Global Information Assurance Certification Paper

SANS Security Essentials (GSEC) Practical Assignment
Version 1.3

LaBrea – A New Approach To Securing Our Networks

Completion Date: March 7, 2002

Completed By: Leigh Haig

Table of Contents

Summary

This paper has been written to illustrate two of the things that are overlooked with most levels of security implemented to provide a depth of defense: what is happening to the IP addresses on the network that are not being used, and how can these be used to tighten security? If available IP addresses could be used to hold spreading worms or probing systems, then there would be a tangible benefit.

LaBrea, an application to address this concept, will be discussed. As LaBrea was written due to the Code Red outbreak, some insight will be provided into how this worm could defeat existing security mechanisms and what benefit the tool LaBrea could offer. To allow us this insight we will look into the history and technology behind this application, the networking fundamentals that allow it to work, and then study the failings of other defenses within existing security solutions.

Introduction

As security professionals, we are required to ensure that a company's assets are protected from confidentiality, integrity and availability attacks. Deployed tools need to offer a level of defense encompassing both protection from external sources (the Internet) and anything from an internal employee. In history, the primary focus for protection has been to safeguard against Internet launched external attacks. Whilst these attacks are more common, "convention in the security industry has held that 80% of computer-system attacks are perpetrated by insiders".[1] This illustrates the need to ensure that there exists an adequate level of protection against this internal threat – not just focusing solely on the perimeters of our networks. After all, an internal attacker is more aware of the structure and environment, and is therefore better able to target systems and information that can result in a larger loss to the company.

The other key area, and one that makes up for a large portion of the 80% of internal attacks, is viruses and worms. In the 6th annual (2001) Computer Crime and Security Survey[2], 94% of all respondents cited that they had detected computer viruses as a form of attack and abuse within their organization. Based on this, a vital policy is that all networked systems must have anti-viral software installed with the latest definition files as a minimum. However, as proven with both Code Red and Nimda, the time taken for definition files to be released can be too late – an entire company can be overrun with such a fast propagating worm. Other methods are needed that will not only slow this spread, but also identify systems that are infected. Enter the freeware product LaBrea (which was voted Most Useful Application of 2001 by Jim Rapoza in eWEEK's Best and Worst of 2001 column).

The History of LaBrea

[1] http://www.smh.com.au/icon/0103/21/news4.html

[2] The Computer Security Institute (CSI) conducts this survey with assistance from the FBI's Computer Intrusion Squad. CSI press release available at http://usinfo.state.gov/topical/global/ecom/01031301.htm

In the wake of the Code Red worm attack, which saw an estimated 359,000 machines infected in just 14 hours (Code Red v2)[3], one person believed there was something that could be done to fight back against the worm. Tom Liston was frustrated at the large amount of attacks/scanning against his network block - he thought that there may be no way to stop the attacks, but surely the probing could be slowed down. On July 31, with these thoughts in mind, he posted the following message to the INTRUSIONS list at http://www.incidents.org:

> …
> I'm pretty sure that most of you are using your allotted ip address space somewhat like I am. At any given time I'm using only about 20-30% of the ip addresses that I have available. What if I could put something on those other 80% of my ip addresses that would give "Code Red" something to play with that would slow it down to a crawl? A sort of "DoS" back at the worm.
>
> Since those "extra" ip addresses aren't actually expecting any inbound traffic, anything fired at them can safely be assumed to be "bad" traffic. If I wrote a little piece of software that sat on those ip's and listened on port 80, anything that it heard could safely be "played" with.
>
> My hypothetical program should be pretty simple: you see an inbound packet at port 80 with SYN set, and you craft up a return packet with SYN/ACK set and perhaps an option to set the MSS to something small... say about 60 bytes. What does that do? Well, as far as the attacking worm is concerned, after replying with an ACK, it has completed a three-way handshake... it's connected. It's also been told to send information back to you in small chunks (to keep traffic to a minimum), which it dutifully does. The only problem is, my program just answers SYN packets and ignores everything else. So now the worm has to sit around while the whole TCP connection times out. I'm not sure what the timeout NT is, but I think most stacks are pretty persistent about "good" connections, so it should hang it up for a good long time.
> …[4]

This posting was enough to prompt Mihnea Stoenescu to further look into this concept. On August 2, 2001, Mihnea posted the following response to Tom's posting:

> Tom's concept works - I have a living proof.
>
> For a few hours I've been teergrubing CodeReds via three-way handshake on behalf of an entire C-block, by using only one host. At a rate of 6 hosts per minute hitting my block, I'm consuming circa 15 minutes of effective attack time every minute. A lot of hosts can be scanned in 15 minutes.
> …
> Basically, the program sits (unbound) on a firewall blackholing all port 80 traffic

[3] http://www.caida.org/analysis/security/code-red/#crv2
[4] http:// www.incidents.org/archives/intrusions/msg01215.html

> (except legitimate servers, which are only 2 out of 254) and shoots SYN|ACK packets at every illegal port 80 SYN packet.
> …[5]

Encouraged, Tom then proceeded to make changes to Couic[6], the program modified by Mihnea to perform the actions from the above posting. His task was to reduce the programs overall functionality down to simply the catching and holding of Code Red infected machines probing his network block, resulting in the first version of LaBrea, then titled CodeRedneck.

From this original version, which was seen as inferior by its author, modifications were made, including a change of name to LaBrea. Initially, CodeRedneck only worked if the machine it was running on was being probed by Code Red. The primary technology change was that LaBrea could pretend to be systems that did not really exist on the network. These 'virtual machines' could 'become' the IP address of any local system that a Code Red infected machine tried to infect. Essentially, it created a 'tarpit'[7] to hold these attempts of infection spread, whilst logging the machines that it 'tarpitted', thus allowing us to know what machines are infected by the worm. There have been several releases of LaBrea. The current release of LaBrea is 2.4 (in beta mode). As this is the most current version, this paper will look at the features of this release.

How Does LaBrea Work?

LaBrea is a tool that relies on some fundamental principles of TCP networking. To better understand how LaBrea works, we need to understand some fundamental aspects of TCP networking and how LaBrea takes advantage of them to create tarpits and virtual machines.

For TCP communication, each machine has two unique addresses:

- IP Address: a unique 32 bit (for IPv4; 128 bit for IPv6) identifier for each system, which is configured within the operating system (software).
- Media Access Control (MAC) Address: a 48-bit identifier that is unique to every LAN card, as these are 'burnt' into the card at time of manufacture (hardware).

If a machine needs to talk to another within the same Local Area Network (LAN), then the MAC address is sufficient to allow this communication. For communication that needs to span across LAN's, both IP and MAC addresses are required. The IP address is used to route the traffic from the source machine's network, through to the destination. Once the packet has arrived at the final router, the MAC address is then required to complete the communication.

A router is tasked with locating the MAC address that corresponds with the IP address to which

[5] http://www.incidents.org/archives/intrusions/msg01239.html
[6] Couic is a freeware application that can be used to cut off network connections that are unwanted – http://michel.arboi.free.fr/UKUSA/couic.html.
[7] A system is said to be in a 'tarpit' when it has successfully established a TCP connection, but no further communication is being conducted. This session will be held for a finite period of time, preventing 'damage' from being done (either probing, trying to spread an infection, or even network scanning). This has also been referred to as a 'sticky honeypot'.

the packet is to be delivered. To get this MAC address, the router will first check it's own Address Resolution Protocol (ARP) cache – a list of stored IP addresses and the corresponding MAC address for devices on networks that are directly connected to the router. If the IP/MAC combination is not found, the router must ask the network 'who owns' the IP address for which the MAC is being sought. This is done through the use of an ARP request – a broadcast packet that is sent to the LAN. The request is asking which device on the LAN owns the IP address in question. If a machine recognizes its own IP address, it will respond to the ARP, providing the LAN card's MAC address.

For communication that is being done between two machines on the same network, the source machine will perform a similar role to the router. Computers have their own ARP cache – if the IP/MAC entry is not located there, a broadcast ARP request is sent.

With LaBrea, we are trying to stop machines that are probing our networks. In many cases, the IP address for which the router is requesting the MAC address will not exist (as the probing may be done by randomly generating a target IP address, as was done with Code Red). It is through the use of the ARP request that allows LaBrea to 'become' the IP address that the router is searching for.

A router is quite persistent in locating the MAC that corresponds to the IP address. If the router does not get a response to its first broadcast, it will resend multiple times, before giving up and indicating to the requesting machine that the destination is non-existent. It is this resilience of the router in trying multiple times that LaBrea uses. If LaBrea sees multiple requests for the same IP address with no corresponding reply, it will craft a response, inform the router that it is the 'owner' of the IP address and supply back a dummy MAC address. [Note: in a switched environment, LaBrea may not always see the responding ARP reply (the requests will be seen, as these are broadcasts – the replies will be a unicast, sent exclusively to the device making the ARP request). To overcome this, LaBrea has a command line option (-s) which will force the application to issue a 'mirrored ARP request'[8] for any ARP requests it sees.]

The time that LaBrea will wait to send a response to unanswered ARP requests can be configured on the command line, but defaults to three seconds. When it does respond, it will craft a new packet and insert a fake MAC address within, telling the transmitting device how the data can be addressed so it will be successfully delivered. [Note: There is no way to ensure that a switch would know the location of this fake MAC, since the ARP reply would actually contain the real machines MAC within its packet. Under version 2.4, the -s argument offers an extra level to ensure that the switches know the location of the machine with the fake MAC address. The fake MAC is inserted into the source address of the packet also, thus the switch will see that the packet has come from this MAC address. As such, the switch will register this MAC and therefore be able to deliver the packets. According to the application author, there may have only been one reported incident where a switch did not register the MAC address, and therefore the data stream did not reach its intended destination.]

[8] A mirrored ARP request is where LaBrea will send out its own ARP request matching an ARP request that LaBrea sees, with itself as the destination – thus if the machine exists, LaBrea will receive an ARP response.

This fake MAC address will now be recorded against the IP address within the ARP cache. Any traffic destined to the IP will now be sent to this fake MAC address. The LaBrea software will monitor the network looking for any packets that are destined to this MAC. As LaBrea is only a 'virtual machine,' it limits what it looks for – it will watch the network for packets destined to the fake MAC address with the SYN flag set[9] – which is the first step of the TCP Three-Way Handshake. LaBrea will also watch for ICMP echo-request and packets with both the SYN and ACK flags set, and will provide an appropriate response (echo-reply, or a RST respectively) thus making LaBrea appear more realistic to systems that are doing network probes. To better understand how the LaBrea software can 'tarpit' a machine, we should understand the three-way handshake process.

As TCP is a connection-oriented protocol, before any data can be transmitted, a reliable connection between source and destination machine is required. The process of creating this reliable connection is known as the three-way handshake. This is performed with the use of three packets being transmitted: two from the source, and one from the destination. Simply, the three-way handshake has the following steps:

- Source machine will send a frame with the SYN flag set to the destination;
- The destination machine will respond to this frame setting both the ACK and SYN flags;
- A frame with the ACK flag set will now be sent from the source machine in response to the previous SYN/ACK flag.

Within frames one and two, the Maximum Segment Size (MSS) is also specified – this is set by the transmitting machine to inform the receiver of the frame, that this is the maximum size of segments that the machine can receive. A window size[10] is also specified within these frames. When all three steps are complete, a reliable connection has been established – and traffic can now be sent.

It is likely that the data that originally had the router sending its ARP request would have been the SYN frame for a three-way handshake. As LaBrea is written to look for these packets destined to the bogus MAC address, it will reply with a SYN/ACK packet of its own. Having returned its own requirement of the three-way handshake, the source machine has now been 'tarpitted'. As the frames will include the source IP Address (as part of the IP header), this information can now be logged. A list of machines that have been tarpitted is created, and therefore it can be seen what machines are possibly infected with some form of worm or Trojan (dependent on what LaBrea is configured to look for).

With LaBrea sending back the SYN/ACK, with a very small value for the MSS, it is effectively able to tie the offending machine for a period of time, as any data transmitted will be in very small segments. With the original LaBrea, a held thread would eventually time out. On

[9] As LaBrea was originally designed to work against Code Red, it would also only be looking for packets that have the destination TCP port as being set to 80 (HTTP).
[10] The window size is used by a machine to indicate the amount of data (bytes) that it will receive before an acknowledgment must be sent by the destination system.

September 6, 2001 (version 2 of LaBrea), Tom added a 'persist' state (via the command line argument of -p). This would now take an established connection and hold it permanently (or until either the attacking machine, or LaBrea, is shutdown). This is done through the setting of a very small window size during the three-way handshake. When the LaBrea system receives this window size worth of data, it responds back to the source machine with a window size of 0, indicating a wait. The source machine will now enter persist state where it will send 'window probes' to LaBrea at intervals, which decrease in frequency to once every four minutes (for Windows 95 and NT 3.51 stacks[11]). These probes will be answered by LaBrea (with a window size of 0), thus keeping the originating machine in this persist state.

Persist state does carry a traffic overhead – packets will be received by the LaBrea system, and it will have to respond to these ensuring that the connection is held. This overhead is quite minimal (reported to be 1215 bytes/hour by the applications author), and can be seen as an essential requirement in helping to block a machine from performing harm to other networked systems. LaBrea is able to hold many threads – any attacks the machine sees incoming to the subnet on which it sits can be grabbed, and kept in this persist state. Due to the small overhead, the quantity of machines to be held will depend on the bandwidth of the network on which the LaBrea machine resides.

This is only a brief synopsis of how LaBrea is able to offer a level of security within an organization. It has many other features that build upon these principles, but these are out of the scope of this paper. For further information, including a full list of options within LaBrea, visit http://www.hackbusters.net/LaBrea/.

Where Would LaBrea Fit into the Security Model against Code Red?

The need to secure our networks and assets was given more focus in the latter half of 2001, than ever before. The computer industry was rocked by three worms, which proved to be some of the biggest security incidents ever faced. There were two versions of the 'Code Red' worm (Code Red v1 (July 12, 2001) and Code Red v2 (July 19, 2001)) and a different worm, dubbed 'Code RedII' (since this string was contained within the source code), which began infecting machines on August 4, 2001. These were followed on September 18, 2001 by the Nimda worm.

As these worms could all spread on one of the most commonly used ports today (HTTP – TCP port 80), many protective measures in place for security purposes were defeated. As it was the 'Code Red' suite of worms that resulted in LaBrea being created, we will use this to identify where our security failed and emphasise how a tool such as LaBrea would have assisted. But we also need to see why LaBrea in itself, was not the solution to outsmart Code Red.

Most forms of security that are deployed within organizations are used to protect the machines that exist. This makes sense – these are the machines that hold essential data and can readily be compromised should appropriate security measures be lacking. An attacker cannot compromise a

[11] (Win 95) http://support.microsoft.com/default.aspx?scid=kb;EN-US;q154579 and (Win NT3.51) http://support.microsoft.com/default.aspx?scid=kb;EN-US;q153596

system that does not exist – however if traffic is destined to non-existent machines, this could be the sign of an attacker probing the networks, a machine with a planted Trojan or even one that is virus infected. Tools we use to give us this protection are personal firewalls, host based Intrusion Detection Systems (IDS), products combining both firewall and IDS, and anti-virus packages.

By utilizing the above tools, good practices are being implemented to protect the systems that exist and contain company information. Thus ensuring we maintain confidentiality, integrity and availability. However, with Code Red, many of these levels of defense could be defeated due to its spread on a 'well known' and accepted port – port 80. LaBrea would not have assisted with preventing a probe from defeating these security measures. However, through its ability to permanently tie down a TCP connection, it would have limited the number of probes available to attack the network and therefore decrease the number of potential attacks to these systems. Add to this the ability to clearly identify the systems that are probing by logging these systems IP details as they are captured, we are better able to take corrective measures. Such measures could be modifying firewalls to block the traffic, or better yet, identification so remediation of infected systems can be performed.

So why would we need LaBrea to reduce the number of probing threads? Let's look at where conventional security methods would have failed at the height of Code Red. One of the biggest problems that is faced with some of the security tools is that we need to know about the problem or threat before they are useful. The best example of this is with anti-virus packages. Until the vendor is able to provide the definition file that detects the attack, it will be unlikely to offer the required protection. Similarly, some traditional Intrusion Detection Systems will monitor the network for packets that contain special data sequences that map to known attacks. If the attack is new, these traditional IDS's will not suspect the packet of having any malicious intent, and therefore allow it through. With a worm that can spread as quickly as Code Red v2 and the time taken for the worm to be broken down and analysed, tens of thousands of machines can already be infected before the definition files and strings to protect the system are available.

With enough LaBrea tarpits in existence, the spread of the worm would be significantly reduced, due to less threads being able to infect. This then has a flow on affect – fewer probes will result in fewer infections, which will once again reduce the number of attacking threads. However, we would have been in a much better position had all system administrators taken action on the advisory of eEye Digital Security dated June 18, 2001 (http://www.eeye.com/html/Research/Advisories/AD20010618.html). If the vulnerability had been removed on all systems, Code Red (and subsequently Nimda) would have had no victim machines that allowed it to spread!

Any good security model will also incorporate some firewalls. Primarily these firewalls are used to protect the perimeter of our networks. However, software based firewalls are becoming commonplace on critical systems. In fact, all mobile computers that may be used to connect directly to the Internet should be running some form of firewall – systems are protected from the Internet whilst on company networks, so should have the same protection when connected directly to the Internet. The Code Red worm could also defeat the use of a firewall. A common form of firewall is packet filtering, which primarily controls the flow of traffic on the port that the

packet is destined to. As Code Red would attack computers with IIS installed, it is likely that TCP port 80 would have been open through these firewalls and, as such, would not have stopped the spread of infection. If however IIS had been unknowingly installed on a system, then a firewall may not have been configured to allow port 80 through. In this case, the Code Red worm would have been prevented from further spreading the infection.

Using these examples, it is obvious that relying on the 'well known' models of security is not enough. Additional steps of limiting the number of attacks that can be made against legitimate machines needs to be taken. To provide the best protection, it must be through the defense in depth idea – apply layers of security to provide this protection. These layers should include the examples used above, but also ensuring that new tools are constantly being sought to improve the security that is being offered. This will give more time to apply definition files and IDS strings to protect the machines that do exist. This is where we apply LaBrea.

So, is LaBrea a silver bullet? Sadly no. It is yet another important tool that we should ensure exists in our security toolkits – it will add further depth to our security model. It has been documented that LaBrea will hold a thread of the Code Red virus. While this is good news, it does not help with the other threads that Code Red generates. For example with Code RedII, it had the ability to create 600 threads per infected machine (if certain conditions were met). Stopping just one of these threads would leave a further 599 probing the networks for vulnerable systems. If LaBrea is to be completely effective, it needs to be located on all networks, as it is only able to capture the threads that are looking for systems on the network where LaBrea is running. And to be conscientious Internet citizens, we would deploy LaBrea machines outside our firewalls to stop probes and threads out on the Internet also. If enough of these tarpits are configured, then we may just be able to dissuade people from creating viruses and worms, as they would have little to no impact to the Internet or within corporations.

LaBrea is a useful tool that may be able to reduce the number of threads but it would not stop a thread to a machine that is live. We still need to ensure that we are deploying security measures such as anti-virus, IDS and firewalls to protect our live systems and provide the layers essential to 'defense in depth'. If no security is in place to protect these systems and they have not been patched then they will quickly become infected.

Further Uses of LaBrea

While LaBrea could have made a difference during the Code Red debacle of 2001, this is simply one instance where the tool could be put to use. As the spread of the worm was akin to a person performing probing or port scanning on an unknown network, it is logical that LaBrea could also be used in these situations.

If a hacker is preparing to attack an unknown network they may first try to gather intelligence about the network. Such intelligence would be the existence of machines on the network; the types of operating systems running on these machines, etc. To do this, it is likely they will first perform some type of scan of an entire subnet range – a likely scan would be via a ping (ICMP

echo-request packet) to see what machines are answering and are therefore logical targets for an attack. If LaBrea is configured to be monitoring the network for pings to systems that are not online, then these pings can be tarpitted. This would hopefully, at a minimum, slow down the reconnaissance but possibly also thwart the attacker's reconnaissance.

From this one example, it can be seen that if LaBrea is configured and listening as per our requirements, it would be able to tarpit many different types of traffic destined to IP addresses that do not have live owners. Remember, LaBrea is completely software configured, so it can be configured to monitor the network for any types of traffic that it is needed to. The limits of the software are only governed by the limits of our minds to invent new ways of deploying it.

A recent addition to the growing LaBrea family is LaBrea@Home. This has been developed and is available as a precompiled executable for any Windows machine. The purpose of this version of LaBrea is to run on a live machine, and to tarpit threads that are directed to that machine itself. In the previous section we discussed how LaBrea will only tarpit threads to IP addresses that are not in use (as they will not respond to ARP requests, and therefore give LaBrea the chance to 'become' that machine). If we could deploy this version on all networks, and run LaBrea@Home on all other machines, we would have all threads of these worms caught in tarpits. It would of course be most effective on systems that are rarely rebooted – this will ensure that captured threads will remain in the tarpit (recall that the only way a thread caught in persist state can be released is through either a reboot of the infected system or the LaBrea instance being terminated).

Another concept that LaBrea works well in conjunction with is honeypots. A honeypot is essentially a decoy network to would be hackers or cyber terrorists. Each machine that resides within these honeypots is configured with minimal security. However excellent logging and monitoring is setup, to allow every action taken on these systems to be recorded for future analysis. This brings about two advantages: hackers will spend time attacking systems that are of little or no value to them leaving valuable systems alone, and the honeypot administrators are able to see what hackers do, and therefore utilize this information in working out ways to combat the threat they pose.

The honeypot compliments the ability of LaBrea. The two tie up the actions of hackers – honeypots will distract them by having them hack systems that will serve them no benefit, whereas LaBrea will tie up probes from viruses and hackers doing scanning (if LaBrea is configured to do such). If we can get enough of both of these 'distractions' on the Internet, then we could possibly see a reduction in the types of cyber threats faced today. As ISP's are the source for most of our traffic to the Internet, then it would be good to see this as a location to setup LaBrea tarpits. If this could be accomplished then much of the 'dangerous' traffic that security tools are currently required for will be eliminated at the ISP. With a sprinkling of honeypots also on the Internet, we may eventually be able to frustrate the attackers out there into turning their attention to new and less harmful hobbies.

LaBrea is Great – How Can I Get It?

LaBrea has been made available for both Linux and Windows NT (this is a full version, in addition to the LaBrea@Home as detailed above). To download either variant of the traditional version of LaBrea, visit http://www.hackbusters.net/LaBrea. Note: Some network drivers will need to be installed for both the Linux and Windows NT versions, as these are required by the application. Full details on this additional software can be found at the above URL.

If you would like to run LaBrea@Home on your machine, then this can be obtained from http://www.hackbusters.org/LaBrea/lbathome.html. As with other versions of LaBrea, some additional drivers will need to be installed – read the web site carefully for full details.

Conclusion

From past experience, we are seeing that there is no one solution for security. LaBrea is yet another example of this. On it's own it would not be an effective tool in combating attacks such as that from the Code Red series of worms. The concepts that LaBrea utilize will however become yet another important piece in any good security model. It will complement the other methods of defense that we deploy to provide a much better level of protection. The only truly effective way to prevent worms like Code Red is to patch all systems against the vulnerabilities that they exploit. This does not seem to be something that will be achieved in the near future, so we need to deploy other methods within our organistations and on the Internet.

If LaBrea had existed at the outset of Code Red we would have been able to slow the spread of the worm. Many threads would have been caught and placed into the LaBrea tarpit. As this connection would permanently be tied up, it could result in one less infection. From even one less infection, we could have saved another 600 threads becoming active (from Code Red v2). For the next worm or major vulnerability, we will be better prepared now that we have LaBrea in our security toolkit.

References

1. "'LaBrea – The Tarpit' Homepage" http://www.hackbusters.net/LaBrea (12 February, 2002)

2. Liston, Tom. "Welcome to My Tarpit: The Tactical and Strategic Use of LaBrea" http://www.hackbusters.net/LaBrea/LaBrea.txt (12 February, 2002)

3. Pillar, Charles. "Computer crime flourishes behind states of denial." f2) network article. Wednesday March 21, 2001 http://www.smh.com.au/icon/0103/21/news4.html (1 March, 2002)

4. Computer Crime and Security Survey. "Text: FBI-backed Survey Shows Increasing Computer Crime." Released to press on 12 March 2001. http://usinfo.state.gov/topical/global/ecom/01031301.htm (1 March, 2002)

5. "eWEEK Labs Picks Best & Worst of 2001." eWEEK, December 27, 2001. http://www.eweek.com/article/0,3658,s%253D1884%2526a%253D20371,00.asp (2 March, 2002)

6. Arboi, Michael. "A connection cutter." November 9, 2000. http://michel.arboi.free.fr/UKUSA/couic.html (23 February 2002)

8. "CAIDA Analysis of Code-Red. Code-Red Worms: A Global Threat." Cooperative Association for Internet Data Analysis (CAIDA) Web Site, 25 Aug 2001 http://www.caida.org/analysis/security/code-red (25 February, 2002)

9. "TCP/IP Performance Degrades When Resuming Large Data Transfer (Q153596)." Microsoft Product Support Services, first published August 9, 1996. http://support.microsoft.com/default.aspx?scid=kb;EN-US;q153596 (24 February March, 2002)

10. "Slow TCP/IP Performance When Resuming Large Data Transfer (Q154579)." Microsoft Product Support Services, first published August 9, 1996 http://support.microsoft.com/default.aspx?scid=kb;EN-US;q154579 (24 February March, 2002)

11. ".ida 'Code Red' Worm." eEye Digital Security, released July 17, 2001. http://www.eeye.com/html/Research/Advisories/AL20010717.html (25 February, 2002)

12. "All versions of Microsoft Internet Information Services Remote buffer overflow (SYSTEM Level Access)." eEye Digital Security, released June 18, 2001. http://www.eeye.com/html/Research/Advisories/AD20010618.html (25 February 2002)

13. Liston, Tom. “Can we make life difficult for Code Red?” July 31, 2001 http://www.incidents.org/archives/intrusions/msg01215.html (13 February, 2002)

14. Stoenescu, Mihnea. “RE: Can we make life difficult for Code Red?” August 2, 2001 http://www.incidents.org/archives/intrusions/msg01239.html (13 February, 2002)

15. Comer, Douglas E. “Internetworking with TCP/IP. Volume I: Principles, Protocols and Architecture.” New Jersey: Prentice Hall, Inc, 1995. Pages 73 - 81

16. The Honeynet Project. “Know Your Enemy.” Indiana: Addison Wesley, 2002

17. “LaBrea@Home.” http://www.hackbusters.org/LaBrea/lbathome.html (4 March 2002)

18. “Explanation of the Three-Way Handshake via TCP/IP (Q172983).” Microsoft Product Support Services, first published April 2 , 1998 http://support.microsoft.com/default.aspx?scid=kb;EN-US;q172983 (27 February 2002)