some limitations, because Microsoft's latest e-mail

clients (Outlook Express 6, Outlook 2002, and Outlook 2003) open HTML e-mail messages in the Restricted Sites zone by default according to the MS04-040 bulletin.

The last complicating factor is that some protocols do not lend themselves easily to being monitored. A simple step an attacker might use to avoid network detection is to redirect the browser to an SSL-encrypted webpage (HTTPS). Once the connection is encrypted, passive network detection would be blind to the malicious HTML.

None of these methods to avoid network detection were studied.

Host Detection
As mentioned earlier, detection at the host is going to be more guaranteed, because the transmission methods can vary greatly and the host is where the code is actually going to run. As far as detecting execution of this particular exploit in an organization's environment, there are several methods that can be used.

While the Symantec anti-virus software that was running on the target machine did not keep the exploit code from running, it did detect the activity. The anti-virus software displayed a dialog window to the user informing them of the activity. It is hopeful that most organizations provide their users with awareness training on viruses and a central place to report them. If the anti-virus product in use supports centralized notification, the appropriate group within an organization can be automatically notified of the activity, without relying on the user.

Another method of detecting the exploit in action in an organization's environment is to check to see if there is a process listening on TCP port 28876. This can be done by performing a scan of the organization's address space looking for machines with a process accepting connections on that particular port.

If there are only a small number of machines that are suspected, performing a "netstat –ano" could be used to see if there was a process listening on this port. Beware that local system binaries might not be reliable on a machine that is suspected of being compromised. A known-good copy of netstat on external media is recommended. There are tools other than netstat that could be used for this purpose, such as openports from DiamondCS or fport from Foundstone.

A quick search of the Internet Assigned Numbers Authority (IANA) and www.portsdb.org do not indicate that TCP port 28876 is a standard port for an application service. A Google search indicated that there are a couple of other exploits that provide a command shell on this port. In any case, a process listening on this port would be worth investigating.

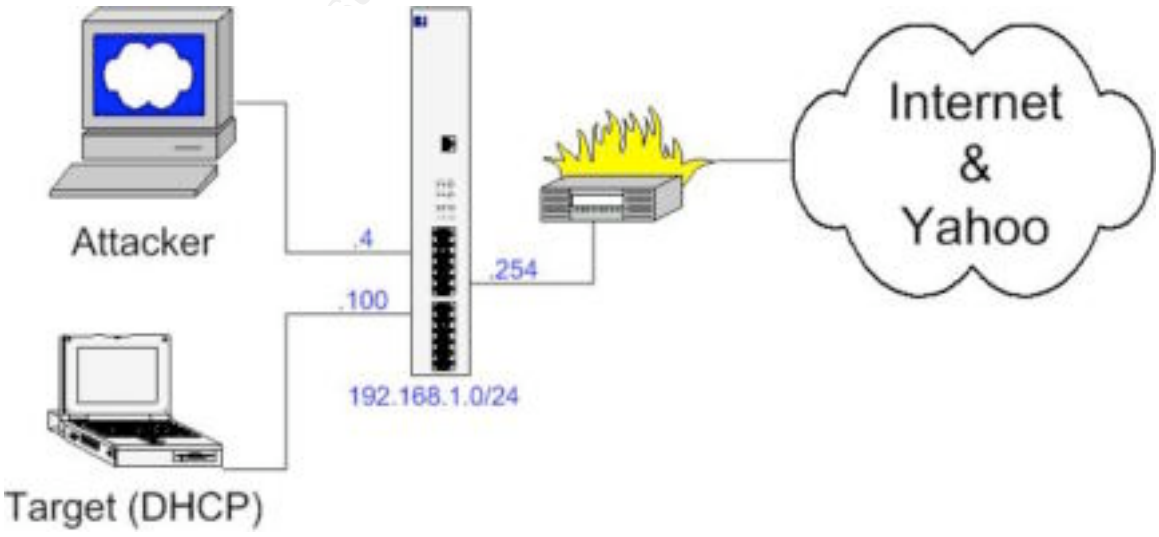
If in the investigation, an active connection is observed involving a foreign network and one of an organization's systems on TCP port 28876, the attacker may be utilizing their newly created backdoor on the system. However, there is a chance that the connection in question is one that was actually originated from the organization's machine to a remote machine, and that the use of TCP port 28876 is the client-side source port of an HTTP connection, ssh connection, etc. The port in use by the remote, foreign machine may give some indication, but sniffing the network traffic to/from your machine in question looking for traffic on TCP port 28876 should provide conclusive evidence one way or the other.

In lab experiments with InternetExploiter, RegShot was used to see if there were any registry changes that were specific to the exploit. A snapshot of the registry was taken before the malicious HTML was visited for the first time. The first time the webpage was requested containing the malicious HTML, the attacker's code executed and opened the command shell backdoor. After which, a second snapshot of the registry was taken and RegShot generated a report of the differences. None of the value/key additions or modifications seemed specific to the InternetExploiter v0.1 exploit code. Nor were there registry changes typically associated with malware, such as additions in the Run and RunOnce keys.

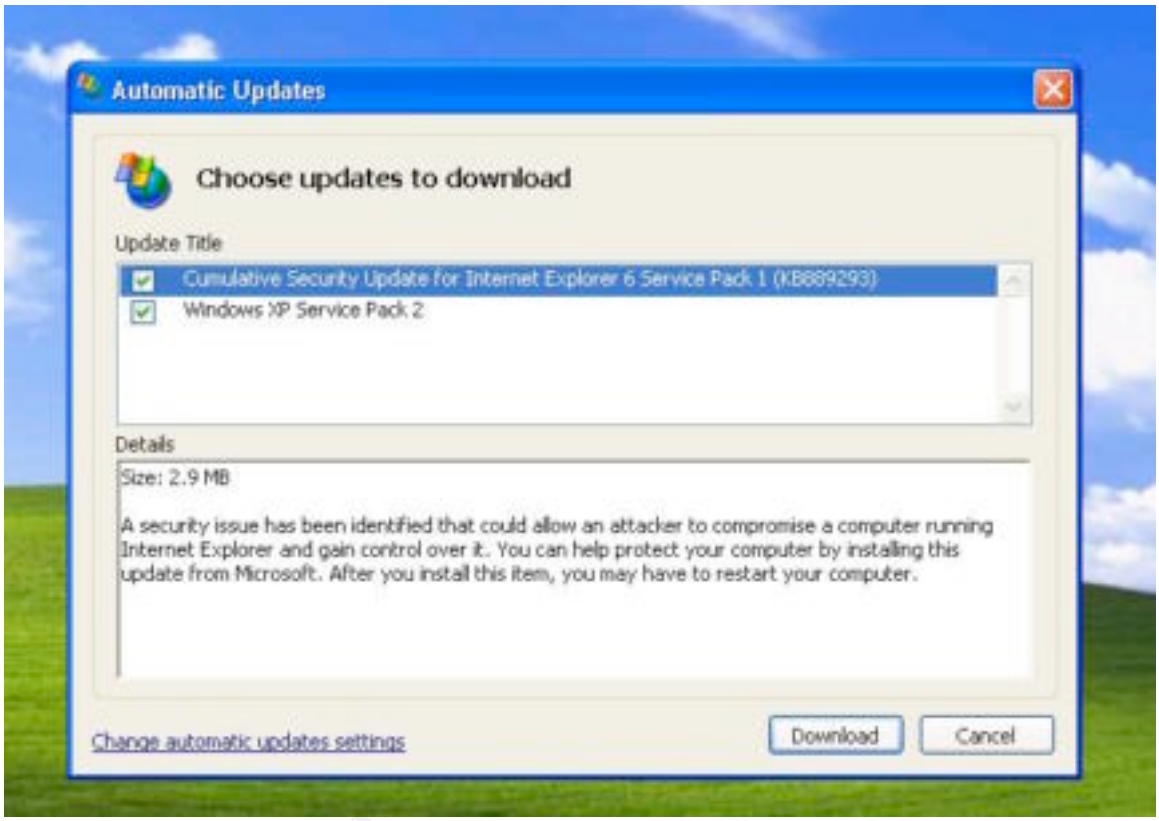
Stages of the Attack Process

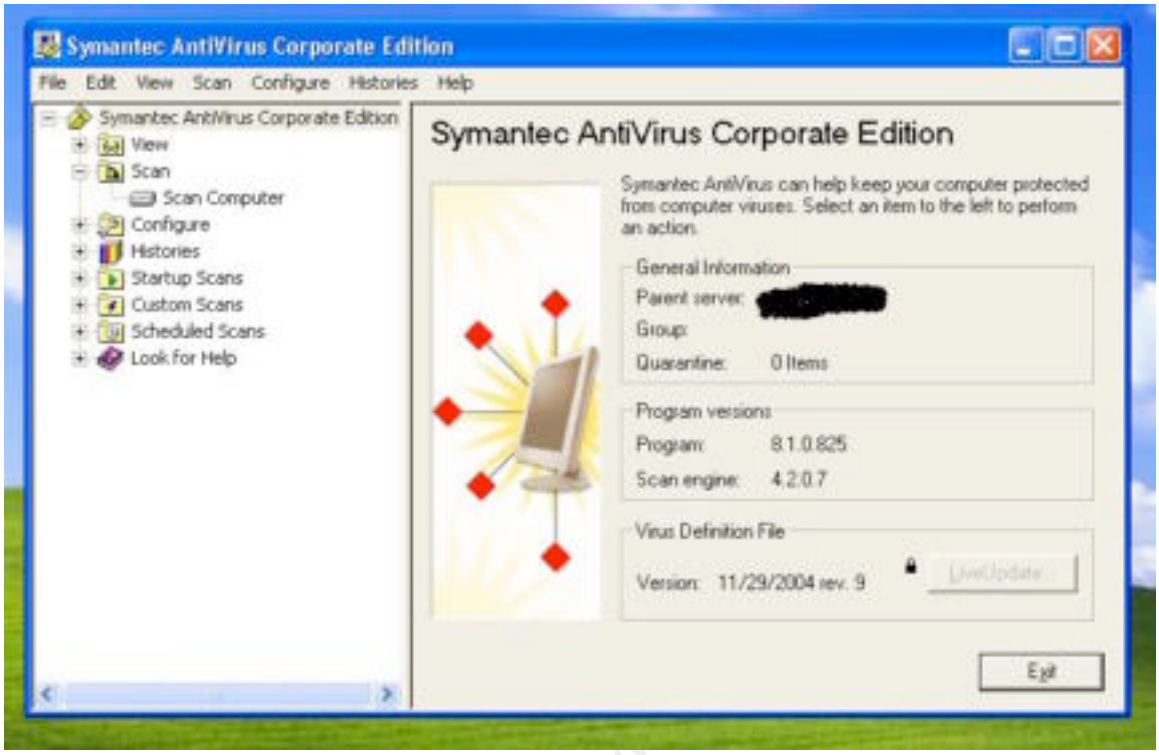
Network Diagram

Before beginning the stages of the Attack Process, a network diagram is provided along with a description of the lab environment that is being used to examine this exploit. The lab environment consists of two machines and use of a free web-based e-mail service. The two machines are physically attached to the same network out of convenience. While having them attached to the same network makes exploitation much easier, it is not a requirement.



The target machine, labeled “Target” in the network diagram is a Windows XP Service Pack 1 laptop. The target machine has all patches applied, except MS04-040. The patch was released while the lab environment was being built and configured. The target machine is also running Symantec Anti-Virus Corporate Edition (SAV-CE) version 8.1 and is using the 11/29/2004 rev9 virus definitions. For these experiments, a local user was added and configured and placed in the Administrators group.





The attacker machine, labeled “Attacker” in the network diagram is a Redhat 8.0 linux machine, running thttpd v2.25b. The use of linux or the use of this particular web server is not required. In real-world scenarios, the malicious machine will most likely be a machine that the attacker has compromised out on the Internet somewhere. They probably will not use their own machine for this purpose, and it will most likely not be located on the same physical network as the target.

In addition to the two machines, the free web-based e-mail service Yahoo was used. The account gcih2004@yahoo.com was created specifically for the purpose of writing this practical. Any provider of e-mail – corporate e-mail, other free web-based e-mail services, etc. – could have been used. E-mail was used only as a delivery mechanism to get the URL to the malicious webpage in front of the user.

Reconnaissance

Reconnaissance is the first of five stages in the attack process. In the reconnaissance stage, the attacker gathers information that might aid him in exploiting or compromising his target. The SANS Hacker Techniques, Exploits, and Incident Handling course covered many forms of reconnaissance. The reconnaissance tool(s) that an attacker would use depend on how they are selecting their target and what information is required to conduct the attack. Are they just picking random IP address space or do they have an organization they wish to attack? Assuming that the attacker has set his sights on compromising a

specific organization, the attacker doesn't have a domain name or IP address space, just the organization's name.

In order for the attacker to get a user at the target organization to visit his malicious webpage containing InternetExploiter, he decides on using e-mail as an attack vector. A typical e-mail address consists of *username@domainname.tld*. Currently, the attacker only has the name of the organization. So the attacker needs to learn the domain name and gather valid usernames before he can send the e-mail.

Enter the ultimate reconnaissance tool – Google. If you provide Google with the name of the organization, chances are that a couple of the top ten results are going to provide you with at least the domain name. Just to be sure, the attacker can visit the website or use WHOIS to verify that the domain name belongs to the target organization. Once the attacker has the domain name, he can search Google again, or search Google Groups to identify e-mail addresses of individuals in the target organization.

Some of the typical reconnaissance tools, such as a few of those covered in the SANS Track 4 course, are probably of limited direct benefit in the scenario being described. Knowing the IP address space assigned to the target organization, such as output from ARIN, is not going to help get the exploit code into a browser there. DNS interrogation using nslookup and dig will only provide IP address and hostname information. This approach will not directly help get the exploit code into a browser either. Sam Spade is another tool that was discussed in the course. Of all the features this tool provides, the only one that may be of value is the SMTP VRFY feature. It is suspected that most organizations have configured their mail servers not to respond to VRFY requests.

Other typical reconnaissance tools, such as a few of those covered in the SANS Track 4 course, might provide the attacker e-mail addresses. For instance, WHOIS lookup results may provide e-mail addresses for individuals in the organization. The administrative or technical contacts may be individuals in the target organization, and their work e-mail addresses may be listed. Also, performing searches on the target's website may provide the attacker individual e-mail addresses or group e-mail addresses. Group e-mail addresses would be especially beneficial from an attacker's standpoint – the effort of one e-mail message would result in the possibility of multiple individuals receiving the e-mail message, thus potentially compromising multiple machines. Some of the common group e-mail addresses include Investor Relations, Corporate Communications, Operations Center, Abuse, Sales, Human Resources, and Support. One company even states on their website (due to acquisition activity) that their e-mail address format is *firstname.lastname@domainname.net*. Armed with this information and the names of their senior management (also from their website), an attacker has just identified several high-profile e-mail addresses.

Scanning

The second of the five stages of the attack process is scanning. Scanning is an activity that an attacker would engage in to identify vulnerable operating systems or application servers on the target organization's network. In this scenario, however, the intent is to exploit an application client, Internet Explorer. Internet Explorer will not have a listening port or a network-accessible service. It requires a user to initiate the application and provide URLs.

Scanning an organization's IP address space is only going to be of limited value to the attacker. It may provide the attacker with information about the operating systems in use, possibly usernames (by using tools such as enum or commands such as nbtstat), and whether connections to high ports are allowed. There is no guarantee that the username determined using tools such as enum or nbtstat will be the same as the username portion of an e-mail address.

The amount of information gathered during the scanning process and the value of that information is dependent on the type of the target organization. If the target organization is a corporate enterprise, they will most likely have an enterprise firewall that will prevent the attacker from scanning the users' machines, determining operating systems, and gathering usernames. The use of tools such as firewalk may provide evidence that connections on high ports to internal machines are being blocked by a firewall. If the target organization or individuals reside on a university or residential broadband network, the results of an attacker's scans might be of some value. Such networks do not appear to take the same level of precautions as their corporate counterparts, therefore leaving their systems more open to scanning.

The purpose of scanning is to proactively identify vulnerable operating systems or application servers. In the case of a browser vulnerability, scanning is relevant to the degree of identifying the operating system in use. In the lab environment, the operating system on the target machine is Windows XP, which includes Internet Explorer 6. While it is not strictly scanning per se, a Google search revealed various statistical websites reporting that Internet Explorer 6 is the leading browser – accounting for approximately two-thirds of browser usage.

Exploiting the System

At this point, let's assume the attacker has identified the e-mail address gcih2004@yahoo.com as belonging to an individual in the organization he wishes to compromise. If the attacker wants the user to click on a URL that points to the InternetExploiter exploit code on a web server, he has to have a web server where he can place the malicious HTML. The web server is most likely going to be a machine that the attacker has compromised – not one of his own machines. In this scenario, the attacker has somehow compromised the Redhat 8.0 machine at 192.168.1.4 and gained root access. He then installs thttpd 2.25b and starts the thttpd daemon. From the Redhat 8.0 box, the attacker

then retrieves the InternetExploiter v0.1 exploit code from Berend-Jan Wever's website, http://www.edup.tudelft.nl/~bjwever/exploits/InternetExploiter.zip, unzips it, and places it in the appropriate directory for thttpd. By default, thttpd v2.25b serves up webpages from the /var/www/thttpd/html/ directory. The attacker's web server is ready.

The next step is to attempt to get the individual using that e-mail address to visit a specific URL. There are any number of ways to get a user to click on a URL depending on how much information is available about that person. If the attacker knows that the target individual is interested in computer security, they could send an e-mail along the lines of "Hey, I saw your post on such-and-such mailing list regarding self-decrypting, polymorphic shellcode. You seem to really know your stuff. I was wondering if you could review a whitepaper I have written on the subject? The whitepaper is located at http://www.malicious.com/whitepaper.html. Thanks!". For the purposes of this practical, a fictitious piece of spam e-mail was used. The piece of spam contains two links for the user, one if they are interested in receiving more spam and the other link if they do not wish to receive any more spam. In an effort to deceive the target individual, both links point to the same webpage – where InternetExploiter is waiting for a visit. The attacker can provide any number of links that ultimately all point to InternetExploiter.

This exploit opens a backdoor on the target machine. How does the attacker know what IP address has the backdoor running on it? The answer lies in the web server's logs. As the recipients of the e-mail click on the links, they make an HTTP request to the web server for the webpage containing InternetExploiter. A log entry is generated for each request:

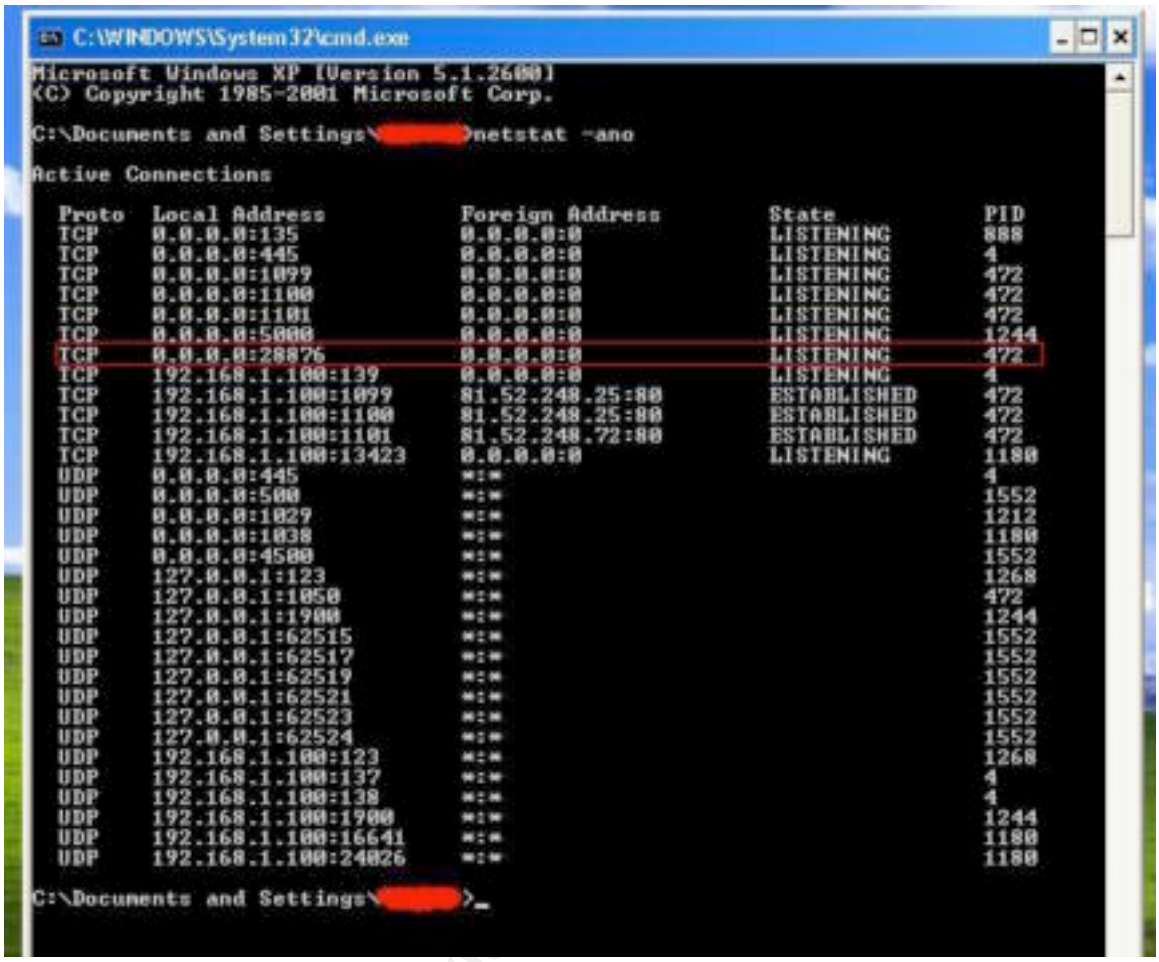
```
192.168.1.100 - - [05/Dec/2004:18:17:10 -0600] "GET /InternetExploiter.html
HTTP/1.1" 200 13562  "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

For the sake of clarity, the lengthy referrer field from the log entry that is present when clicking on the link from Yahoo was removed. The first field in the log entry indicates the source IP address, 192.168.1.100, where the HTTP request came from. Thus, 192.168.1.100 is where the attacker might find the backdoor running.

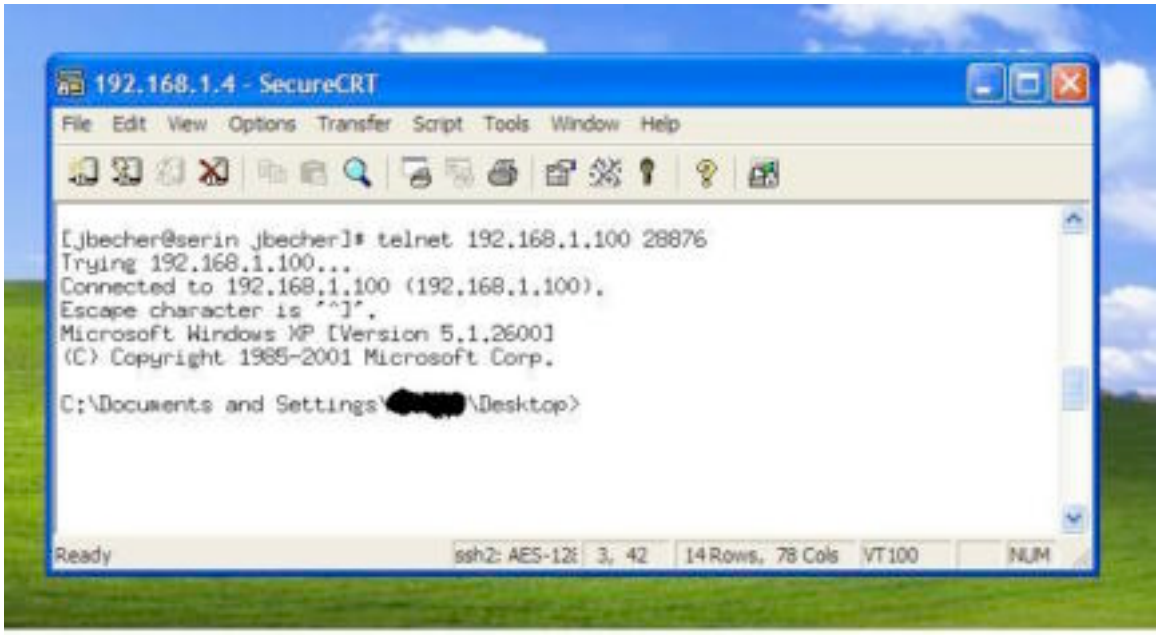
A series of three different exploit attempts was conducted – 1) with the patch not installed and anti-virus disabled; 2) with the patch not installed and anti-virus enabled; and 3) with the patch installed and anti-virus enabled.

Case 1 – Patch not installed, anti-virus disabled

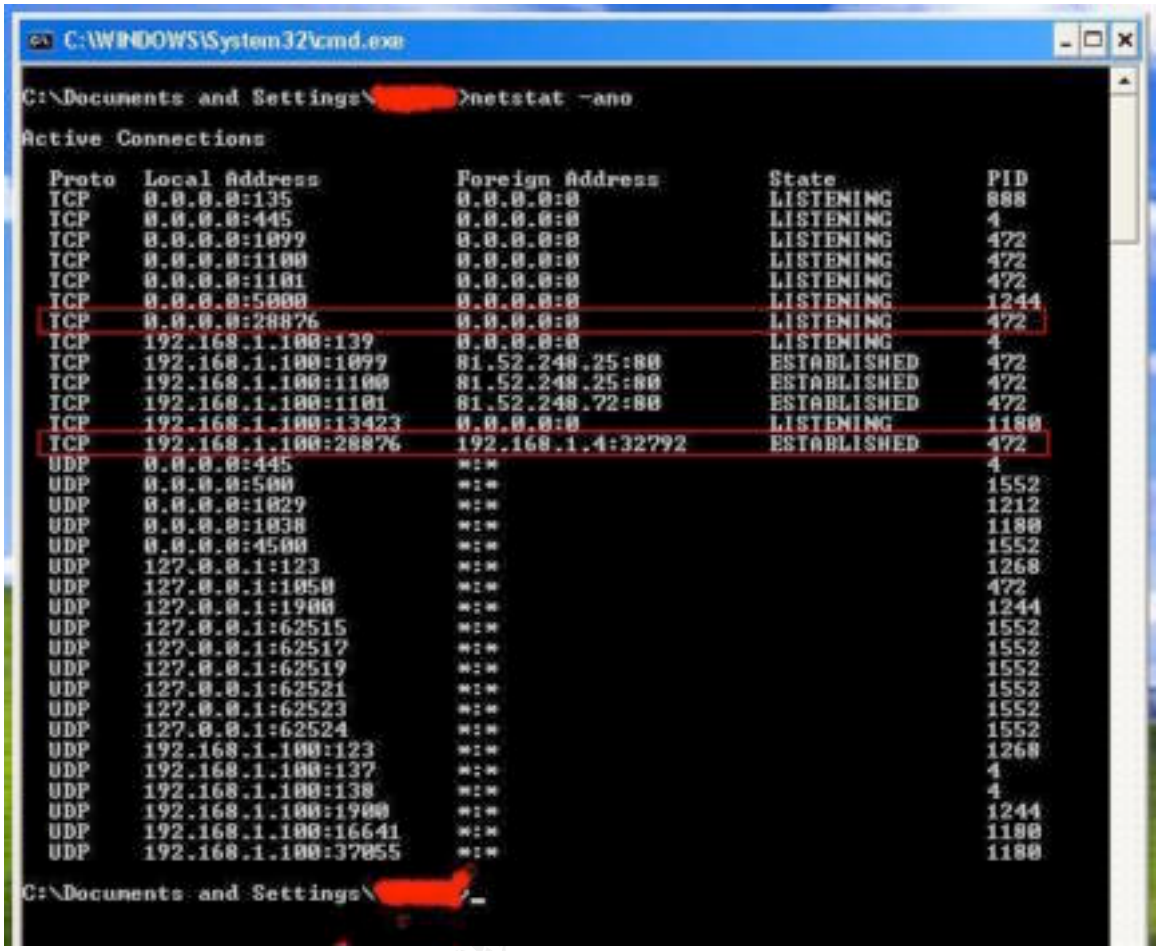
In this test case the patch from MS04-040 had not been applied. The Symantec Anti-Virus Client service had been disabled to simulate not having an anti-virus product installed. As illustrated below, the exploit successfully created the backdoor on the target machine.



The attacker, noticing from the web server logs that a machine has retrieved the exploit code, quickly checks to see if the backdoor was successfully created. From the attacker's view, the exploit was successful.

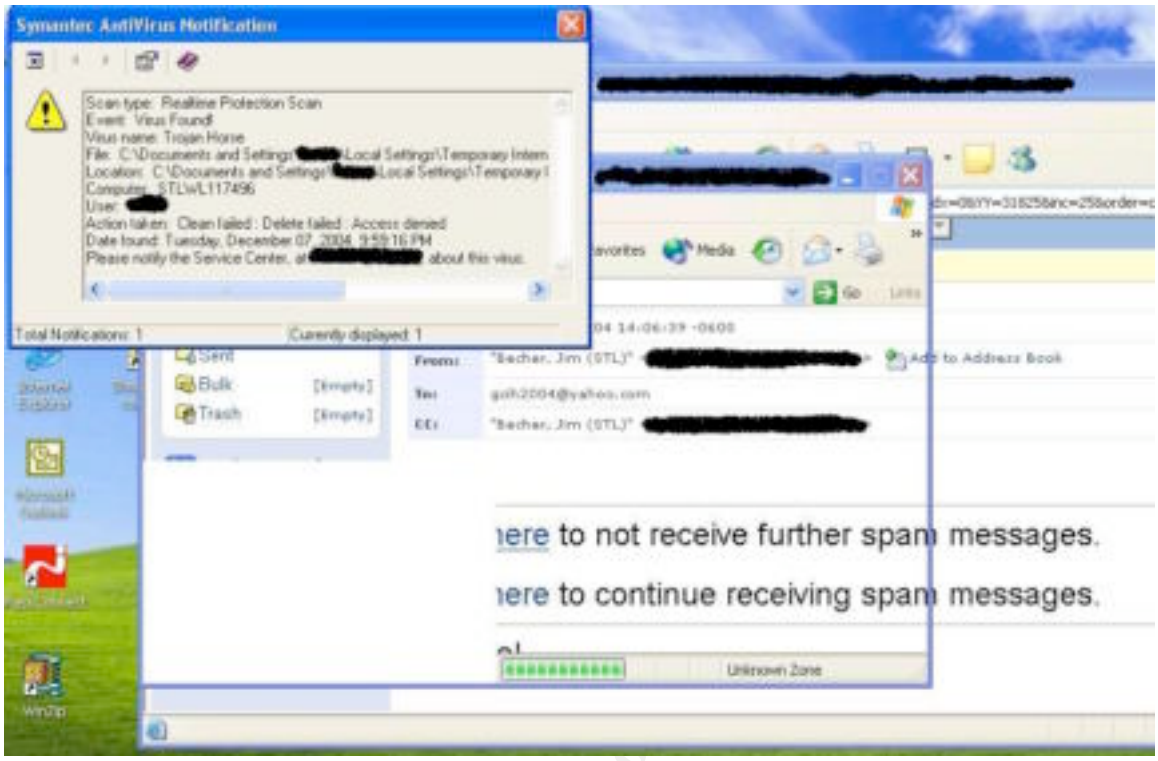


Now that the attacker has connected to his newly created backdoor, the connection is visible in the “netstat –ano” output.

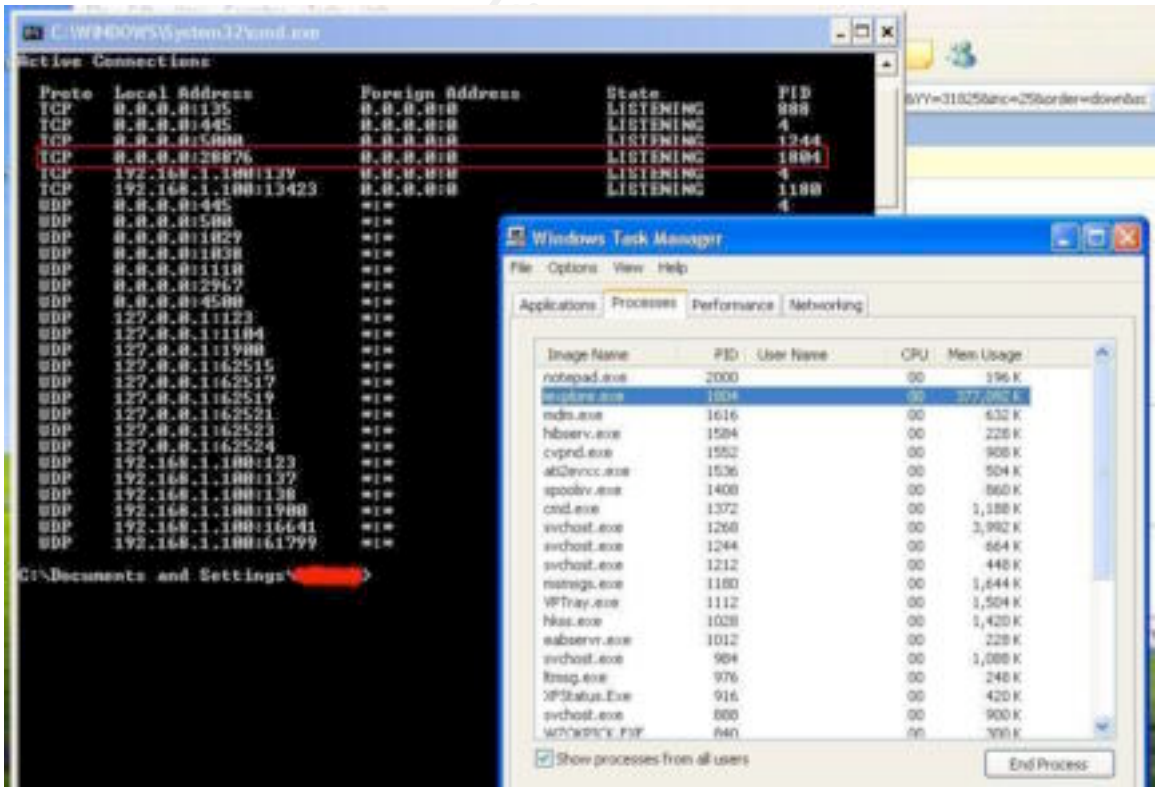


Case 2 – Patch not installed, anti-virus enabled

In this test case, the patch from MS04-040 had not been applied, and the Symantec Anti-Virus Client service was enabled. This configuration would be similar to most Windows machines (with the exception of Windows XP Service Pack 2 machines) between 2 November 2004 and 1 December 2004. This is the timeframe that the exploit was released, but the patch was not available. As illustrated below, the anti-virus product detected the creation of the backdoor…

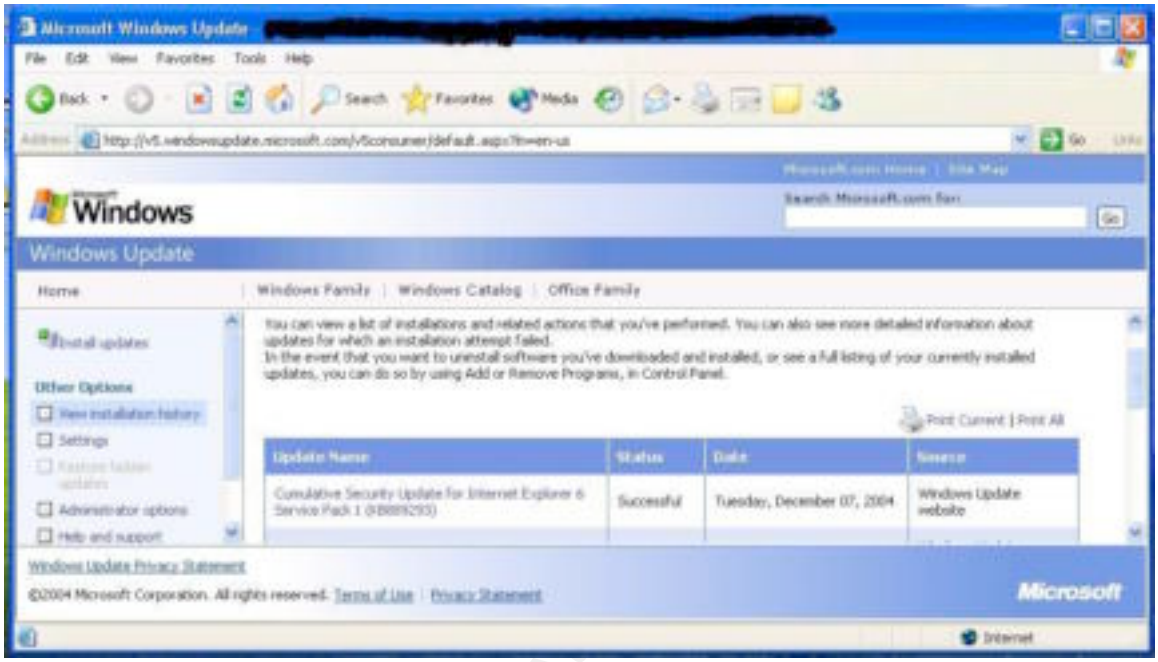


… but did not prevent it. The machine was still compromised.

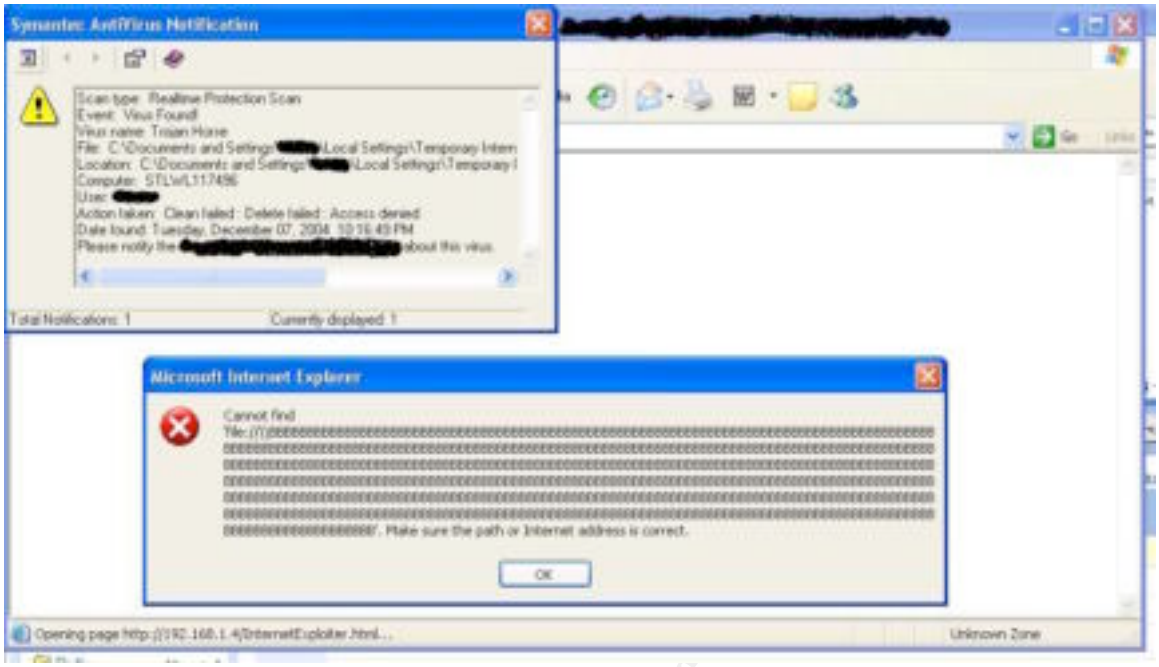


Case 3 – Patch installed, anti-virus enabled

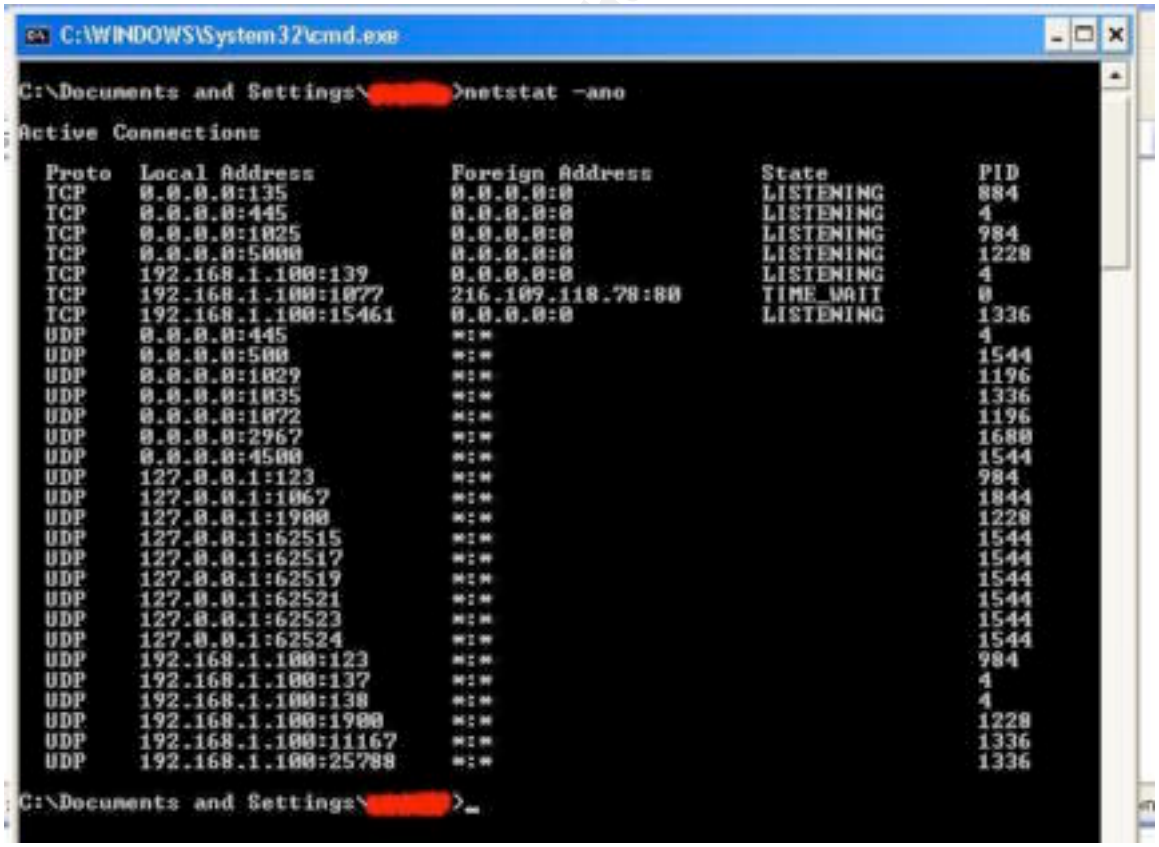
In this test case, the patch from MS04-040 had been applied and the Symantec Anti-Virus Client service was enabled.



This configuration would hopefully be similar to the configuration of Windows machines after 1 December 2004 – after the patch was released from Microsoft. As illustrated below, the anti-virus product detected the attempted creation of the backdoor…



… and the patch prevented the machine from being compromised.



Keeping Access
The attacker will have to act quickly to take advantage of the newly created command shell backdoor on the target system. In the test cases, it was noticed that upon closing the Internet Explorer window that had been opened by clicking on the URL to the malicious HTML, the backdoor closed as well. The Internet Explorer window would also be closed when the user logs out or reboots the machine.

In order for the attacker to keep access to the compromised machine, he will need to initiate a connection to the backdoor before the user closes the Internet Explorer window associated with the backdoor. The attacker will then need to take additional steps to ensure that he can maintain access to the box if the user logs out, a reboot occurs, the machine changes its IP address (DHCP), and so forth. If the attacker was interested in continued access to a single machine, or a small handful of machines, then shoveling a command shell or GUI to a machine the attacker controls might be acceptable. If the attacker is interested in handling a significant quantity of compromised machines, then this might not be practical.

The most logical choice for maintaining access to a significant number of machines would be installing or infecting the machine with malware that connects back to a central location (e.g., botnet). There are several benefits that the attacker enjoys by going this route:

- not having to know what IP address the machine is currently on;
- not having to worry about accepting the significant number of shoveled command shells or GUIs from all the compromised machines;
- the ability to script or automate the command and control of a significant number of machines;
- ingress policies on firewalls are usually more restrictive than egress policies.

The downside to not using application-level trojan horse backdoor suites (such as VNC, Sub7, Back Orifice 2000) is that the attacker has to sacrifice some functionality. However, more and more features are being added to bot malware.

In and of itself, the InternetExploiter exploit code will not provide an attacker an avenue of reliable, continued access.

Covering Tracks
The attacker has successfully used InternetExploiter to compromise a Windows XP machine and is now sitting at a backdoor command shell prompt. Up until now the privileges of the user who visited the malicious webpage have not been discussed – as it would not have impacted the test cases.

InternetExploiter does not involve placing executables/DLLs, modifying system configuration settings, etc. After closing the Internet Explorer window associated with the backdoor, the only remnants that could be found that indicated malicious activity had occurred were the Symantec Anti-Virus Event Logs in the Application

Log. The text contained in the Application Event Logs was basically the same information that was presented to the user via a pop-up when the machine was compromised.

If the attacker were interested in eliminating these traces from the Application Event Log, he would have to delete or clear the application event log. Either one of these actions may cause additional scrutiny of the situation. And trying to remove this evidence from the Application Event Log is probably pointless, as Symantec Anti-Virus Corporate Edition client has a means of reporting virus/trojan horse infections back to the Symantec System Center. Evidence that the machine encountered a trojan horse is still available – and again, combined with a missing/cleared Application Event Log it might bring additional attention.

In fact, the user that was used in the test cases was in the Administrators group. If the attacker was so inclined, they would have privileges to delete or clear the Application Log.

If additional malware was employed on the system to maintain access, then the attacker will need to take appropriate steps to cover his tracks for those activities.

From the network view, there are probably several ways the attacker could avoid detection of either the initial exploitation or the subsequent activity. An example of each will be addressed. First, the attacker can tunnel InternetExploiter in over HTTPS to avoid detection of the initial exploitation. Instead of providing the URL of http://192.168.1.4/InternetExploiter.html in the fictitious spam e-mail to the user, the attacker could have provided the URL https://192.168.1.4/InternetExploiter.html. The encrypted request would have slipped right by passive network IDS. Another method of obscuring the network tracks of the post-exploit activities that may help avoid detection is to change the shellcode that is delivered. In an e-mail conversation with Berend-Jan Wever, he indicated that substituting in alternative shellcode was a possibility. The alternative shell code could be used to modify the port the backdoor listens on, shovel a command shell out, or to download-and-execute additional malware. As for the latter, LURHQ did an advisory on this very scenario.

The Incident Handling Process

Preparation

The SANS Track 4 course materials has a very appropriate quote for preparation, and that is that “chance truly favors the prepared mind.” There are various forms of preparation that an organization could employ to increase awareness and reduce the likelihood and impact if this exploit was being used in their environment.

The more advanced warning that an organization receives about a threat to its environment, the more time they have to prepare. Berend-Jan Wever's initial analysis appeared on 24 October 2004. On 2 November the exploit code was released. The patch for this vulnerability wasn't released until 1 December. By monitoring mailing lists such as BugTraq, an organization could have gained about a month of advanced warning about this vulnerability.

An organization's incident response policy might dictate notification requirements as well as the response strategies for various types of incidents. When the incident response team became aware of the vulnerability, notifying information security management (or CSO/ISSO) would have been prudent. Once the exploit was released, additional notifications could have been sent to the group(s) responsible for maintaining the organization's networks, systems (servers and desktops), anti-virus tools, intrusion detection systems, and firewalls. The notification should have included what to look for and how to look for machines that have been compromised. The incident response policy should stipulate the response strategy in the cases of incidents involving malicious code or unauthorized access.

An appendix to an organization's incident response policy should be an emergency contact list and conference bridge information. In case of an incident that occurs after hours or that impacts the normal communications methods, having cell phone, pager, and home phone numbers handy would allow an organization to react quickly. The larger and more geographically disperse the organization is, the more important the emergency contact list becomes. The emergency contact list should include at least one individual from each of the organization's locations. Conference bridge information should also be included, as incidents often involve more than two or three people in a sizeable organization. The emergency contact list and conference bridge information should be maintained and updated frequently and shared with the core and extended incident handling team.

An enterprise firewall would be one capability that an organization may have in place in order to mitigate incidents. While a firewall might not prevent the initial use of the exploit to open a backdoor command shell, any sanely configured firewall should prevent the attacker from utilizing the backdoor. In order for the attacker to use the backdoor, the organization's firewall would have to allow inbound connections from the Internet to the internal address space for TCP port 28876. In addition, port address translation (PAT) of unrouteable internal address space would also provide a similar protection. The attacker could get the exploit to occur, but the IP address that is logged in the webserver log is the PAT'ing device.

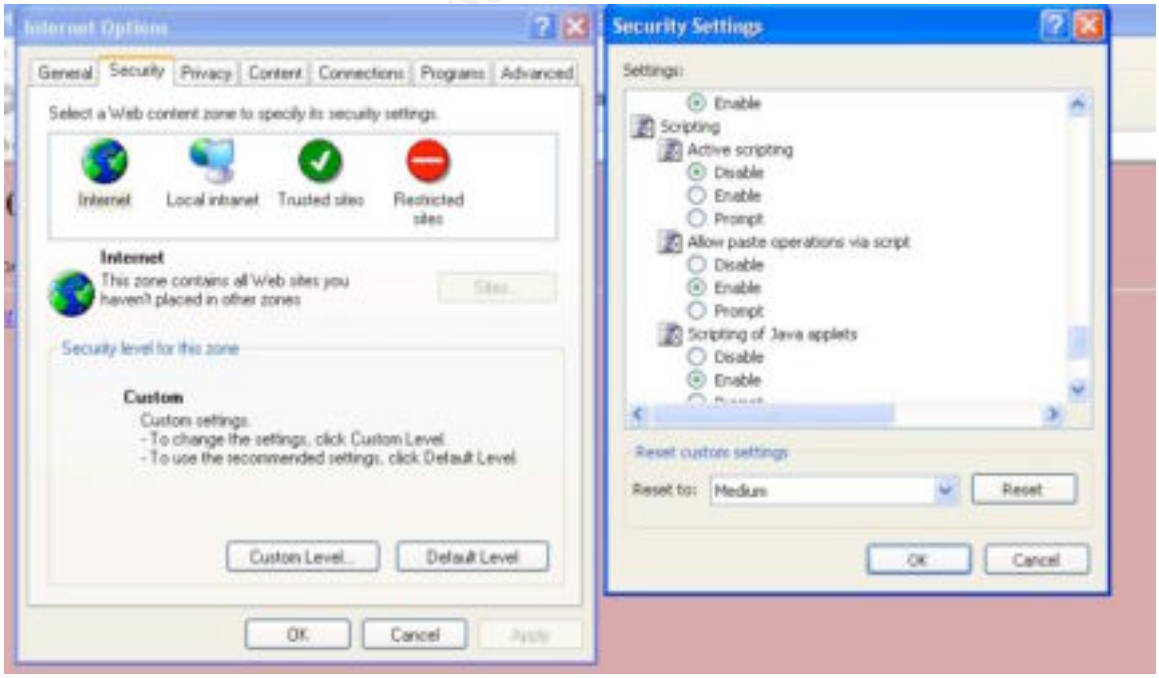
Anti-virus and passive IDS have shown that it did not provide effective countermeasures against InternetExploiter. They may have alerted to the presence of the exploit or the backdoor being created, but were unable to keep it

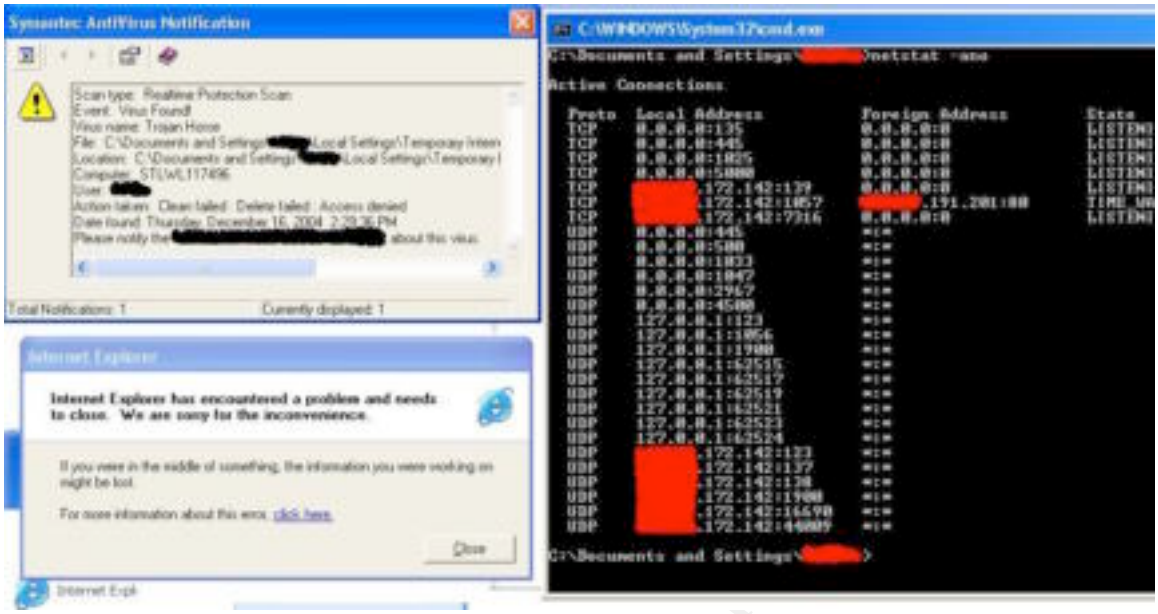
from occurring. They would be able to provide a listing of systems that were potentially or confirmed to have been compromised.

Intrusion Prevention Systems (IPS) may be able to provide an effective countermeasure against exploitation of this vulnerability. Network IPS are basically IDS that are placed in the flow of traffic, and have the capability to block malicious network traffic. If the IDS engines that can detect the malicious payload are placed in the flow of traffic, it is assumed that the IPS can make the decision not to forward the packet(s) containing the InternetExploiter code to the client that requested it.

An aggressive patching policy would provide an effective countermeasure. For those organizations that had rolled out Service Pack 2 to all their Windows XP workstations, they were protected, as Windows XP Service Pack 2 was not vulnerable. Service Pack 2 for Windows XP was released in early August of 2004. If the organization's workstations were Windows 2000, Windows XP or Windows XP Service Pack 1, an aggressive patch policy would have provided an effective countermeasure, but only once the patch had been released. In this case, the exploit was available for almost a month before a patch was available.

There are other preventative steps that could be taken to minimize the chance or impact of exploitation, but they may not be practical. Disabling scripting or moving to another web browser would minimize the impact of the exploit code. When scripting on Internet Explorer was disabled and it was exposed to the exploit code, the browser crashed and the backdoor was not created.





As the discussion on BugTraq unfolded, other browsers were found to crash due to the HTML that was generated out of the “mangleme” tool. In the original post where the tool was announced, Zalewski speculated that some of the other browsers are likely exploitable via vulnerabilities found by the tool.

Identification

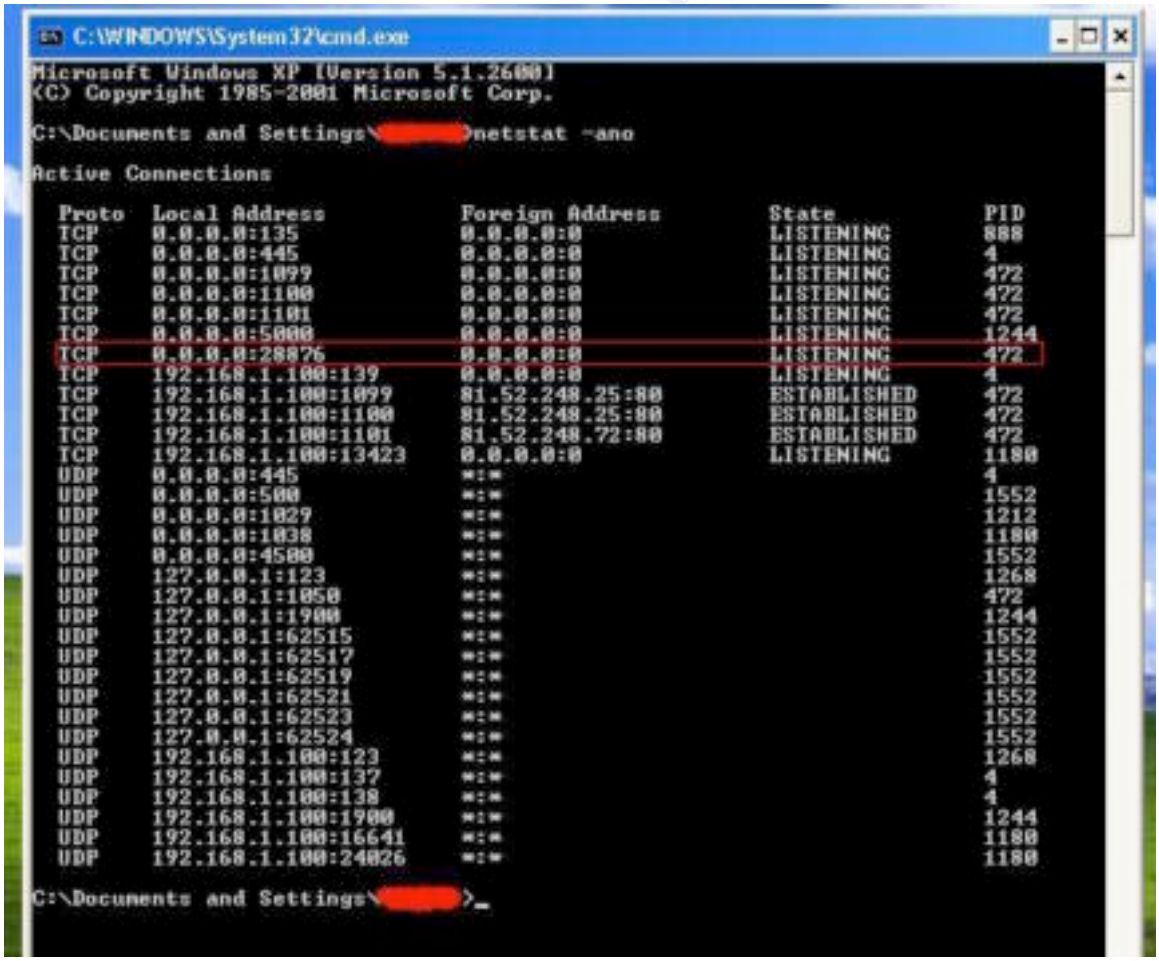
The timeline necessary to respond to the use, detection, and mitigation of this attack depends on which countermeasures were in place and whether the attacker is actively accessing the backdoor. In the case where an enterprise firewall and PAT are in use and the attacker is not actively accessing the backdoor, the incident handling team would most likely not be deployed. They would rely on the desktop support organization to address the machine(s) in question. In the case where the attacker is actively accessing the backdoor, the incident handling team should be deployed immediately. The machine in question would be removed from the network, and enterprise scanning for TCP port 28876 would commence. The Information Systems Security Officer (ISSO) or Chief Security Officer (CSO) would be notified.

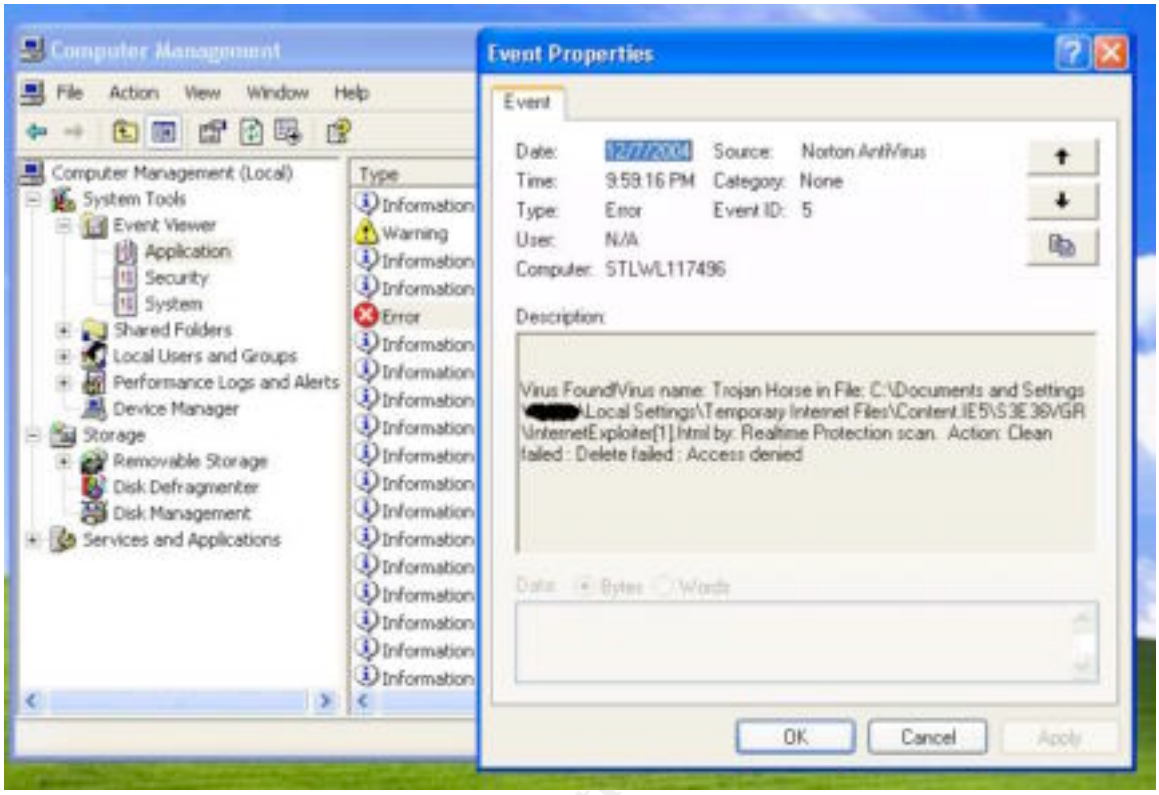
Detection and notification of a possible incident can come from automated tools such as IDS, anti-virus, and port scanners; or detection and notification might come from a user or system administrator that noticed unusual activity on their machine. The automated tools, as well as the administrator or user, will provide the machine name or IP address in their notification. The steps taken to determine whether the machine has been compromised should be documented in incident response procedures and be performed using trusted binaries. The commands referenced below should be trusted binaries that the responder has brought with them. Some organizations have compiled diskettes and CD-ROMs that contain batch scripts that run a series of commands in an automated fashion

so that they get consistent discovery information. If the organization wishes to pursue the incident legally, proper documentation of the exact steps, commands, and output from the machine in question will need to be preserved.

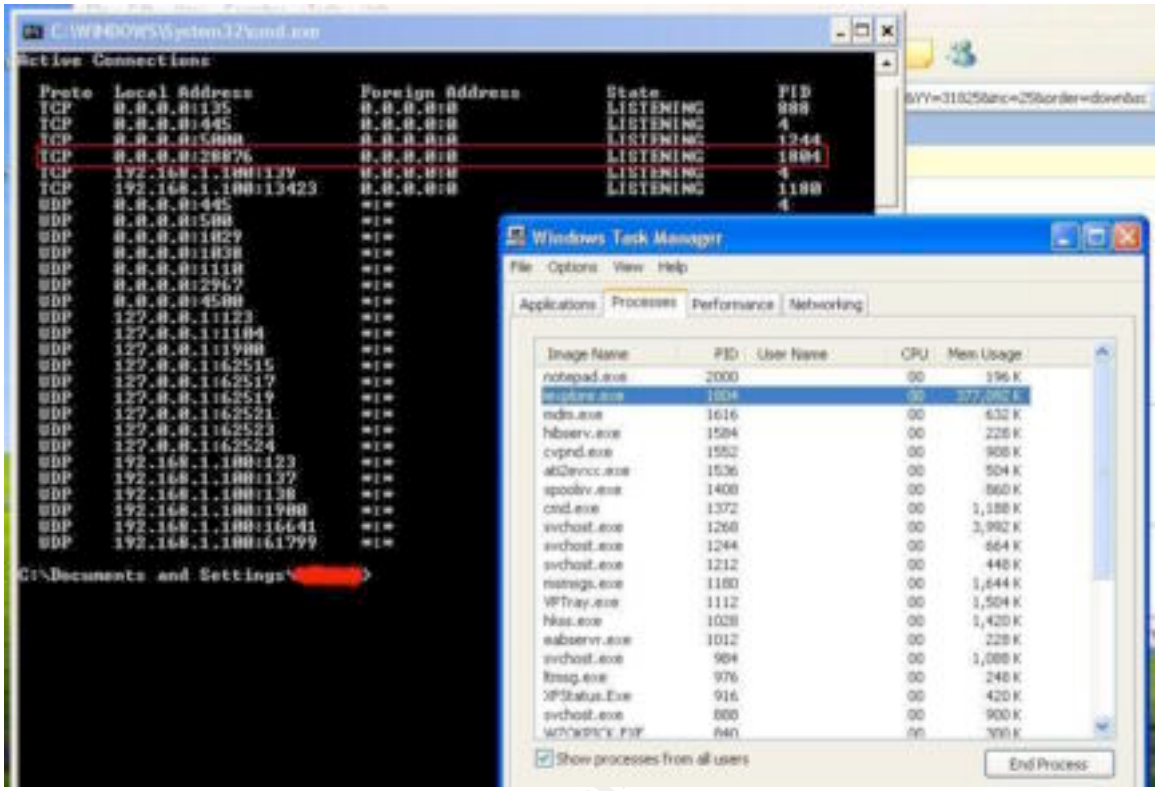
The first step to determine whether an incident has occurred or not is to determine if the system in question is vulnerable. The system is vulnerable if it is not Windows XP Service Pack 2, or if it does not have the MS04-040 patch applied. Checking the service pack and patch level can be done through an enterprise patch management tool, or by using tools such as hfnetchk, psinfo, MSBA, or Windows Update. Once the machine in question is identified as vulnerable, a check should be performed for the existence of a process listening on TCP port 28876. If such a process exists, it is necessary to see if the process ID associated with that listening port is Internet Explorer. If so, and by connecting to the port, a Windows command shell is provided – that would be confirmation that an incident has occurred.

Here is a graphical representation of the process described in the preceding paragraph:

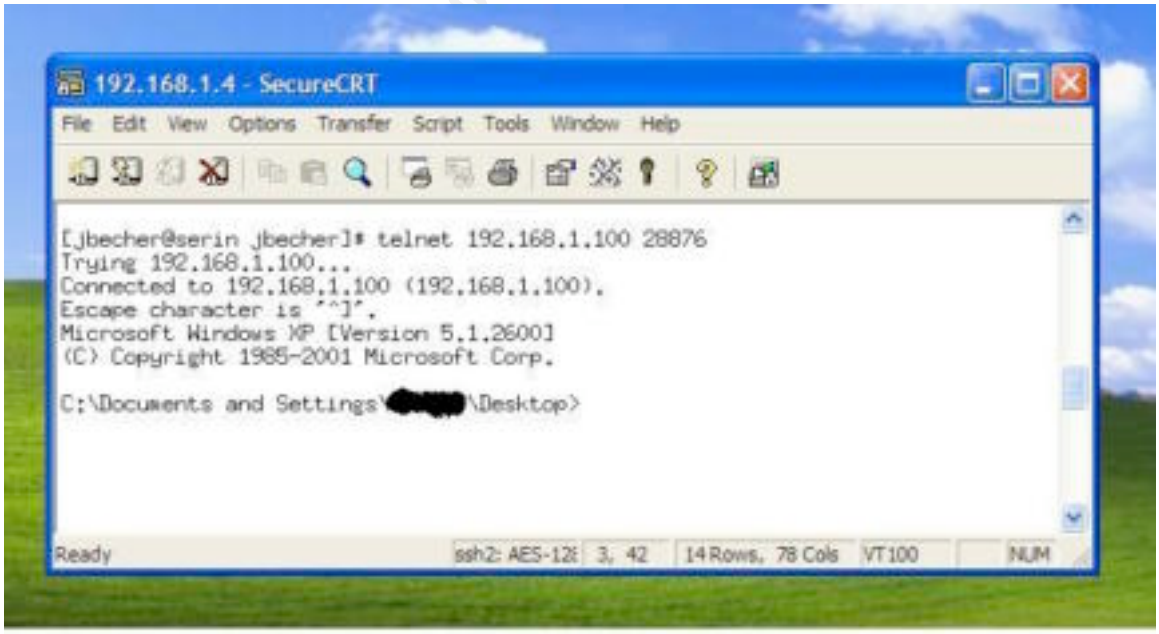




As indicated below, the machine has a process listening on TCP port 28876. The process ID of the process listening on that port is 1804. Note that this process ID is different than the process ID of the previous image – the previous image was from the successful exploitation in Case 1. The image below is from the successful exploitation in Case 2. In order to determine which application is associated with 1804, Task Manager is used.



It has been established that Internet Explorer with process ID 1804 is listening on TCP port 28876. To see if there is a Windows command shell there, it is necessary to connect to the port.



The “netstat –ano” output will let us know if the attacker is currently accessing the machine. But there is the possibility that the attacker has already used the backdoor, performed some other nefarious deeds, and disconnected. It is also possible that the attacker has not yet, or cannot, access the backdoor. Either way, a security incident should be declared.

If the machine has been confirmed to have been actively exploited, it should be immediately disconnected from the network and a hard shutdown performed. The technical support person that has physical access to the machine should label it as compromised and store it in a locked cabinet or room until the organization makes the determination to just rebuild and patch, or to perform a forensic investigation. The technical support person should document each piece of equipment that was collected.

There are three countermeasures that were tested that provided varying levels of protection against the exploit. Obviously, applying the MS04-040 patch or applying Service Pack 2 prevented the exploit from occurring and kept Internet Explorer from crashing. Since the InternetExploiter exploit code took advantage of JavaScript – disabling scripting in the browser was another countermeasure that was successful against the attack – that is, successful in that the backdoor was not created, however, the browser still crashed. Given the impact to the web browsing experience, it is not realistic or practical to disable scripting.

Containment

Unfortunately, it appears most of the countermeasures have come up short when completely trying to address the issue. Two of the countermeasures tested were successful in completely controlling the problem – applying the vendor-provided service pack and applying the patch. Disabling scripting in the browser severely impacts the web browsing experience and does not keep the browser from crashing when it encounters InternetExploiter.

The use of an alternative browser is probably impractical based solely on the existence of this vulnerability, especially since many other browsers were also found to crash. In addition, they are possibly exploitable when exposed to the razor-sharp shards of HTML that were generated by the “mangleme” tool. A malicious website could simply redirect the website visitor to the appropriate malicious HTML, based on the USER_AGENT provided in the initial web request.

Anti-virus proved itself as a worthy detection mechanism, but did not seem to provide much in the way of prevention. Unless the attacker is on the internal network, enterprise firewalls or port address translation (PAT) should prevent active exploitation of the compromised system, but they do not prevent the backdoor from being created. If shell-shoveling or trojan downloader payload is sent, then it’s a whole different ballgame. Desktop-based protection, such as Cisco’s CSA and ISS’s Desktop Protector, may block the exploit and keep it from

successfully creating the backdoor, but access to these products was not available. Applying the patch and/or service pack from the vendor was the best countermeasure tested – it kept the backdoor from being created, and Internet Explorer did not crash.

The process and associated screenshots of assessing the incident were included in the Identification step.

As this is a client-side vulnerability in Internet Explorer, it is expected that only user workstations would be impacted. A server is normally not going to be used for general Internet web browsing or reading e-mail. The fertile ground to the attacker would be the user workstations – taking advantage of the law of averages that an ordinary user inside the target organization will click on the URL to the malicious HTML. The containment strategy used for user workstations is different than the containment strategy used for servers.

The process of containment for user workstations that have been compromised would be a system rebuild. For a single user workstation, this containment approach is minimal as it impacts only one user. In the author's organization, the workstations are imaged and any additional applications needed by the user are pushed via an enterprise software distribution method. Users are encouraged to place all data on network drives in the event of a workstation issue. Thus the backup and recovery of a user's workstation would include a re-image and an application push. If the user did maintain some data on the local drive that they needed, the specific directories and files could be copied off the compromised machine before it was re-imaged.

Hypothetically, if the InternetExploiter-compromised workstation was in an environment behind an enterprise-class firewall that was allowing inbound connections from the Internet only to a handful of approved systems and was utilizing PAT, the organization might impose a less severe containment process. The organization might choose to have the user close the Internet Explorer window associated with the backdoor, verify the backdoor is no longer listening, and apply the vendor provided patch. Additional measures, such as running a full virus scan of the system and performing a vulnerability and/or port scan of the machine could be used to confirm that no additional malware infestation has occurred.

If the organization's infrastructure (e.g., proxy, firewall, content monitoring software, etc.) is logging all URLs visited by users, there are additional containment activities that could be done. If the time of compromise can be narrowed down using the time from IDS events or anti-virus log entries, perhaps the URL in question can be identified. Network IDS may also provide the offending host and URL in the alert. The incident handler could use netcat or wget on a unix system to safely retrieve suspect webpages. If the host and URL of the malicious HTML can be identified, the organization can null-route the host

in question. In addition, the appropriate abuse contacts for the address space should be alerted.

Eradication

In the previous incident response step, a containment strategy based on identifying the host or URL was mentioned. In confirming the host and URL that delivered the malicious HTML, the incident handler could have used netcat or wget from a unix platform to safely retrieve the malicious HTML. This leads directly to the eradication step of the incident response process. To make an informed recommendation about the method of eradication that should be used, the incident handler should attempt to determine the root cause based on the evidence from the incident. The evidence could be from the affected system, IDS alerts, proxy logs, or similar items, which hopefully would lead the incident handler to the particular exploit that was used.

In the instances where the author has encountered this exploit being used in the wild, information from the IDS alert has been used to positively identify the exploit. A sanitized text output of one of the ISS Real Secure IDS alerts from one of those instances is detailed below:

Date/Time	2004-11-15 00:46:33 CST
Tag Name	HTTP_IE_IFrame_BO
Alert Name	HTTP_IE_IFrame_BO
Severity	High
Observance Type	Intrusion Detection
Combined Event Count	1
Cleared Flag	false
Target IP Address	192.168.249.189
Target Object Name	1816
Target Object Type	Target Port
Source IP Address	83.149.86.131
SourcePort Name	80
Sensor DNS Name	network_sensor_1
Sensor IP Address	192.168.224.243
Sensor Name	network_sensor_1
:intruder-ip-addr	83.149.86.131
:intruder-port	80
:server	83.149.86.131
:URL	/indexms.html
:victim-ip-addr	192.168.249.189
:victim-port	1816
algorithm-id	2107036
Packet DestinationAddress	192.168.249.189
Packet DestinationPort	1816
Packet SourceAddress	83.149.86.131

Packet SourcePort	80
Packet SourcePortName	HTTP

Armed with the IP address and URL from the IDS alert, wget was used to retrieve the malicious HTML on a unix machine. The IDS did in fact alert on another instance of the HTTP_IE_IFRAME_BO event. The HTML retrieved was InternetExploiter v0.1. Thus, the root cause of the incident had been identified.

Now that the root cause has been identified, the incident handler can make recommendations for eradication. It does not appear that there is additional code pushed to the system by default, nor are critical registry keys changed by InternetExploiter. In lab testing, the backdoor is closed when the associated Internet Explorer window is closed. The eradication was concluded in the lab testing by applying the MS04-040 patch.

If the environment supports it, and there is evidence that the attacker accessed the system via the backdoor, the recommendation for eradication would change. Once an attacker has achieved this level of access to the system, a complete system rebuild would be in order. Forensic investigation for prosecution purposes or intellectual curiosity can occur on the forensic image or backup that was made during the containment step.

There are additional steps that the incident handler can perform or recommend to ensure that the full extent of the incident has been eradicated. An nmap scan of the enterprise looking for machines where TCP port 28876 is listening wouldn't hurt. If the organization uses an enterprise patch management solution that can generate reports, it would be helpful to know how many other machines do not have the MS04-040 patch applied.

Recovery

Once the method of eradication has been completed, there needs to be validation that the system has been restored to proper working order and is not subject to being compromised in the same manner. This validation will be provided by the user. If the eradication was closing Internet Explorer and applying the patch, the user would experience no differences in the operation of their machine.

The steps required to re-spin and bring a machine up to organizational standards depends on the size and capabilities of the organization. The process may be as manual and resource-intensive as re-building the operating system from vendor media, manually installing applications, then manually securing the machine with patches, anti-virus, and so forth. At the other end of the spectrum, some organizations have a network install process that automates the installation of the operating system, applications, and steps to secure the machine.

The incident handler needs to be assured that the machine is secure to organizational standards before it is returned to operational status. The incident handler can verify that the MS04-040 patch or Service Pack 2 has been applied using WindowsUpdate, hfnetchk, MSBA, or similar. In addition to MS04-040, the incident handler should take the opportunity to ensure that the machine is up to corporate patch levels. In addition to patches, the incident handler should verify that the anti-virus product is running the latest corporate-approved virus definitions. A final step would be to check to ensure that the desktop machine is pulling Group Policy.

Most organizations strive for consistency in their desktop environment, requiring that the incident handler bring the machine up to corporate standards. If opportunities to improve the security of the desktop environment were noted, those changes would be communicated back to the group responsible for the configuration control and base image of corporate desktops.

If the incident handler wanted to test that the vulnerability had been mitigated, they could re-expose the machine to InternetExploiter. A web server could be setup temporarily in a lab environment for testing the patch as well as other countermeasures.

The last aspect of the recovery step would be to enhance monitoring on a temporary or permanent basis. The incident handler might recommend that operational changes be made based on IDS alerts or anti-virus detection. This could include having pages or SNMP traps issued if this was not previously occurring. If the organization is using router access control lists (ACLs) or firewall rules to block traffic to IP addresses that have been involved in previous incidents, enhanced monitoring of those logs should occur. A tool such as SWATCH could be used to monitor text logs and can be configured to page or e-mail based on pattern matching.

Lessons Learned

One of the phrases that rings true from the SANS Track 4 course was “the browser is enemy territory.” The presentation was focused on protecting web and application servers, and not on the browser client; but, the phrase still applies. Unless the organization was running Windows XP Service Pack 2 on all their desktops as of 2 November 2004, then Internet Explorer was a potential entry point to the internal network for an attacker.

The various steps and countermeasures that a site can take in order to protect themselves against this particular attack have been discussed throughout this paper. They are summarized here:

1. Windows XP Service Pack 2
2. MS04-040 patch
3. Host and network Intrusion Prevention Systems (IPS)

4. Application firewall or content security solution
5. Default deny firewall rule set
6. Port address translation (PAT)

The countermeasures listed above can be applied individually or in combination. If the organization was already running Windows XP Service Pack 2 across their entire user base, they were not affected. For operating systems and service packs other than Windows XP Service Pack 2, this is a case where the exploit was publicly released prior to the patch being available. While they were not available in the test environment some of the host-based IPS products were touting that they protected against the exploit by default. The network-based IPS products required updates to detect and prevent the exploit. The updates for the network-based IPS products came out days after the exploit code was released. Again, while the application firewall or content security solutions were not available in the test environment, they appear to provide protection in a similar timeframe to the network-based IPS products. The use of a default deny inbound firewall policy and the use of PAT do not prevent the machines from being exploited by InternetExploiter, but they do offer some protection against the attacker trying to use the newly created backdoors. This scenario highlights the need for defense-in-depth and not relying on any single technology.

References

"16.5 Inline frames: the IFRAME elements" URL: http://www.w3.org/TR/REC-html40/present/frames.html#edef-IFRAME

ACME Laboratories. "thttpd v2.25b." URL: http://www.acme.com/software/thttpd

Atkins, Todd. "SWATCH v3.1.1." 18 July 2004. URL: http://sourceforge.net/projects/swatch/

Common Vulnerabilities and Exposures. "CAN-2004-1050." 17 November 2004. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-1050

DiamondCS. "OpenPorts v1.0." URL: http://www.diamondcs.com.au/openports/

Foundstone. "Fport v2.0." URL: http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/fport.htm

Insecure.org. "nmap v3.75." URL: http://www.insecure.org/nmap/index.html

Internet Security Systems. "Microsoft Internet Explorer IFRAME SRC NAME buffer overflow." URL: http://xforce.iss.net/xforce/xfdb/17889

Internet Security Systems. “Network Sensor XPUs.” URL: http://www.iss.net/db_data/xpu/RSNS.php

Jonkman, Matt. “IE IFRAME Exploit Rule.” 3 November 2004. URL: http://www.bleedingsnort.com/article.php?story=20041102154452 14&query=iframe, http://www.bleedingsnort.com/comment.php?mode=view&cid=22

LURHQ. “IFRAME Exploit via Banner Ads.” 21 November 2004. URL: http://www.lurhq.com/iframeads.html

Microsoft. “Microsoft Baseline Security Analyzer v1.2.1.” URL: http://www.microsoft.com/technet/security/tools/mbsahome.mspx

Microsoft TechNet. “Microsoft Security Bulletin MS04-040.” 1 December 2004. URL: http://www.microsoft.com/technet/security/bulletin/ms04-040.mspx

Ned. “python does mangleme (with IE bugs!).” BugTraq mailing list. 24 October 2004. URL: http://www.securityfocus.com/archive/1/379261

Ritter, Jordan. “Enum v1.0.” URL: http://www.bindview.com/Support/RAZOR/Utilities/Windows/enum_readme.cfm

SamSpade.org. “Sam Spade v1.14.” URL: http://www.samspade.org/ssw/download.html

Schiffman, Mike D. “Firewalk v5.0.” URL: http://www.packetfactory.net/projects/firewalk/

Security Focus. “Microsoft Internet Explorer Malformed IFRAME Remote Buffer Overflow Vulnerability.” 24 October 2004. URL: http://www.securityfocus.com/bid/11515

Shavlik Technologies. “NFNetChk v3.86.” 20 November 2002. URL: http://hfnetchk.shavlik.com/

Snort.org. URL: http://www.snort.org/dl/rules/

SysInternals. “PsInfo v1.62.” 9 August 2004. URL: http://www.sysinternals.com/ntw2k/freeware/psinfo.shtml

TechTarget. “heap – a Whatis.com definition”. URL: http://whatis.techtarget.com/definition/0,,sid9_gci212239,00.html

TiANWEi. “RegShot v1.7.2.” URL: http://regshot.yeah.net/

Upsdell, Chuck. “Browser News: Stats.” 18 December 2004. URL: http://www.upsdell.com/BrowserNews/stat.htm

US-CERT. “Vulnerability Note VU#842160.” 3 November 2004. URL: http://www.kb.cert.org/vuls/id/842160

“Using inline frames to embed documents into HTML documents.” URL: http://www.cs.tut.fi/~jkorpela/html/iframe.html.

W3Schools. “Browser Statistics.” December 2004. URL: http://www.w3schools.com/browsers/browsers_stats.asp

Wever, Berend-Jan. “InternetExploiter.zip.” 2 November 2004. URL: http://www.edup.tudelft.nl/~bjwever/exploits/InternetExploiter.zip

Wever, Berend-Jan. “Re: [Full-Disclosure] python does mangleme (with IE bugs!).” BugTraq mailing list. 25 October 2004. URL: http://www.securityfocus.com/archive/1/379341

Wever, Berend-Jan. “Skylined: The Homepage For Absolutely Nothing.” 2 November 2004. URL: http://www.edup.tudelft.nl/~bjwever/menu.php

Wever, Berend-Jan. “w32_bind_0free_loop.c.” URL: http://www.edup.tudelft.nl/~bjwever/shellcode/w32_bind_0free_loop.c

Zalewski, Michal. “mangleme.” October 2004. URL: http://lcamtuf.coredump.cx/soft/mangleme.tgz

Zalewski, Michal. “Update: Web browsers – a mini-farce (MSIE gives in).“ BugTraq mailing list. 22 October 2004. URL: http://www.securityfocus.com/archive/1/379207

Zalewski, Michal. “Web browsers – a mini-farce.“ BugTraq mailing list. 18 October 2004. URL: http://www.securityfocus.com/archive/1/378632