Revisiting the Code Red Worm

GCIH Incident Handling and Hacker Exploits

Practical

Submitted by Ravila Helen White

August 20, 2001

Table of Contents

Introduction

The purpose of this paper is to satisfy the practical requirement of the SANS Incident Handling and Hacker exploits. While this paper will focus on the exploit used in the July and August Code Red Worm debacle, it will also identify how this particular exploit could have been avoided within the business community or any entity that employs technical professionals through the application of countermeasures.

Constraints

Constraint 1 – References to CRv1 refers to the original worm reported on July 11, 2001.

Constraint 2 – References to CRv2 refers to the fixed CRv1 which flooded the Internet in force on July 19, 2001.

Constraint 3 – References to CRII in this document refers to the version of Code Red which backdoors exploited systems and appeared on August 4, 2001.

Constraint 3 – References to Code Red is a general all encompassing reference to the entire Code Red exploit.

Practical Requirements

This practical will examine and exploit and malicious program. The actual code used in the exploit will not be examined at much length. The following subjects will be reviewed throughout this paper:

Exploit Details
Name: name of exploit

Variants: name of different variants of the exploit, if any

Operating System: operating systems impacted

Protocols/Services: protocols or services that the exploit uses

Brief Description: 1-2 sentence description

Protocol Description: A description of the protocol(s) used in the exploit will be covered

Description of variants: In most cases there are variants based on the original exploits. Therefore variants will also be considered.

How the exploit works: A detailed description of how the exploit works and why it is able to exploit the particular vulnerability in the protocol or application program. A step-by-step analysis of the actions the exploit takes should also be covered in this section.

Diagram: sample output (screen captures, packet captures, etc. as appropriate) of running the exploit on a test network.

How to use the exploit: A description of the programs that exist to exploit this vulnerability will be provided and an explanation of how to use the program(s) Finally, an explanation on how one would manually run the exploit against a system if an automated program did not exist.

Signature of the attack: Sniffer, log file, or other output as appropriate will be provided to show what to look for to determine an attack.

How to protect against it: Provide information on how to protect against the exploit and/or fix the vulnerability. This will be a two-part section. First, what can someone who is running the vulnerable software do so that his or her system cannot be compromised? Second, what could or should the vendor do to fix the vulnerability?

Source code/ Pseudo code: Links to the source code can be found, and a description of the pseudo code in your own words. This section should clearly explain what the code is doing in order to exploit the vulnerability.

Background

On June 18, 2001 Microsoft Security issued a security bulletin related to a buffer overflow vulnerability in the Index Server 2.0 for Microsoft® Windows NT® 4.0 and the Indexing Service for Windows® 2000. This vulnerability was found in the ISAPI extensions installed during the installation of IIS. In particular the idq.dll. The idq.dll provides support for administrative scripts and Internet Data Queries and as such must run in the System context. Plainly put, the idq.dll can interact with server-side client requests. This coupled with the unchecked buffer provides an attacker the ability to initiate a web session with a vulnerable server and conduct a buffer overrun attack. Once the attack has been initiated successfully, the attacker can then execute code on the exploited web server with the same privileges of the System, in effect giving the attacker complete control of the server and any potential network resources it is connected to.

While it can be a tedious and exhaustive chore to research, test and apply the various patches which software vendors provide, it is best to take notice of any bulletin one receives which exposes vulnerabilities which allow attackers to gain either root or administrator access on a given system. By far, this particular bulletin appears to have been ignored for the most part. This was seen in force on July 19, 2001 when over 350,000 IIS servers were compromised by the CRv2 worm. While some of the compromised servers where those owned by private individuals there were many compromised servers owned by government and professional entities. Entities should have taken the information provided in the June 18, 2001 bulletin and protected not only their systems but by doing so, mitigated the spread of the worm and the degradation of the internet services during the July 19, 2001 onslaught.

Evidence of CRv1 was first seen on July 11, 2001 and appears to have emanated from a University in China. By July 19, 2001 more than 350,000 infections were reported.

Exploits in Depth

With the spread of the Internet, malicious attackers can now easily band together to combine the their knowledge and develop attacks which utilize more than one attack signature. Just as the security community is teaching the practicing the defense in depth methodology, the Blackhat community is practicing offenses in depth. This is the methodology employed by the creator(s) of the Code Red worm.

While the average security professional may not have the expertise to dissect the code used in the Code Red worm, they should understand the techniques employed which in turn may allow and individual the opportunity to deploy countermeasures they are familiar with against such attacks.

Buffer Overflows

In computer science from a software perspective a buffer refers to a reserved segment of memory used to hold data while it is begin processed. In software development, buffers are created to hold some amount of data from each of the files

that will be read or written. The amount of data is defined within the code and when combined with bounds checking will prevent someone malicious or non-malicious from putting more data into the buffer than allocated. Bounds checking is an additional piece of code or a statement that tracks how much data is being written to our buffer. Should the attempt we made to write more data then has been allocated; the bounds check will deny the write request. In figure 1 below we see a normal stack where the overall routine of Get Shoes calls two subroutines of 'get sandals' and 'get evening shoes'. The routines and amount of space allocated to each routine is called in a back to front fashion. With a bounds checking mechanism each routine will be checked to ensure that only the allocated space is assigned to the routine. Once the last subroutine is executed the Return Pointer will return control of the program to Get More Shoes for further instructions.

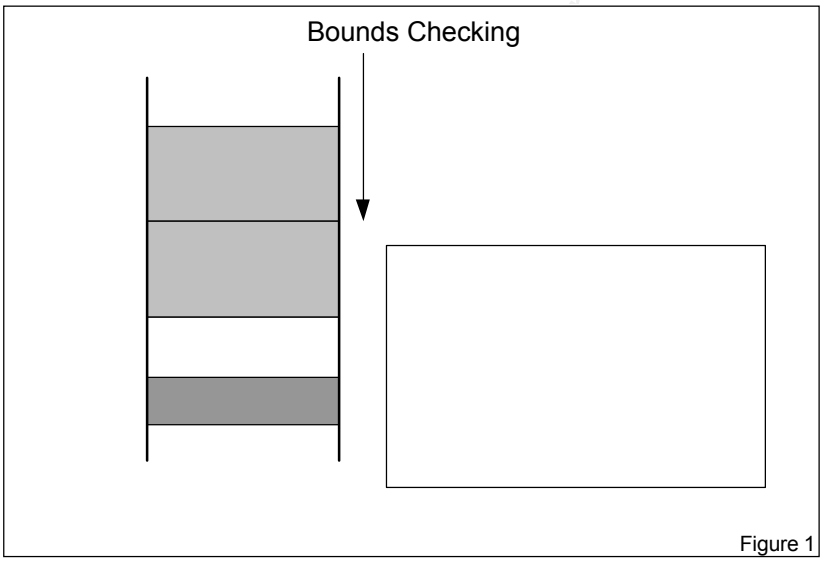


Figure 1

When a buffer has been written and no bounds checking has been implemented, the risk is run of more data being forced into the buffer than has been allocated thus causing a buffer overflow. Buffer overflows take advantage of the way information is store by computer programs. Generally, programs rely upon subroutines to execute and perform various tasks. Once the subroutines have executed the requested task a call is made back to the main program. However when a buffer overflow occurs the overflowing data will go into the next subroutines space creating a cascading effect until it finally reaches the Return Pointer. When the Return Pointer has been compromised we see that rather than control being returned to the actual program control is returned in a program installed by an attacker.

In the figure 2 we see that that bounds checking has not been enabled to verify the amount of data for either subroutine. Here an attacker overflows the first subroutine causing and overflow into the next subroutine. The second subroutines are overflowed as a result of the data coming from the first subroutine. The overflowing

data (usually machine specific byte code) sent by the attacker then executes a command and inserts a new address for a return pointer full of malicious code tuned to the attackers needs.

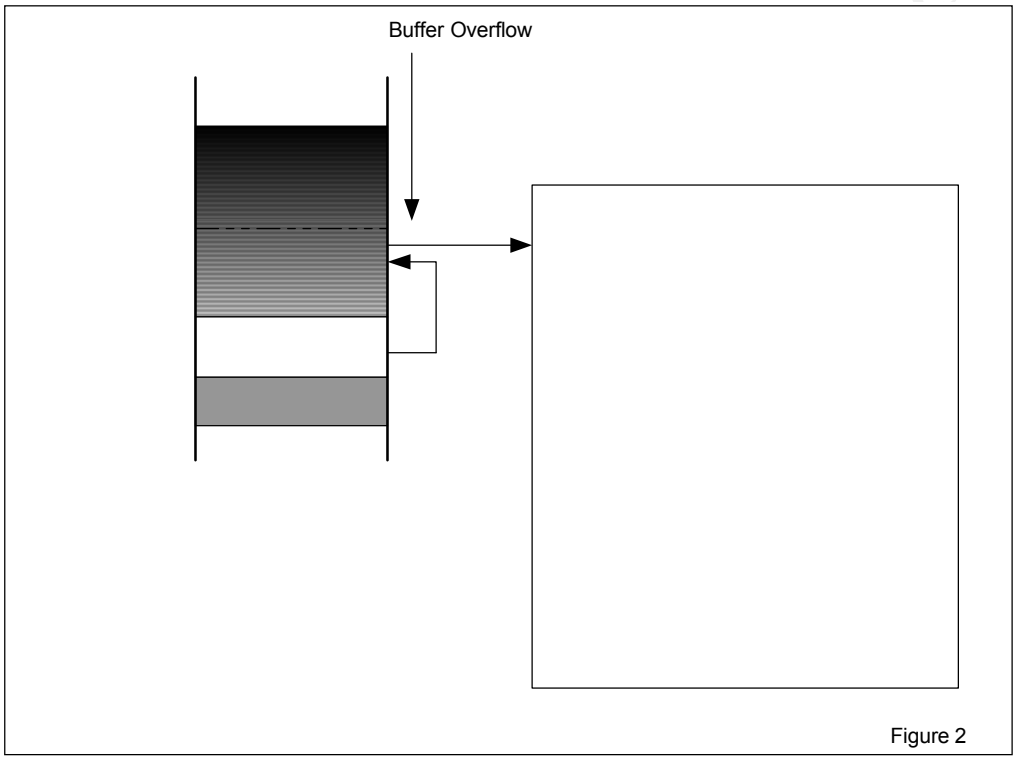


Figure 2

CGI Exploitation

CGI is the Common Gateway Interfaced used for executing server-side programs with the pages ones sees while they are surfing the net. CGI programs are executed then a client initiates an HTTP GET request. The program runs on the server in the background and appears in HTML format on the web page the client is viewing. This is very useful when the GET request executes a command that will perhaps gather information that will be delivered to the requesting recipient via e-mail or to find out the status of an online order placed with an E-commerce company. The vulnerability of CGI is seen when the CGI program as the ability to execute operating systems commands thus allowing an attacker to send malicious commands to the operating system. An attacker does not need a web browser to initiate a GET request. GET requests can be sent from the Run command in Windows NT, TELNET or even terminal windows.

While the client sends the request it is actually the Web server executing the program on behalf of the requesting client. The program will run at whatever privileges it has been given by the Web Server. This becomes a problem is the

privileges are that of Root in *nix environments or Administrator in Windows environments.

Name

The worm was given then name Code Red by the eEye Digital Security research team of Ryan Permeh and Marc Maiffret in part because of the Chinese origins as well as the fact that they drank Code Red Mountain Dew to keep themselves awake during the analysis of the worm.

Variants

At the writing of this paper there are two official variants of the Code Red Worm. The first variant is referred to as Code Red and the second variant that installed a backdoor on exploited systems is referred to as Code Red II based on the actual words Code Red II round in the code. It should be noted that the first release of Code Red was flawed and later fixed. Thus began the evolution of Code Red. CRv1 released July 11, 2001, and fixed on July 18, 2001 to become CRv2. and finally Code Red II that still runs a buffer overflow but has a different payload.

Operating System

Both versions of the Code Red Worm were written to explicitly exploit the vulnerability report by Microsoft on Window NT 4.0 server running Microsoft IIS 4.0 with Index Server 2.0 and Windows 2000 server running Microsoft IIS 5.0 with the Index service. Unix servers running Apache or even Windows servers running non-Microsoft web server were not vulnerable to the worm.

Code Red was particularly devastating to servers running English (US) versions of Microsoft IIS as it performed web page defacement. Non-English language servers are still vulnerable and while web defacing does not occur DDOS capabilities were still intact.

Protocols/Services

The TCP application level protocol of HTTP the basis for exchange of information over the World Wide Web and used by Microsoft IIS servers to negotiate client/server requests. In particular the HTTP GET command was used in the Code Red worm to exploit the vulnerability reported by Microsoft.

Brief Description

CRv1 is a worm that sends it code as and HTTP request to Microsoft IIS servers running version 2.0 of the Index server and IIS 5. The HTTP request exploits the buffer-overflow vulnerability published by Microsoft on June 18, 2001.

CRv2 exhibits many of the same characteristics as CRv1. However, CRv2 does not deface web pages of exploited hosts but rather it installs a backdoor on a compromised host.

Protocol Description

Three protocols were used in to exploit the idq/ida vulnerability. The first being IP, the second being TCP and the last HTTP

IP

IP is the Internetworking Protocol that provides all of the Internet's data transport services. IP is the base protocol of every other Internet protocol (e.g. TCP, UDP, HTTP, FTP, TELNT, etc). Other Internet protocols are either laid on top of IP or used to support IP from below.

IP is a datagram-oriented protocol and treats each packet is processes independently. This independence forces the requirement that each packet must contain complete addressing information. Additionally, IP does not perform any type of verification to insure that packets sent actually reach their destination or take corrective action if they do not. With regard to error checking only the IP header is checked for which leaves the checksum portion of the packet vulnerable. IP operates at layer 3 also referred to as the Network layer. When we think of the network layer think routers.

TCP

TCP is the Transmission Control Protocol and is based on IP. TCP provides reliable, stream-oriented connections and insures that data sent will reach its destination. The reliability of TCP in part acts as a stabilizing agent to IP. Streaming enables TCP data to be organized as a stream of bytes. Streaming also allows for datagram concealment on a network and finally the Urgent Point mechanism provides flagging of out-of-band data. TCP is known for its reliability. In part this is due to the sequence number used to coordinate which data has been transmitted and received. TCP will arrange for retransmission if it determines that data has been lost. TCP also can adapt to a specific network. If delays are detected, TCP will dynamically learn the delay characteristics of a network and adjust is operation to maximize throughput without overloading the network. Where client and server may be mismatched in send/receive sequences, TCP will coordinate traffic to prevent buffer overflow. TCP operates layer 4 also called the Transport layer.

HTTP

HTTP is the Hypertext Transfer Protocol and is used for distributed, collaborative, hypermedia information systems. It is generic in nature as well as stateless. These two attributes allows HTTP use beyond that of hypertext. It can be used in name servers as well as distributed object management systems. The typing and negotiation of data representations, allows systems to be build independently of the data being transferred. HTTP employs a number of methods for retrieving information:

GET - retrieve whatever information is identified by the Request-URI

HEAD - is identical to GET except that the server must not return any Entity-Body in the response
POST - to request that the destination server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line
PUT - requests that the enclosed entity be stored under the supplied Request-URI

HTTP operates at layer 7 also known as the Application layer of the OSI model.

The Three Faces of Code Red

There are three variants of the Code Red Worm. The first variant had a flaw and was corrected. The second was the first release fixed and the third was modified to backdoor exploited systems. All variants were programmed avoid attacking avoid attacking the 127.x.x.x (loopback) and 224.x.x.x (multicast) subnets.

CRv1

Appears to have been released July 11, 2001, first official report was July 13, 2001 and analysis released on July 17, 2001 by Eeye.com. This worm basically exploited the ida/idq vulnerability reported by Microsoft on June 18, 2001 through an HTTP GET command. Once the ida/idq vulnerability had been exploited the worm was programmed to setup 100 threads of the worm. This allowed for propagation through creating a sequence of random IP addresses based on a statically seeded IP address. The static address caused a bit of a slowdown in the spread of Code Red, as the same IP addresses would be generated on a compromised server, thereby resulting in the exploited servers attacking and re-attacking each other. The 100th thread initiated a check regarding and language version being run on the compromised server. If the version were found to be English (US), then the compromised servers web page would be defaced. Non-English (US) systems 100th thread will attack another computer based on the IP address sequencer. In addition to attacking other vulnerable servers the threads also check for the file c:\notworm. If the file is found the worm becomes dormant, if not found then the threads continue to try to find more vulnerable hosts.

This worm also included a date checking function and based on the date executed different attacks. If the date is past the 20th of the month the threads stopped searching for vulnerable servers and instead perform a DDOS against port 80 of whitehouse.gov by sending 100k bytes of data. Should the date query return values between the 1st and 19th of a month the worm would not attack The White House site just vulnerable web servers.

CRv2

This variant is identical to the above variant with the exception of the way in which the random seed sequence generator acts. The original version caused cross attacks.

Meaning that servers exploited would re-infect other servers a as the seed generator held a static address. Its appears that either very late on July 18, 2001 or very early on July 19, 2001 someone fixed the generator to be truly random which caused the onslaught of attacks and compromised on July 19, 2001.

Code Red II

This variant used the same buffer overflow technique of the aforementioned worms. However this variant was programmed to open a backdoor on compromised systems to allow attackers the ability to keep access once the affected system had been patched. While the earlier worms would execute on both Windows NT and Windows 2000, this variant crashed Windows NT and left Windows 2000 systems open to the backdoor compromise. In addition to the backdoor, a trojaned version of explorer.exe is installed. Propagation to was a bit different. This variant did not look for English (US) versions. Rather it relied upon Chinese versions. Additionally the feature to DDOS whitehouse.gov was disabled. Upon finding a vulnerable system in the Chinese language, the worm will create 600 threads and attempt to spread for 48 hours. Non-Chinese language systems create 300 threads and spread for 24 hours. Finally, once the threads have become dormant the worm forcibly reboots the system, flushing the worm yet leaving the backdoors and Trojans in place.

Variants Info

SANS
http://www.incidents.org/react/code_redII.php

Security Focus BugTraq Archive (search on "Code Red Worm, New information")
http://www.securityfocus.com/

CERT
http://www.cert.org/advisories/CA-2001-23.html

Exploits in Depth

The Code Red Worm is a nicely packaged exploit in depth. The maker(s) were quite thoughtful in it creation. The exploit was accomplished not through some new hack/crack but rather based on the information provided in the Microsoft bulletin, the open design of the Internet and lastly ignorance/lack of action on the part of those who managed servers vulnerable to the compromise.

In addition to the information Microsoft provide is the prevalence of IIS servers being run all over the world by business and private individuals. Most businesses today small or large will have a web server to provide information about their business to anyone who happens to surf by. With the introduction of click and point web authoring, private individuals can also maintain their own web servers as well. In order for the information

to be provided to those who might decided to take a look at what is being offered the underlying system must support a client/server architecture. This means someone and really anyone can query the server for some type of data/response. HTTP GET calls happen everyday, all day, every second of the day without causing bad repercussions. In this case however, the query of GET some.unpatched.host/default.ida coupled with the buffer overflow would allow an attacker to gain control of a server depending on the configuration.

Considering the amount of servers infected we can safely assume that many systems were installed with the out-of-the-box settings related to IIS and minimal if any security measures were taken to harden or secure these servers. This shows the importance of following the guides that have been provided by various security entities related to system hardening. A bit of hardening in this case would have gone a long way. This fact was really highlighted with Code Red II as it took advantage of an earlier report vulnerability that should and could have been patched by the time Code Red II was released, thereby preventing its spread to some degree.

The last major vulnerability that was exploited was the location from which the worm appears to have originated. It appears from analysis that the first sightings of Code Red were from a well known IP block which hackers/crackers flock to take advantage of the lack or laws and knowledge related to the internet and security. Any organization that has been attacked by hosts originating from this corner of the Internet will run into a veritable brick wall when attempting to report exploitation. So it stands to reason that even if the source of the attack is proven, little can or will be done to prevent the attackers from mounting future attacks. If the attacker(s) were from a different part of the Internet and used a zombie to carry out their attack and exploit, little can still be done based on the current state of things in this neck of the Internet.

Below is a rough hypothesis of the line used to develop and introduce the Internet community to the Code Red Worm.

Phase I Reconnaissance and Information Gathering

1. Attacker(s) receive information regarding Microsoft Security Bulletin01-033 that outlined the ida/idq buffer overflow vulnerability.
2. Attacker(s) may have visited the eEye.com site to look over their analysis of the vulnerability and based on the information obtained the amount of data required to attempt a buffer overflow.
3. Attacker(s) gather information from ARIN or some entity which will tell them the information they need for the whitehouse DDOS.

Phase II Test Phase

1. Attacker(s) begin testing buffer overflow on IIS server they either own or on vulnerable servers they've probably already compromised with some other hack/crack.

2. Attacker(s) test a portion of the code to verify functionality of the CGI exploit followed by the buffer overflow. The rest of the code is useless if either of the two core exploits cannot be accomplished.
3. Existing pieces of code (e.g. seed generator) are added.
4. Testing on personal machines or already compromised hosts begins.
5. Adjustments are made based on contained tests.

Phase III Release

1. A mini-recon is performed to find host from which to launch exploit.
2. Hosts are identified and compromised.
3. Traces of hacks are removed and code is left to run from exploited hosts.

Phase IV Wait, Watch & Fix

1. July 11, 2001 is when the first strains of CRv1 were seen by July 18, 2001 an advisory has been issued by Eeye.com regarding the worm including the seed generator flaw which prevents full scale spread.
2. July 18, 2001 someone (probably part of the original team) fixes the seed generator to be truly random thus insuring a full scale spread of the worm.
3. July 19, 2001 the worm runs through the Internet and compromises hosts that have not been patched. This includes hosts behind mis-configured firewalls.

Phase V Why Re-Invent the Wheel

1. Between July 20, 2001 and August 3, 2001 someone modifies the code and re-engineers it to backdoor vulnerable system and install a Trojan to allow them to keep access even after a vulnerable system as been patched.
2. Release of the Code Red II is verified.

Code Red Step by Step

1. HTTP GET against the default.ida
2. Buffer overflow is initiated and new pointer and a new stack for the worm is installed.
3. Worm code begins execution by loading functions GetProcAddress and LoadLibraryA.
4. The later functions then load the following functions:

   From kernel32.dll:

   a. GetSystemTime
   b. CreateThread
   c. CreateFileA
   d. Sleep
   e. GetSystemDefaultLangID
   f. VirtualProtect

   From infocomm.dll:

   g. TcpSockSend

From WS2_32.dll:

h. Socket
i. Connect
j. Send
k. Recv
l. Closesocket

5. Store the based address of w3svc.dll for web defacement if the system is English (US).
6. A check is run for the amount of threads being run and creates new threads until they reach 100. If they reach 100 then the worm begin check for web page defacement.
    a. Check the local system default language, if in English (US) then deface existing web page If not then loop back to step 7.
    b. Sleep for two hours.
    c. Attempt is made to modify the exploited systems web pages in memory by hooking the W3svc.dll making it writable.
    d. The access for TCPSockSend is located and replaced with an address pointing back to the worm resulting in web page defacement.
    e. Hack Thread 100 sleeps for ten hour during which the defaced web page is displayed.
    f. After ten hours the w3svc.dll is returned to its original state, including re-protecting the memory.
    g. Return to step 7.
7. A check for the existence of c:\notworm is executed.
8. If c:\notworm is found then the worm will become dormant.
9. If the c:\notworm is not found then the worm check the computer clock based on UTC. If the date is greater than 20 UTC the worm will begins its DOS against whitehouse.gov if the date is less than 20 UTC the worm will continue trying to infect new systems.
    a. Open a socket and connect to whitehouse.gov on port 80 and send 100kbyes of data
    b. If connection is successful then loop and send 18000h single byes SEND() to whitehouse.gov
    c. After 18000g SEND()'s the worm will sleep for about four and a half hours then loop to step "a" above.
10. Send .ida worm to a host from seed generator on port 80 using multiple SEND()'s to break up packet traffic. Successful SEND the socket is closed and control is returned in step 7.

Diagram

This information to build this diagram was provided by a contributor (lcp@bofh.sh) on Security Focus-bugtraq. The traces were based on a test network involving three

© SANS Institute 2000 - 2005 Author retains full rights.

computers. One computer gathered the data as and infected host attacked a vulnerable host.

CRv2 as seen prior to UTC 20

Lines 1- 10 show the infected computer attacking a vulnerable host. Line four shows the actual HTTP GET/default.ida string. Lines 5 – 7 include the information sent in the buffer overflow as well as the newly exploited server communicating with the infected host. The remaining lines show the threads from the seed generator to begin probing for other vulnerable host. Please take notice of the multiple TCP SYN request from our newly infect host at 192.168.1.105 to a group of IP address which follow no real order. In this case, the ACK returned in 0 as the random host is not available since this was run in a test environment.

Since this capture was taken prior to the 20th of the month none of the IP addresses resolve to whitehouse.gov. Queries to whitehouse.gov will only occur after the 20th of the month as defined in the code.

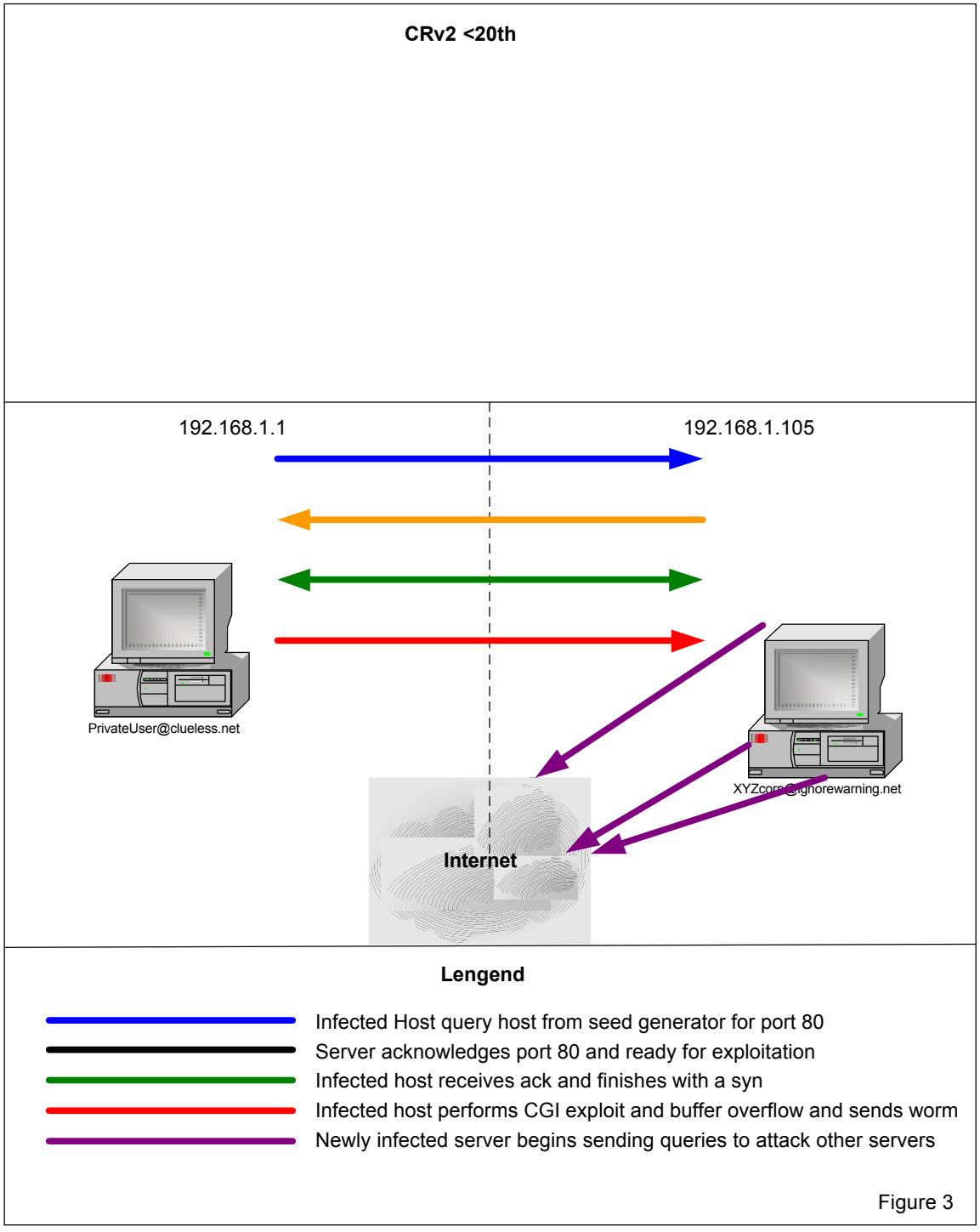


Figure 3

CRv2 attacking whitehouse.gov

Figure 4 below is identical in action to Figure until we get to line 10 where we see and ICMP redirect has been executed. This redirect tells the infected host of 192.168.1.105 to create socket and connect to www.whitehouse.gov on port 80 and send 100k bytes of data. In a non-test environment this would have set up the DOS

as multiple infected hosts would have participated.

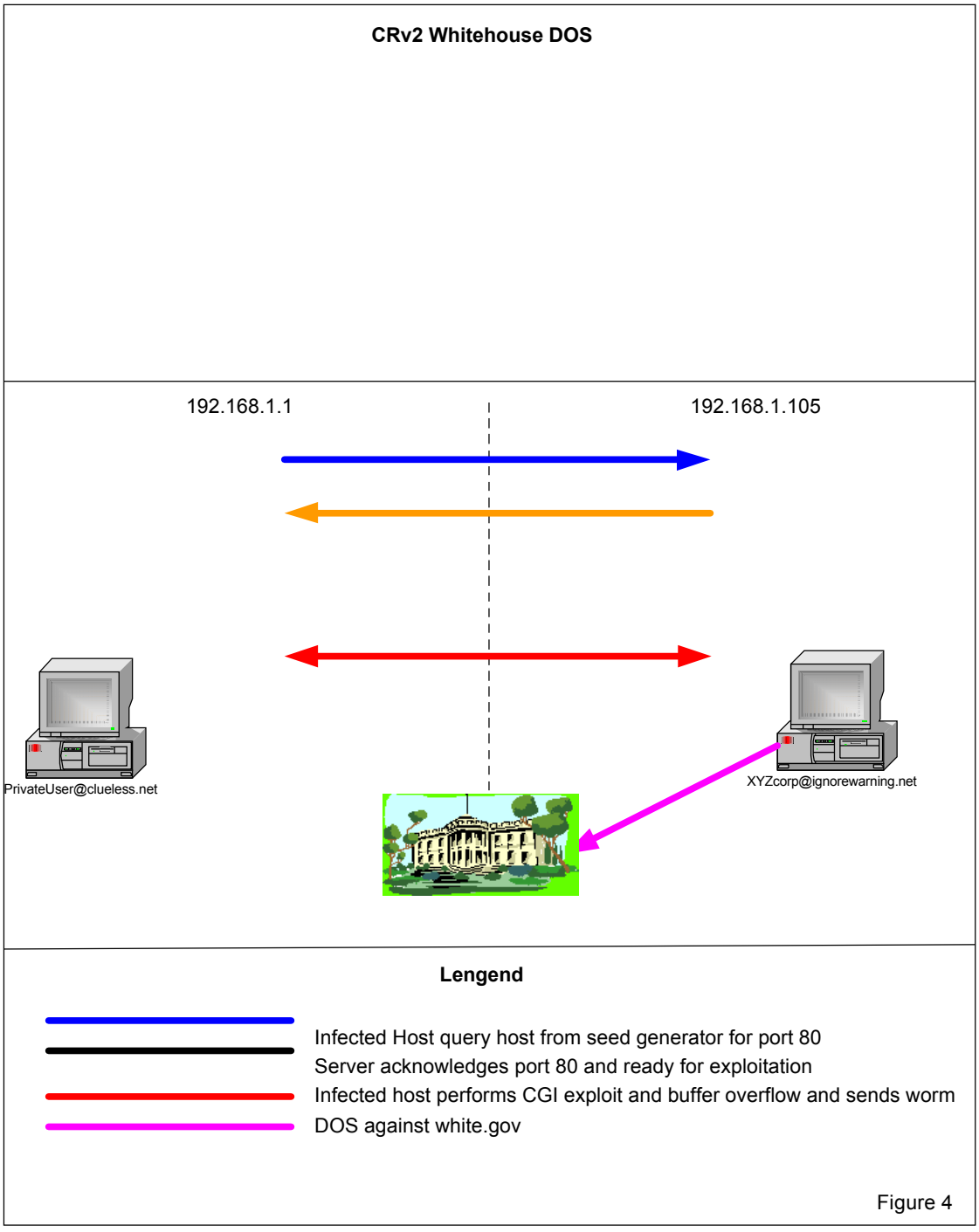


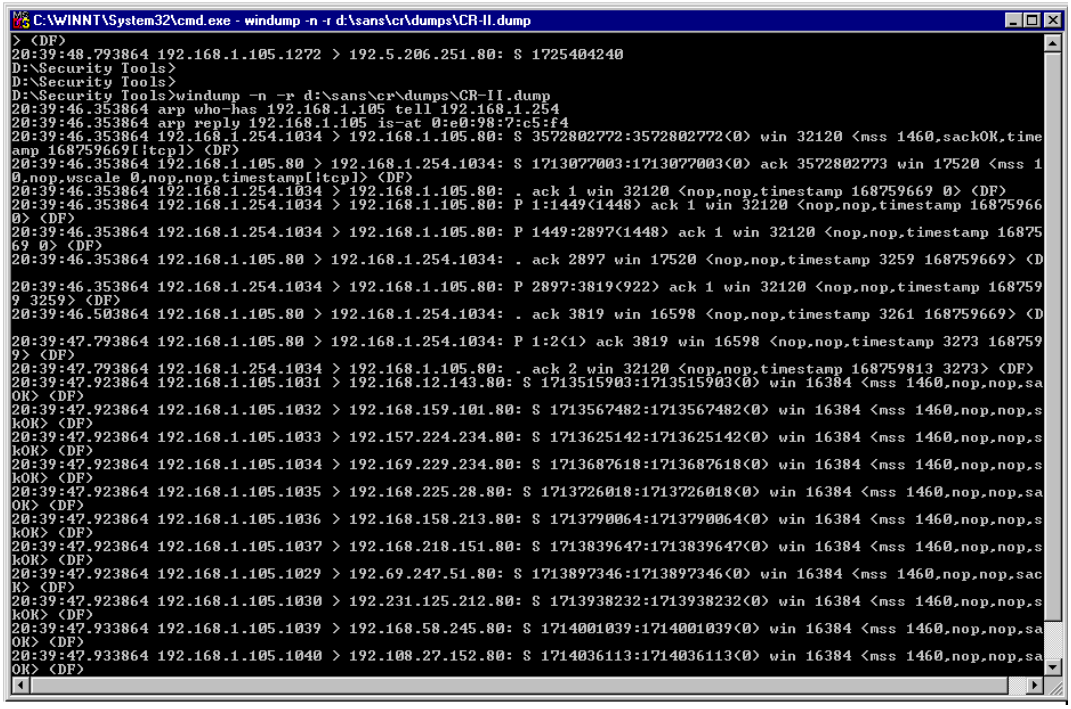
Figure 4

CRII Backdoor + Trojan

Below in Figure 5 are traces of the Code Red II worm. This trace differs from the Trace of CRv2 in that we do not explicitly see the HTTP GET/default.ida. The familiar SYN-ACK-SYN sequence between the infector (192.168.1.254 and the

infected (192.168.1.105) is seen. Afterwhich the the infected host of 192.168.1.105 has initiated is random seed and is now searching for its own set of hosts to infect.

Code Red II Trace



How to use the exploit

CRv1 and CRv2 used three exploits to achieve its goal. CGI exploit layered with a buffer overflow followed by a DDOS against The White House.

CGI Exploit

In examining IIS logs, IDS, logs, Firewalls logs and finally a look at the source code, the first thing seen is the use of the HTTP GET command. Discussed earlier in this paper was CGI exploitation. In this case the CGI exploit was leveled against the default.ida file on vulnerable hosts. Figure 6 below show an actual excerpt from the Code Red code provided by Eeye.com.

CGI Exploit

Figure 6

Buffer Overflow

Once the execution of GET /default.ida was accomplish a buffer overflow was

initiated as seen in Figure 7 below. This buffer overflow was designed to overflow the allocated 240-byte buffer in the default.ida. Once this was accomplished EIP was overwritten with 0x7801CBD3 which is an address within msvcrt.dll. The code at 0x7801CBD3 disassembles to: call ebx. When EIP is overwritten with call ebx, it causes program flow to divert back to the stack. The code on the stack jumps into the worm code that is held in the body of the initial HTTP request. It is at this point that the worm now has gained control of the vulnerable host. We remember that the point of overflowing buffers is to redirect return pointer in a stack to malicious code planted by an attacker. Figure 4 shows buffer overflow.

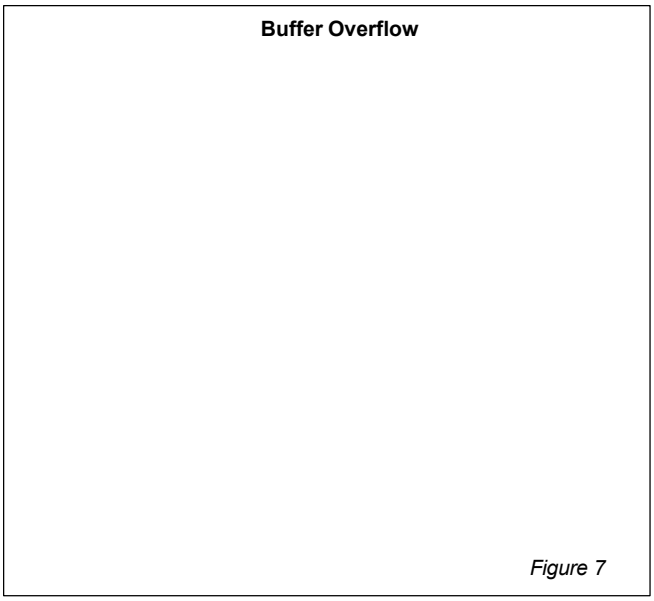


Figure 7

Manually exploiting the idq/ida vulnerability

To manually exploit the ida/ida vulnerability, a malicious attacker can simply issue the following command from a browser, DOS prompt of Telnet session. This can be accomplished by first executing a HTTP GET command against the default.ida followed by an insertion characters whose value is more then the allocation of the buffer vulnerability. Figure 8 shows the data that can be inserted into your browser or a telnet session.

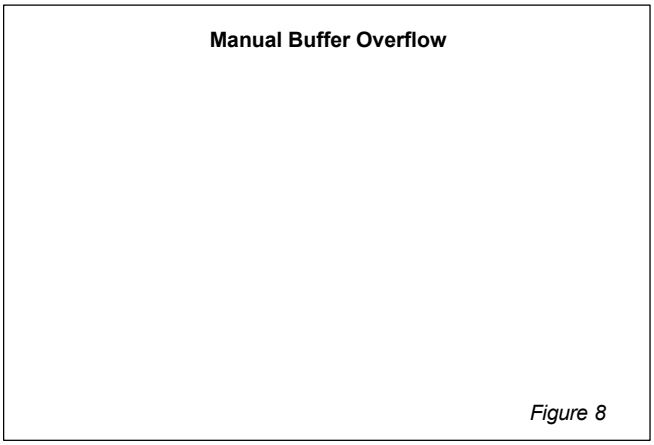


Figure 8

DDOS

A subroutine was contained with in the worm to DDOS the whitehouse.gov site. This subroutine consisted of three functions: WHITEHOUSE_SOCKET_SETUP, WHITEHOUSE_SOCKET_SEND and WHITEHOUSE_SLEEP_LOOP.

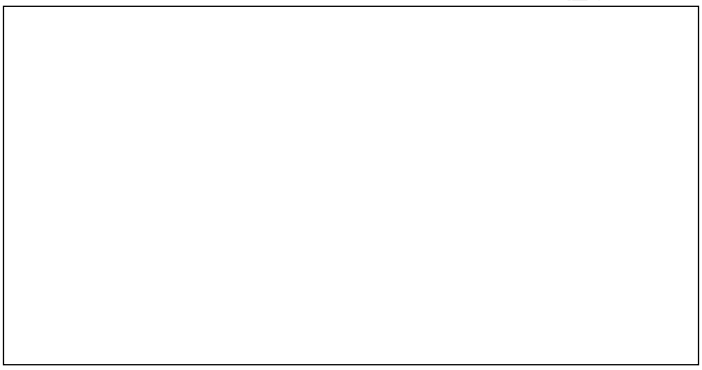
The first function opens a socket on the compromised host and establishes a connection to whitehouse.gov that translates to 198.137.240.91/80. During initiation, 100k bytes of data are sent. The second function is the actually part of the code which performs the DOS by sending 18000h single byte SEND()'s to whitehouse.gov programmed to loop. This amount of data might be trivial if it emanated from a single host. However with the amount of hosts which were compromised the WHITEHOUSE_SOCKET_SEND function amounts to a DDOS against the target site, in this case whitehouse.gov. The third and last function causes the packet flood to stop for four and a half hours. After this time a loop is initiated and the WHITEHOUSE_SOCKET_SETUP function is run again. A cursory glance might lead some to believe that the DDOS would last for only four and half hour intervals. However, when we consider that various compromised hosts varying time zones we understand that the DDOS would have been relentless until the sleep function after the 20th of the month (based on UTC) began.

Signature of the attack

Some attacks are hard to spot. However Code Red was fairly obnoxious and visible. Entities that heeded the Microsoft Security Bulletin 033-01 did have much to worry about except the onslaught of traffic that their routers and firewalls had to manage. For those organizations running English (US) versions of IIS the compromise was readily visible in the form of a hacked web page emblazed with the words, "Welcome to http://www.worm.com!, Hacked By Chinese!". For those systems running non-English (US) versions of IIS, the clue would be either the ungraceful crash of their web server(s), or in increase in processor/network. During the worms threading session typing in 'netstat-an" would show the external or attempted connections from local ports to port 80 of random IP addresses.

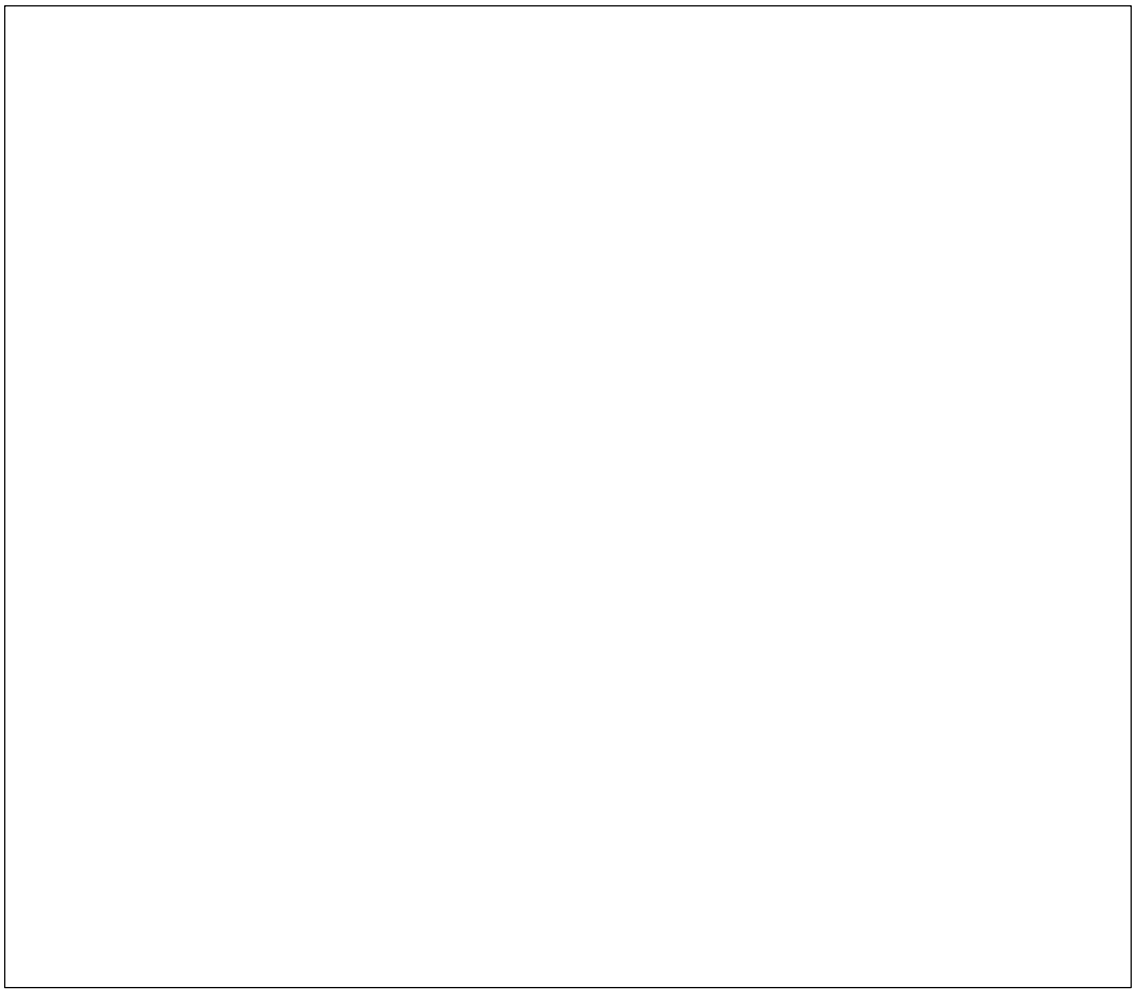
For those entities that did patch their servers prior to the release of Code Red, and in particular e-commerce sites it was of important to at least identify what may have caused an increase in network traffic or failures in network hardware overwhelm by contaminated hosts. In this case a signature to detect the worm was gleaned from IIS logs. The signature was then posted for insertion in IDS systems to recognize the particular attack.

Please examine Figure 9 below. This is the information gleaned from the IIS servers that had been compromised. Note the first command initiated is the HTTP GET command. The GET command here is pointed at the vulnerable default.ida file for which the buffer overflow vulnerability was reported. Next we see a string of alpha characters to initiate the buffer overflow, followed by special and numeric characters.



After the initial HTTP GET, we see a string of character. These characters are the cause and overflow idq.dll once they exceed the 240-byte limitation set in the buffer.

Depending on the firewall and its configuration the following signature would have been seen in Figure 10 below.



With regard to the above log entries. The firewall administrator can determine which interface the Code Red worm is attempting to access as part of the log file indicates what interface the request was made on this is represented in the srcip= field. This is very usefully in an organization where perhaps the internal web servers may have been compromised. A validation of the interface will enable the administrator to determine if the interface serves the internal network or the external network. Likewise the dstif= field will tell the administrator what interface the request is bound for. If this particular interface were an internal interface then the firewall administrator would need to patch any internal systems to help mitigate the spread of Code Red. In this case the interface being hammered by Code Red is an external interface. If this organization were one which participated in the information gathering and host alert effort spearheaded on Bugtraq then the src=field would have provided adequate information regarding the identify of the attacking host. In this case there were attacks originating from within the United States, Europe and Asia based on lookups using ARIN. The attack is stopped by a fairly tight rule set. More than likely HTTP access is allowed within this company but only under very control circumstances. The tight rule set basically defeats the attempt of Code Red to locate and compromise servers within this network.

Examining firewall logs showed activity of the first virus signature peaking on July 19, 2001 and appearing to end on July 21, 2001. The same signature appeared against beginning August 1, 2001, however beginning August 4th, 2001 a new signature appeared. This signature was differentiated by the "XXXX" patterns as well as the "GET default.ida" was no longer called but the host and a specific path. Further examined revealed that CRv2 had been modified to install the backdoor and trojan. With two worm variants loose with differing date functions security, firewall and system administrators can expect to see both of these signatures in their log file separately or together depending on the date.

An interesting side note that must be considered is the abuse of the tool created by eEye.com to help identify vulnerable servers. Since the release of this tools there have been reports of HTTP GET/default.ida?AAAAAAAAAAAAAA appearing in logs. This is a portion of the signature seen when using the eEye scanner. Unless the person seeing such activity has performed the scan then this would indicate that someone was using the tool to locate vulnerable servers and perhaps initiate a manual exploit of the ida/idq vulnerability.

Stopping Code Red

There are a number of ways that entities running vulnerable IIS servers can protect themselves from being exploited by an automated program such as Code Red or those who may attempt to exploit the .idq and .ida vulnerability manually.

Workaround Patching

Remove script mappings for Internet Data Administration (.ida) and Internet Data Query (.idq) files. However, such mappings may be recreated when installing other related software components.

Windows NT Patching

1. Go here: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=30833
2. Select your language from the drop-down list at the top of the page.
3. Click **Next**.
4. Click **Download Now**.
5. Do one of the following:
   - To start the installation immediately, click **Run this program from its current location**.
   - To copy the download to your computer for installation at a later time, click **Save this program to disk**.
6. Click **OK**.
7. Verify that the patch has been installed on the machine, confirm that the following registry key has been created on the machine: **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Hotfix\Q300972**.
8. To verify the individual files, consult the file manifest in Knowledge Base article

Q300972.

Windows 2000 Patching

1. Go here: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=30800
2. Select your language from the drop-down list at the top of the page.
3. Click **Go**.
4. Click **Security Update**.
5. Do one of the following:
   - To start the installation immediately, click **Run this program from its current location**.
   - To copy the download to your computer for installation at a later time, click **Save this program to disk**.
6. Click **OK**.
7. Verify that the patch has been installed on the machine, confirm that the following registry key has been created on the machine: **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Updates\Windows 2000\SP3\Q300972**.
8. Verify the individual files, use the date/time and version information provided in the following registry key: **HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Updates\Windows 2000\SP3\Q300972\Filelist**.

Preventing DDOS Against whitehouse.gov

To prevent a DDOS against the whitehouse.gov the site was moved to a different IP address than targeted by the worm.

Cleaning Code Red II

After Code Red II surfaced, Microsoft provided a cleaner. The cleaner can be found at: http://www.microsoft.com/Downloads/Release.asp?ReleaseID=31878

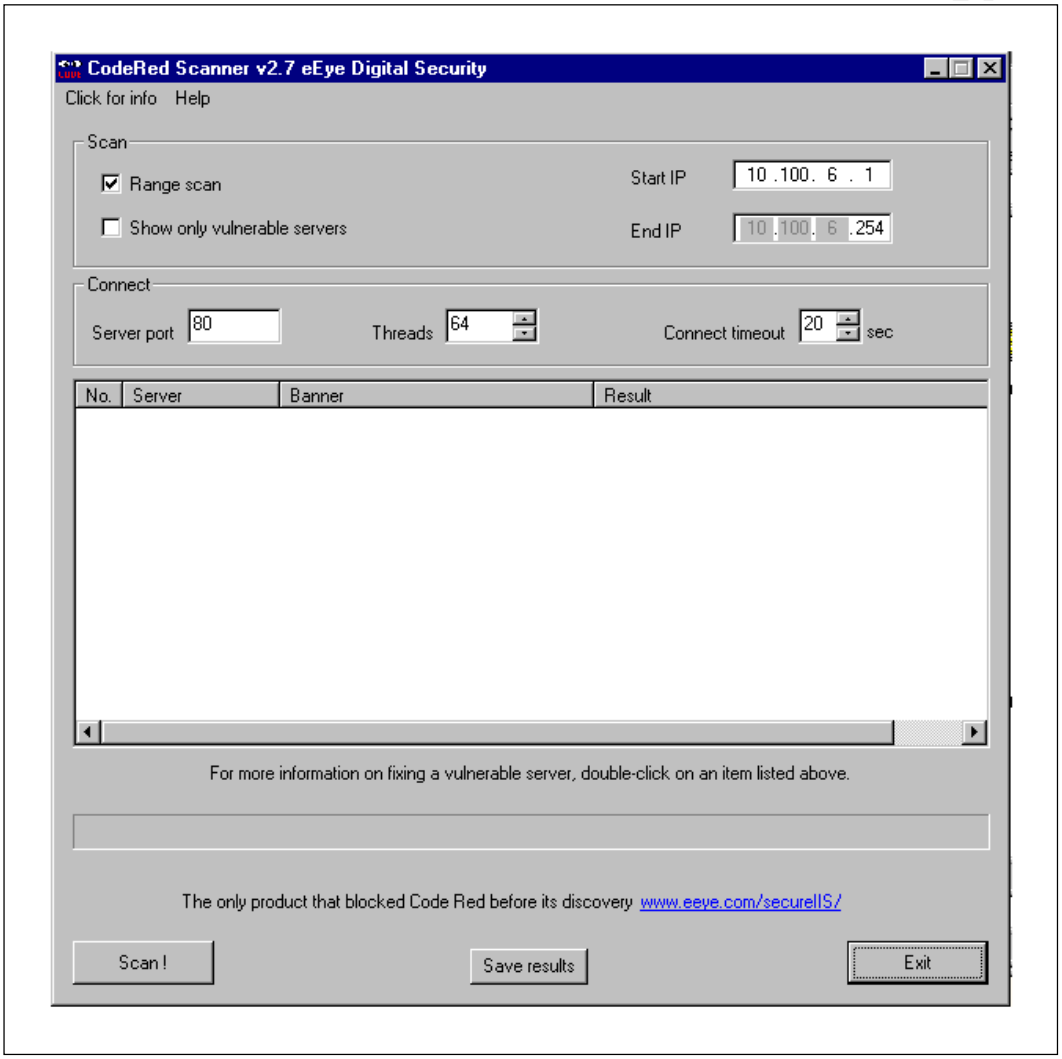
Step for running the cleaner are below:

1. Go to the link above.
2. Download the CodeRedCleanup.exe
3. Select "Save this Program to Disk" to copy the download to your machine.
4. Copy the file to the local directory of any infected systems.
5. Go to the start Menu and choose Run and type "cmd" in the Run field.
6. Go to the CodeRedCleanup location and type the following command: "CodeRedCleanup"
7. To disable IIS on your system type the command: "CodeRedCleanup –disable".

Scanning for Code Red

Several vendors provided tools to allow those with IIS scanners to determine if their

servers were vulnerable to the exploit. Eye.com provided the first scanner to detect the ida/idq vulnerability. The scanner provided was a simple yet pleasant gui interface designed to scan single IP address or an entire blocks. The initial scanner was configured to scan Class C address. Later a second scanner was release for Class B addresses. Below in Figure 11 we see the scanner.



Responsible Coding

To protect against buffer overflows in code a bound check should be required for all code that contains buffers. No doubt this could prove to be an overwhelming task. However a bound checking mechanism based on what type of application may make it applications that are Internet accessible a bit more secure. For instance a rating or weighing applications could be implemented. All applications that are vulnerable to Internet protocols (e.g. IIS, Netmeeting, Proxy Server, ISA Server, etc) can be given a critical rating and aggressive measure taken to insure that improved checking to identify and fix unchecked buffers. Applications which are network aware but localized to LANs could be rating less critical with some of the same

measures but less aggressive. Finally those applications which are network accessible only because they are on devices which are network aware would receive a lower risk rating (e.g. Microsoft Word, no one is going to try to attempt a buffer overflow on word to gain network access).

Source code/ Pseudo code for CRv1 and CRv2

Source code for this was taken from the eEye.com site and can be found in it entirety at http://www.eeye.com/html/advisories/coderedII.zip Break down of the code is as follows:

CODEREF:seg000: :00000000

In order for the CRv1 and CRv2 to work the first requirement is a vulnerable host. This means a Microsoft® Windows NT® 4.0 server or Windows® 2000 server running IIS with the Index Server 2.0 for Microsoft® Windows NT® 4.0 and the Indexing Service for Windows® 2000 unpatched and unprotected through countermeasures. Once a vulnerable host is located an HTTP GET/default.ida command is issued and followed by a buffer overflow.

CODEREF:seg000:000001D6 – CODEREF: seg000:000001FE

This is the portion of the code were the buffer overflow results are reaped by allowing for a new pointer to be inserted and pointed to a new stack and some type of memory allocations take place

CODEREF: seg000:00000203 DataSetup.

This is where the function address table is setup that will be used later by the worm to execute various code.

CODEREF: seg000:00000224 - CODEREF: seg000:000005F9

This part of the code is a subroutine of the above seg000:00000203 and stores the main functionality of the program through the use of to major functions GetProcAddress and LoadLibraryA. We see seg000:000005C6 setting of the thread counts

CODEREF: seg000:000005FE

This is the part of the code does the check to see if the language is English. Depending on the results of the query the code will either proceed to seg000:00000636 or return to seg000:00000512 to move the check threads and begin moving toward attack and infestation of other vulnerable hosts.

CODEREF:seg000:00000636 - seg000:0000064A

This is the code that performs the web page hack against the w3svc.dll

CODEREF: seg000:00000803 - seg000:0000088

This is where the date check is done to prepare for sleep or DOS against

whitehouse.gov.

CODEREF: seg000:0000088C - seg000:00000986

This is the part of the code that causes the infected host to open a socket which queries port 80 of whitehouse.gov. After SYN-ACK-SYN is completed the payload of is sent to DOS whitehouse.gov at IP address 198.137.240.91. In here as well is the sleep function which loops back to the DOS after four and half hours.

Source code/ Pseudo code for Code Red II

Source code for this was taken from the bugtraq archives where it had been submitted by Antony Riley and can be found in it entirety at: http://www.securityfocus.com/ (search on Re: new codered variant (very initial analysis)) Break down of the code is as follows:

Code Red II

Code Red II uses only the buffer overflow from CRv1 and CRv2. Beyond that the code changed to install a backdoor and a trojaned version of explorer.exe. IIS owners who still failed to patch there systems and explicitly Windows 2000 hosts vulnerable to attack would have to not only eradicate the worm but also check for the backdoor and the Trojan which would allow attackers to gain control of the host even after the initial worm was removed and the system patched. The trojan takes advantage of an earlier exploit release in Microsoft Security Bulletin MS00-052.

In CODEREF:seg00000000 - seg000001a0 the initial HTTP GET is initiated followed by a buffer overflow of the default/ida. Seg00000230 is where we see the new attacker has changed the name to Code Red II thus producing a variant.

CODEREF:seg00000820 - seg00000880 of the worm copies %windir%\CMD.EXE to the following locations:

c:\inetpub\scripts\root.exe
c:\progra~1\common~1\system\MSADC\root.exe
d:\inetpub\scripts\root.exe
d:\progra~1\common~1\system\MSADC\root.exe

If the administrator of the web server has not bothered to adjust the \scripts and \MSADC virtual folders against execute permission, an attacker can move a copy of CMD.EXE to these externally accessible locations providing a means for the attacker to execute arbitrary commands on the compromised server. IIS will pass commands to root.exe for execution when the server is presented with a request such as (where ARBITRARY_COMMAND is any command):
http://IpAddress/c/inetpub/scripts/root.exe?/c+ARBITRARY_COMMAND

CODEREF:seg00000bd0 - seg00000c90 of the Code Red II worm creates a trojaned copy of explorer.exe The trojaned explorer.exe will cause IIS will to make the C: and D: root directories accessible to a remote attacker even if the root.exe command

© SANS Institute 2000 - 2005 Author retains full rights.

shell program is removed from the \scripts and \MSADC directories.

The worm places its own copy of explorer.exe at c:\explorer.exe and d:\explorer.exe. By placing the trojaned file in these locations, Windows will find and run the trojan rather than the real explorer.exe because of the way Windows searches for executables by default. This will fail if the system has been patched against the Relative Shell Path vulnerability; if not, then the trojaned explorer.exe will be executed when the next user logs into the system.

CODEREF: seg00000240 - seg00000770 appears to be the main body of the new worm. New date and time references are here as well as the thread creation and infection instructions.

Beyond IIS and on to Cisco

While IIS received most of the attention regarding Code Red, other issues began to arise from infected servers. On July 20, 2001 Cicso Systems released an advisory detailing a number of concerns related to some of their products. These issues were both hardware and software based.

CISCO Software

Cisco Systems has done as many vendors in creating products that are web-based. A number of their software products which were web based an running on IIS server became vulnerable. How? Well we must remember that the ida/idq exploit allows an attacker the ability to run commands according to the privileges granted. If the privileges are that of system the attacker can control those applications which rely upon IIS. Below is a list of software and products vulnerable to directly vulnerable to exploit by the ida/idq vulnerability.

Cisco Unity Server
Cisco uOne
Cisco ICS7750
Cisco Building Broadband Service Manager
IP/VC 3540 Application Server
Cisco Collaboration Server (CCS)
Cisco Dynamic Content Adapter (DCA)
Cisco Media Blender (CMB)
TrailHead (Part of the Web Gateway solution)

Additionally, other products while not directly vulnerable as a result of side-effects may be compromised. The list of products are listed below:

Cisco IP/VC 3510 H.323 Videoconference Multipoint Control Units
Cisco Aironet Wireless products
Cisco CSS 11000 series Content Service Switches
Cisco 600 series of DSL routers that have not been patched for a previously published vulnerability.

Cisco IP Phone 7960, 7940, 7910
CiscoSecure User Changeable Password software
Cisco WebView

CISCO Hardware

Owners of Cisco 600 series DSL routers experienced the most visible problem on Cisco hard. The routers would lockup from Code Red as they attempted to process the HTTP request thereby triggering an unrelated vulnerability that causes the router to stop forwarding packets. To fix this problem QWEST recommended the following steps:

1. Unplug your router from the phone line
2. Do a power cycle
3. Telnet to your router
4. Enter ENABLE mode
5. Type "set web disable"
6. Type "set web port XXXX" (XXXX can be anything in the 1025-9999 range)
7. Type "write" to write the values to nvram
8. Type "reboot"
9. Plug the router to the phone line

Lessons Learned

At the writing of this paper the original Code Red Worm is still active as are variants. It is yet to be seen what else will happen as a result of Code Red. However security professionals and anyone who is as a result of owning and operating a computer becomes a defacto security professional should make the following list to help guard against and at least mitigate malicious attempts to exploit vulnerabilities:

- ✓ Inventory your software and hardware
- ✓ Go to your software/hardware vendors website and sign up for their security bulletins
- ✓ When security bulletins are released apply patches in a timely manner
- ✓ When security bulletins are released which indicated vulnerabilities based on root or administrator patch immediately
- ✓ Get hardening guides for your favorite OS and implement as many of the recommendations as it is reasonable
- ✓ Visit CERT and SANS and get on their mailers
- ✓ If you don't mind being inundated with mail get on a Security Focus mailer.
- ✓ For organizations with application based firewalls, review your rule set and modify accordingly
- ✓ Private individuals should get some type of IDS or Firewall to at least alert them that a threat is present
- ✓ For everyone private or professional, stop accepting installations with default settings

While Code Red was and is still bothersome, we did see the security community pull together and exchange valuable information to help mitigate this particular vulnerability. The numerous bulletins helped those agencies and individuals affected a clear path of recovery.

Links to additional information.

CERT
http://www.cert.org/advisories/CA-2001-13.html
http://www.cert.org/advisories/CA-2001-19.html
http://www.cert.org/advisories/CA-2001-13.html\

Cisco
http://www.cisco.com/warp/public/707/cisco-code-red-worm-pub.shtml

Corecode's Analysis:
http://archives.neohapsis.com/archives/incidents/2001-08/0092.html

CNET
http://news.cnet.com/news/0-1003-201-6658647-0.html?tag=tp_pr

CAIDA
http://www.caida.org/analysis/security/code-red/

Code Red Test Lab Traces
http://www.bofh.sh/CodeRed/index.html

Connected: An Internet Encyclopedia – Third Edition
http://www.freesoft.org/CIE/index.htm

eEye
eEye's Analysis:
http://www.eeye.com/html/Research/Advisories/AL20010717.html

eEye code Analysis
http://www.eeye.com/html/advisories/coderedII.zip

F-Secure
http://www.europe.f-secure.com/v-descs/bady.shtml

Microsoft
Microsoft Security Bulletin
http://www.microsoft.com/technet/treeview/default.asp?URI=/technet/security/bulletin/M

S01-033.asp

Microsoft Follow-up
http://www.microsoft.com/technet/treeview/default.asp?URI=/technet/itsolutions/security/topics/codealrt.asp

Microsoft Security Checklist/Hardening Guides
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/itsolutions/security/tools/tools.asp

NAI's Analysis:
http://vil.nai.com/vil/virusChar.asp?virus_k=99177

Symantec's Analysis:
http://www.sarc.com/avcenter/venc/data/codered.v3.html

The Tao of Windows Buffer Overflow
http://www.cultdeadcow.com/cDc_files/cDc-351/

Security Focus
SecurityFocus Analysis:
http://archives.neohapsis.com/archives/bugtraq/2001-08/0066.html

New Attacks Trend Report
http://www.securityfocus.com/data/staff/Trends.pdf

Top 10 Destination (Attacked Countries) for the Core Red Worm
http://www.securityfocus.com/data/staff/destination.pdf

Average Attacks Based On Averaged Time Of Day (10 days)
http://www.securityfocus.com/data/staff/timeofday.pdf

Average Attacks Based On Averaged Time Of Day (1 day)
http://www.securityfocus.com/data/staff/timeofday-1.pdf

Attacked Industries Report
http://www.securityfocus.com/data/staff/industry.pdf

Targets As Determined By Revenue
http://www.securityfocus.com/data/staff/revenue.pdf

Bibliography

SANS Hacker Exploits Curriculum – "Buffer Overflow Attacks"

SANS Hacker Exploits Curriculum – "Getting Admin"

SANS Hacker Exploits Curriculum – "DOS"