Global Information Assurance Certification Paper

GIAC Advanced Incident Handling and Hacker Exploits

GCIH Practical Assignment
Option 1. Illustrate an Incident
Dominick Glavach
July 2001

TABLE OF CONTENTS

Dominick Glavach
GCIH Practical Option 1
Illustrate an Incident

EXECUTIVE SUMMARY

Friday July 20, 2001 at 10:30am, two email messages were received from Minnesota State University and BellSouth Net stating a host on MYNET.COM network was attempting several HTTP connections to a host on their respective networks. The host identified in both emails is a Microsoft Windows 2000 web server used for collaboration and testing on the ACME project. This web server is located in the CLB lab area and connected to a network on the public side of the company firewall. The original intent of the CLB lab was to house two Internet workstations accessible to CLB project staff for the use of collaborative tools not permitted on the company Local Area Network (LAN). During the CLB project life cycle, the original project manager and team members were reassigned and the CLB lab began to house ACME project equipment. The ACME project manager was under the impression that the CLB lab was protected by the company firewall and began to deploy a web server, DNS server, LDAP server, and Microsoft Windows 2000 workstations. As a result, the ACME project equipment was not monitored, audited, or updated.

The activity reported to MYNET.COM and the observed egress traffic from the suspected host was consistent with activities described in the 7/19/2001 SANS News alert. A three member Computer Incident Response Team (CIRT) was assigned to the incident and deployed to the CLB lab. The initial investigation of the ACME project web server the CIRT confirmed that the Code Red Worm had compromised this host. In addition, CIRT found evidence of prior and undetected compromise consistent with the sadmin/IIS Worm. The Code Red Worm and the sadmin/IIS Worm are automated attacks against Microsoft IIS web servers, which leverage documented flaws in the web server software. The worms modify the server's default HTML document and attack other web servers connected to the Internet. The compromises identified and investigated during this incident,

- **Did not** impact data or networks behind the company firewall.
- Both attacks were part of an industry wide attack. The attack was random and does not appear to be a coordinated or targeted attack against MYNET.COM
- The attack was isolated to the single ACME project web server; no other MYNET.COM servers were affected by this compromise.
- The compromised ACME project web server did infect other web servers connected to the Internet.
- The incident was successfully contained and eradicated in a short time frame.
- The CLB lab was brought offline to redesign to suite the needs of the ACME project.

This incident demonstrates the need to revisit company policies and procedures for the development and sustaining of project-based labs. In addition, precautionary measures and security audits should be conducted on existing project-based labs to ensure a similar situation or a larger impact compromise does not occur in the future.

The following sections describe the Computer Incident Response Team's approach to handling the ACME web server incident, incident number 010720-1304-1. The incident response process is divided into six phases: preparation, identification, containment, eradication, recovery, and follow-up.

PHASE 1: PREPARATION.

The intent of the preparation phase is to ensure a solid incident response process is established prior to an incident occurrence. During this phase administrative controls are established or refined for incident reporting, incident response, and system banners. This phase also includes identifying and training staff responsible for incident response, establishing hardware and software associated with incident response, establishing communication plans and developing relationships with law enforcement agencies or other CIRTs.

I attended Baltimore SANS 2001 as part of my company's preparation phase. Currently company's IT security is responsible for the coordination and implementation of protection, detection, and response mechanisms on a network with presence in 13 states across the country. The company is currently modifying IT security related policies and procedures. My responsibility was to refined the overall IT security policy and create specific policy statements and procedures for Incident Reporting, Incident Response, System Security, Internet Access, Identification and Authorization, Data Access Controls, and acceptable use. The policies are intended for and must be acknowledged by all users of company information resources. Contractors and business partners acknowledge a compact version of the company IT security policy. The Incident Reporting and Incident Response policies and procedures define what is considered an activity and an incident, identifies points of contact for incident reporting and incident response, how incidents response activities are communicated, identifies a core Computer Incident Response Team and the six phases of incident response.

The CIRT is comprised of technical and non-technical staff members. The CIRT members have defined roles and communication flows. The following is a summary of each CIRT member's role; my role in the CIRT is the CIRT Lead as well as a member of the primary CIRT member.

Executive Manager Representative decides to involve law enforcement agencies or not and speaks on behalf of MYNET.COM in the event of media involvement.

Human Resources Representative coordinates with supervisors to address misuse by internal staff.

Security Resources Representative works with law enforcement agencies and will coordinate with other security agencies if required via applicable government directives when the Executive Management Representative authorizes their involvement

CIRT Management Representative: Communicates incident status to the Executive management Representative and Human Resources Representative when necessary. The CIRT Management Representative is responsible for making the decision of recovery or investigation.

CIRT Lead: Acts as point of contact for Incident Reports and coordinates Incident Response actions. Provides action recommendations and communicates incident status to the CIRT Management. Provides incident response reports to CIRT Management and maintains record of all incidents. Ensures proper execution of the Incident Response phases during all incident handling.

Primary CIRT: Responsible for the on-site investigation and incident response. The primary CIRT is identified individuals in 4 technical expertise areas including network routing devices, Unix systems, NT systems, Data Backups/Recovery.

Secondary CIRT: Responsible for the support of incident response when necessary. Many of the system administrators are involved in this role. The secondary CIRT members are involved when possible to cross-train and prepare an individual to act as a Primary CIRT member.

Executive management is currently reviewing an updated warning banner. Currently, warning banners are only posted on Unix hosts and a few Microsoft Windows workstations. The existing banner does not issue warning of monitoring or logging and is only acknowledged on Microsoft Windows-based workstations. The updated banner will be posted to all MYNET.COM hosts and will need to be acknowledged before proceeding on to the user identification and authorization process. Appendix A contains the current and updated warning banners.

MYNET.COM has management acceptance for incident response activities. We are in the process of purchasing equipment to establish an isolated test lab. A portion of the lab will be used as a practice field for incident response and computing forensics. We are also in the process of identifying available hardware and software to be used as CIRT jump kits. The jump kits will include the following:

- Incident Response checklist and call list
- Wire-bound note book and pens
- Dual-boot laptop computer with 2 batteries
- 8 port hub
- Two 3 meter Ethernet cables
- One 3 meter crossover Ethernet cable
- One 50 foot Ethernet cable

- Small 5 plug surge protector.
- Floppy disk containing our PGP public keys.
- Formatted floppy disks and new tapes.
- CD-ROMs containing common OS binaries
- CD-ROM containing forensic utilities
- Windows NT resource kit

A disk duplicator and CD Burner are available when necessary.

The primary CIRT members keep current with Internet attack trends and activities through security web site and mailing lists. A well-informed CIRT is able to quickly assess suspicious activities or identify incidents based on current activity. This type of preparation does not substitute for an on-site assessment. This preparation gives the handler a starting point. It is important to point out that this type of preparation can work against the identification phase when a CIRT member determines the incident prior to an on-site visit. Incident handlers must be aware of attack trends and signatures, but should not use attack trends and signatures as the primary identification method. The following are the main resources used to remain current with Internet attack trends.

- SANS Internet Storm Center at www.incidents.org
- Packet Storm Security at www.packetstormsecurity.org
- @stake security news at www.atstake.com/security_news
- CERT Advisories at www.cert.org
- SANS News bytes
- VulnWatch mailing list at www.vulnwatch.org
- Bugtraq mailing list at www.securityfocus.com
- Incidents mailing list at www.securityfocus.com
- Forensics mailing list at www.securityfocus.com

Our goal is to focus a great deal of attention to the preparation phase. We have found that incidents are reported at the most inconvenient of times. A well-prepared CIRT is able to efficiently respond and recover under stressful conditions.

The preparation phase for the Code Red Worm included all of the above information and as well as sending notice to system administrators on the Microsoft IIS patch MS01-33 addressing the Microsoft Index Server security flaw on June 21, 2001. A second notice was sent on July 20. 2001 to system administrators as the Code Red Worm activity began to appear in mailing list postings. All CIRT members were familiar with the Code Red Worm activity. I created a small Perl script, which attempted to retrieve the “default.ida” file on all of the known Microsoft IIS web servers. The Perl script is listing in Appendix B. The results of the script execution indicated that the test systems where either patched on were configured with “*.ida” extensions. We also created custom signatures on the primary Network Intrusion Detection System (NIDS) to trigger an alarm when any host attempted an HTTP or HTTPS connection that contained the string “default.ida”. The NIDS trigger provided the CIRT team with a

threat level and was intended to identify unknown systems vulnerable to the Code Red Worm attack.

PHASE 2: IDENTIFICATION.

The identification phase involves determining whether or not an incident has occurred, and if one has occurred, determining the nature of the incident[1]. This phase also includes the communication of incident status and identifying the necessary staff to recover from the incident.

This incident was reported on July 20,2001 at 11:31am through a PGP encrypted email from the administrative contact listed in the whois information for MYNET.COM. The emails claimed a host in MYNET.COM's address space was attempting numerous unauthorized HTTP connections to a host in Minnesota State University and BellSouth Net's address space. The forwarded email was saved to project space assigned for CIRT activities. I began to investigate the reported activity by creating an entry in my handler's notebook, noting the date, and time, and reported activity.

Since I was unable to recognize the reported IP address as a common MYNET.COM network segment, I contacted another primary CIRT member via phone to help identify the location and purpose. The other CIRT member recognized the network segment and identified the IP address as an active host in the CBL lab. We have now confirmed the reported IP address is part of our network. In order to verify the claims stated in the email, I decided to have the primary CIRT member with router expertise insert a packet filter to log all activity from the suspected host on the router in front of the CBL lab. The packet filter listed below did not interrupt any network traffic with respect to the suspected host. The tracer filter was installed on July 20, 2001 at 11:50am and noted in my handler notebook.

```
access-list 150 permit ip <suspected host> any log
```

The router is configured to log to a centralized log-host. The log output indicated the suspected host was attempting to establish numerous HTTP connections to various hosts outside of MYNET.COM's address space. The log output was saved and copied from the log-host to project space via SSH scp utility. The saved log files were PGP signed with my PGP private key to ensure integrity. We have now verified the activity reported in the emails. The observed activity and current happenings of the Code Red Worm was enough information to an on site team to further investigate the activity.

On July 20, 2001 at 12:17pm a 3 person CIRT was deployed on site to further investigate. The team included myself, the primary CIRT member with NT expertise and another incident handler. Prior to the on-site visit, I notified the CIRT Management Representative (CIRT-MR) via phone of the suspected activities and stated a follow on

[1] The SANS Institute, Computer Security Incident Handling Step By Step, Version 1.5, page 15.

call within an hour with an Identification decision. The initial CIRT-MR notification was noted in the notebook. The time of day enabled the CIRT to keep a low profile and proceed to the lab location without interruptions.

On July 20, 2001 at 12:30pm the NT-based CIRT member and myself began to look over the suspected system. The first action was to inspect the established connections on the host with the "`netstat -an`" command. The output was consistent with the observations made from the router logs. The netstat output (listed in Appendix C) was captured and saved to a floppy disk. The floppy was label with the date and contents. The next action was to search web log for unusual activity. The search identified activity consistent with exploits targeted at vulnerable IIS servers and the Code Red Worm.

The combination of observed log entries on the suspected host and the router logs, a high system load, and numerous entries in the netstat output confirmed that this host had been compromised. On July 20, 2001 at 12:46pm, the CIRT team identified the reported activity as an incident. The incident has been identified and is labeled as incident number 010720-1304-1. I notified the CIRT-MR via phone and began to contain the incident. I do not typically deploy a CIRT during the Identification phase, a designated primary CIRT member or myself normally conduct the Identification phase. Since the host was located in a lab, which, originally was to contain only workstations and now is housing servers operating without a system administrator, I deployed the small team as a precautionary measure. The CIRT deployment is normally done during the containment phase. I have found that early CIRT deployment can be very helpful and a data saving event during situations similar to this incident.

PHASE 3: CONTAINMENT.

The purpose of the containment phase is to limit the incident magnitude and impact from phase forward. CIRT teams must exercise caution during this stage to ensure the their actions do not increase the incident impact or destroy forensic evidence.

On July 20, 2001 at 12:55pm the first order events was to limit the incident scope by stopping the Code Red Worm from infecting other vulnerable servers. The simplest and most effective method to eliminate a network-based attack is to isolate the network or remove the physical network connection from the compromised host. The Code Red Worm analysis, (listed in Appendix D), did not indicate any negative impacts associated with removing network connectivity which made physical network termination a possibility. However, since the compromised host had not been administered or monitored for unauthorized access we could not be certain that the Code Red Worm was the only compromise to this host. Removing the network connection may cause a malicious program to destroy data or other harmful events to the system. For this reason, the network activity will need examined to determine if this threat exists. The other two CIRT members began to setup the 8-port hub and the laptop from the jump kit

as I contacted the CIRT-MR and system owner in order to communicate the status. During my phone conversation with the system owner, the communication between myself and system owner became confused with a conversation with the CIRT members. As a result of the confusion, on July 20, 2001 at 1:02pm, one of the CIRT members removed the physical network cable from the host.

Removing the network connection did prevent the attack from spreading and did not cause catastrophic system failure. However, removing the network connectivity caused a significant increase in system load making the system unresponsive. Due to the lack of system response, system investigation could not be continued until resources became unallocated or the system was reset. The system load was monitored for a ten-minute period and we observed a slight system load decrease. This enabled us, on July 20, 2001 at 1:16pm, to create backup copies of all log files web content. We used the gzip and ftp binaries from the OS CD-ROM from the jump kit to create archive files and transfer the data to the laptop. The compromised system remained off the network and rebooted at 1:34pm. After the reboot, the host did not attempt any connections to external web servers. On July 20,2001 at 1:36pm, Incident 010720-1304-1 was contained.

The system owner was contacted to gain information on the impact to the ACME project web server outage. I described intent of the CBL lab and its exposure to the Internet to the system owner. The system owner was unaware of the network configuration and agreed to move to project web content to an existing web server until the lab is redesigned to meet the needs of the project.

The system owner's decision to use another web server as the host enabled us to down the compromised system, remove the hard drive and create a duplicate drive with the Image MASSter 500 SCSI disk duplicator. The disk duplicator creates a bit by bit copy of an entire hard disk onto another hard disk. A spare 18 GB Seagate disk drive was inserted connected to the disk duplicator target slot. The original 18 GB Seagate was connect to the source slot. The duplicate was created using the "100% copy selection" with error checking enabled. The duplicate disk drive was labeled "Incident 010720-1304-1, evidence duplicate" and sealed in storage. The original drive was reinstalled into the original system.

PHASE 4: ERADICATION.

The eradication phase is used to determine the incident cause and improve defenses to ensure similar incident in nature does not occur in the future.

The log files from the compromised host were examined to determine an estimated compromise time and attack source. The copied log files were examined on the jump kit laptop computer running Slackware Linux. I have found that *nix utilities such as `less cat fgrep awk vi strings` and `od` the most useful and flexible tools for

examining log files. From the log file mining, I made the following observations and notes.

- Web server log files start on 07-05-2001 18:02:16. The system was in operation prior to this date and the start time is odd. It is possible this is when the web server was started for the first time. The possibility also exists of an attacker modifying the log files.
- Found log entries indicating the host was compromised by the sadmin/IIS worm.

```
2001-07-12 16:07:05 Unexpected.Net.241.1 - MYNET.COM.98.5 80 GET
/scripts/../../winnt/system32/cmd.exe
/c+copy+\winnt\system32\cmd.exe+root.exe 502 -
```

- Identified the estimated compromise time and attack source. The compromise time is considered an estimate since the compromised host was not running any type of time synchronization utilities.

```
2001-07-12 16:07:05 Unexpected.Net.241.1 - MYNET.COM.98.5 80 GET
/scripts/../../winnt/system32/cmd.exe
/c+copy+\winnt\system32\cmd.exe+root.exe 502 -
```

The system has been located on the unprotected lab for 30 plus days. During that time frame I identified two log entries consistence with known attack techniques. The compromise system did not have system back ups and appeared to have modified or deleted log files.

The two identified compromises and this incident was eradicated by reformatting the hard drive and reinstalling the OS, web server software and retrieving the copied web content. In addition, the CBL lab is in the redesign planning stage to include the following defense improvement implemented.

- Identify primary and secondary system administrators responsible for the monitoring and updating of ACME project servers.
- Move the lab network segment to the private side of MYCOM.COM's perimeter protection devices and detection devices.
- Implement data backup devices and data backup schedules. The data backups and data restoration should be consistent with MYCOM.COM's backup and recovery policy.
- Install the newly configured web server in MYCOM.COM's demilitarized zone (DMZ.) DMZ hosts are monitored, audited, and updated on a regular scheduled basis.
- Install host-based IDS tools on the ACME project web server

With additional security measures in place and a properly configured and maintained hosts will prevent reoccurrence of this incident.

PHASE 5: RECOVERY.

In the recovery phase, the goal is to validate the system and return to a fully operational state. For this incident the recovery stage will focus on the compromised host. The overall incident recovery includes modifications to the CBL lab, validating the existing servers located in the lab and returning the lab to a fully operational state. The compromised host has reloaded from scratch with the latest patches and host fixes installed. The first step in this process was to perform a random overwrite and low-level format to the hard drive. A system administrator completed this process with my presence. The overwrite process and low-level format is used to ensure the existing data and malicious code is inaccessible. The system administrator proceeded to install the OS, patch levels and modified configurations according to the OS hardening guidelines. Since backups were not performed prior to the incident, the web content was restored from a development area or recreated based on the backup data created during the incident response.

System validation was validated through the use of auditing utilities including ISS Scanner, nmap and Patchworks. The ISS scanner was used to validate web server configuration and identify possible vulnerabilities. The nmap utility was used to validate TCP and UDP services, and Patchworks was used to validate server patch levels. The system was restored to an operational state on Monday July 23, 2001.

PHASE 6: FOLLOW-UP AND LESSONS LEARNED.

The follow-up phase is used to improve the incident response process by identifying areas of improvement and recapping the incident for all CIRT members. The follow-up phase also includes creating Incident Executive summary and recommendations for improving security mechanisms.

This incident was part of an industry wide compromise, which did not affect the company's daily operations or impact any host or data on the private side of the company's firewall. This incident could be been avoided by monitoring and maintaining good system administration practices on the ACME web server.

The lessons learned during this incident response include the value of preparation and the impact communication failures have during the response. Having the available staff, available equipment and the awareness of attack trends enable us to identify contain and eradicate the incident. During the containment phase, the Ethernet cable was removed from the compromised host before the CIRT validated the impact of this action. This was a result of haste and a moment of chaos. I was trying to accomplish too much at one time and had lost the "don't run, walk" approach.

The recommendations resulting from this incident includes a physical inspection of all existing test lab to ensure a similar incident has occurred or will occur in the future, the redesign and redeployment of the CBL test lab, and the creating of policy addressing test lab configurations and staffing requirements. The policy should establish the minimum protection mechanisms necessary to establish a test lab including:

- When possible, test lab should be isolated and not connected to the company LAN.
- Test lab configurations should be documented and equipment located in labs should bear a label identifying then as test equipment.
- Test labs should undergo periodic walk through by security staff.
- Test labs connected to the Internet should have a firewall installed and configured to protect against unauthorized access, IDS devices and staff to monitor and update.
- Test labs should have a backup and restore process.

A follow up meeting was held with the system owner to discuss the redesign of the lab and to provide information on how to avoid future compromises.

CHAIN OF CUSTODY AND EVIDENCE.

All physical evidence collected during this incident was label, sealed in plastic bags and stored in a fireproof safe. As items are entered into the safe, they are added to the safe inventory sheet, which includes date and time of initial storage. The log files and data collected during this incident were stored on tape, labeled and transferred to the fireproof safe. The incident notes were labeled, signed, photocopied, stapled and stored in the fireproof safe. During the incident the laptop and all note remain in my possession until transfer to tape and the safe.

APPENDIX A. LOGIN BANNERS

Current system banner :

```
                        **  WARNING  **

The programs and data held on this system are the property of Company
Name (ABC) and are lawfully available to authorized
users for authorized purposes only. Access to any data or program must
be authorized by ABC.

It is a criminal offense to secure unauthorized access to any program
or data in, or make unauthorized modification to the contents of, this
computer system. Offenders are liable to criminal prosecution.

If you are -not- an authorized user, disconnect -immediately-.

If you have any problems, or need help, please phone:

Technical Support,  555.555.5555
```

Proposed System banner

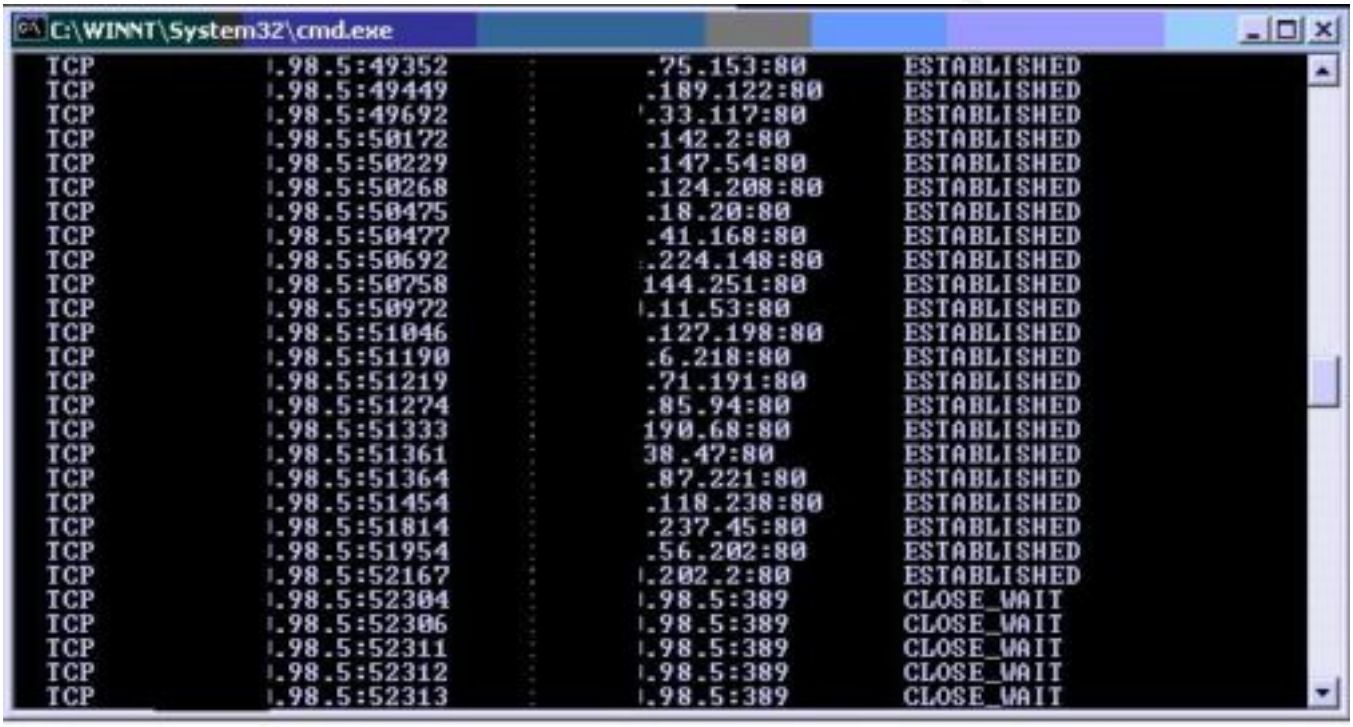
```
                        ** WARNING **

The programs and data held on this system is the property of Concurrent
Technologies Corporation (CTC) and is available to authorized users for
authorized purposes only.

All connections require authorization and are subject to monitoring,
inspection, and investigation. Unauthorized use may subject you to criminal
prosecution. Evidence of unauthorized use collected during monitoring may be
used for administrative, criminal, or adverse action.

All persons are hereby notified that use of this system constitutes consent
to monitoring and auditing.
```

APPENDIX B. DEFAULT.IDA SCANNER

```
#!/usr/bin/perl
## very quick and dirty script to check default.ida existance.
##
## Dominick Glavach <dg@ctc.com>
##
## Usage: crw-test.pl xxx.xxx.xxx.xxx
##
#     #include <stddisclaimer.h>


use Socket;

chomp ($remote = $ARGV[0]);
$port = 80; $vul=0;

$data = "GET /default.ida\r\n\r\n";

$iaddr = inet_aton($remote) || die "$!\n";
$paddr = sockaddr_in($port, $iaddr) || die "$!\n";
$proto = getprotobyname('tcp') || die "$!\n";

socket(SOCK, PF_INET, SOCK_STREAM, $proto) || die "$!\n";
connect(SOCK, $paddr) || die "$!: $remote is not listening or is down. \n";
send(SOCK,$data,0);
while(<SOCK>)
   {
            ## From Roelof Temmingh 2001/05/15 (decodecheck.pl)
     if ($_ =~ /Directory/) { print "** Vulnerable **\n"; $vul=1;break;}
   }
close (SOCK);

if ($vul==0) {print "Looks OK\n"};
# **EOF**
```

APPENDIX C. NETSTAT DATA AND ROUTER LOG

Sample of captured network status from the compromised host. The information listed was obtained from the `netstat –an` command from the Microsoft Windows command line.



Sample router log information collected from the syslog server.

```
Jul 20 11:56:24 [MY.NET.99.9.29.221] 117349: 46w0d: %SEC-6-IPACCESSLOGP: list
121 permitted tcp MY.NET.98.5(21437) -> xxx.xxx.127.254(80), 1 packet
Jul 20 11:56:24 [MY.NET.99.9.29.221] 117350: 46w0d: %SEC-6-IPACCESSLOGRP:
list 120 permitted tcp MY.NET.98.5(21338)-> xxx.xxx.215.254(80), 2 packets
Jul 20 11:56:32 [MY.NET.99.9.29.221] 117351: 46w0d: %SEC-6-IPACCESSLOGRP:
list 120 permitted igmp MY.NET.98.1 -> xxx.xxx.xxx.255, 1 packet
Jul 20 11:56:36 [MY.NET.99.9.29.221] 117353: 46w0d: %SEC-6-IPACCESSLOGP: list
120 permitted tcp MY.NET.98.5(21368) -> xxx.xxx.70.138(80), 1 packet
Jul 20 11:56:36 [MY.NET.99.9.29.221] 117354: 46w0d: %SEC-6-IPACCESSLOGRP:
list 120 permitted igmp MY.NET.98.1 -> xxx.xxx.127.254, 1 packet
Jul 20 11:56:48 [MY.NET.99.9.29.221] 117355: 46w0d: %SEC-6-IPACCESSLOGP: list
121 permitted tcp 194.80.112.17(80) -> MY.NET.98.5(11452), 1 packet
Jul 20 11:57:03 [MY.NET.99.9.29.221] 117356: 46w0d: %SEC-6-IPACCESSLOGDP:
list 121 permitted icmp MY.NET.35.67 -> MY.NET.98.5 (0/0), 69 packets
Jul 20 11:57:03 [MY.NET.99.9.29.221] 117357: 46w0d: %SEC-6-IPACCESSLOGP: list
```

```
120 permitted tcp MY.NET.98.5(21374) -> xxx.xxx.5.78(80), 1 packet
Jul 20 11:57:50 [MY.NET.99.9.29.221] 117358: 46w0d: %SEC-6-IPACCESSLOGP: list
120 permitted tcp MY.NET.98.5(21388) -> xxx.xxx.70.138(80), 1 packet
Jul 20 11:57:50 [MY.NET.99.9.29.221] 117359: 46w0d: %SEC-6-IPACCESSLOGRP:
list 120 permitted igmp MY.NET.98.1 -> xxx.xxx.127.254, 1 packet
```

APPENDIX D. CODE RED WORM ANALYSIS.

Code Red Worm analysis Marc Maiffret < marc@eeye.com >

The following is a detailed analysis of the "Code Red" .ida worm that we reported on July 17th 2001.

This analysis was performed by Ryan Permeh and Marc Maiffret of eEye Digital Security. The disassembly (complete with comments) was done by Ryan "Shellcode Ninja" Permeh.

Table of Contents
==========
1. Introduction
2. Explanation
3. Commentary
4. Deep Analysis
5. Conclusion
6. Appendix
7. Credits

You can get a copy of this analysis, commented disassembly, full IDA database, and binary of worm from
http://www.eeye.com/html/advisories/codered.zip

Introduction
==========
On Friday July 13th we received packet logs and information from 2 network administrators that were experiencing large amounts of attacks targeting the recent .ida vulnerability that eEye Digital Security discovered (http://www.eeye.com/html/Research/Advisories/AD20010618.html) on June 18, 2001. After reviewing the logs sent to us we determined that in fact someone had released a worm into the Internet that was spreading rapidly through IIS web servers.

The full analysis of the .ida "Code Red" worm has provided numerous new details as to the functionality and method of propagation of this worm. For instance this worms purpose ultimately seems to be to perform a denial of service attack against www.whitehouse.gov. Also it has been found that only US English Windows NT/2000 systems will show the defaced ("Hacked by Chinese !") web page.

We've designated this the .ida "Code Red" worm, because part of the worm is designed to deface web pages with the text "Hacked by Chinese" and also because code red mountain dew was the only thing that kept us awake all last night to be able to disassemble this exploit even further.

Explanation
==========
As stated earlier the .ida "Code Red" worm is spreading throughout IIS web servers on the Internet via the .ida buffer overflow attack that was published weeks ago.

Dominick Glavach
GCIH Practical Option 1
Illustrate an Incident

The following are the steps that the worm takes once it has infected a vulnerable web server.

1. Setup initial worm environment on infected system.
2. Setup a 100 threads of the worm
3. The first 99 threads are used to spread the worm (infect other web servers).
	-The worm spreads itself by creating a sequence of random IP addresses.
However, the worm's randomization of IP addresses to attack is not all together random. In fact there seems to be a static seed that the worm uses when generating new IP addresses to try to attack. Therefore every computer infected by this worm is going to go through the same list of random IP addresses to try to infect. The "problem" with that is that the worm is going to end up reinfecting systems and also end up crossing traffic back and forth between hosts to end up creating a denial of service type affect because of the amount of data that will be transferred between all the IP addresses in the sequence of random IP addresses. The worm could have done truly random IP generation and that would have allowed it to infect a lot more systems a lot faster. We are not sure why that was not done but a friend of ours did pose an interesting idea... If the person who wrote this worm owned an IP address that was one of the first hundred or thousand etc... to be scanned then they could setup a sniffer and anytime and IP address tried to connect to port 80 on their IP address they would know that the IP address that connected to them was infected with the worm and they would therefore be able to create a list of the majority of systems that were infected by this worm.
4. The 100th thread checks to see if it is running on a English (US) Windows NT/2000 system.
	-If the infected system is found to be a English (US) system then the worm
will proceed to deface the infected systems website. That means... the local web servers web page will be changed to a message that says Welcome to http://www.worm.com !, Hacked By Chinese!. This hacked web page message will stay "live" on the web server for 10 hours and then disappear and never appear again unless the infected system is re-infected by another host.
	-If the system is not a English (US) Windows NT/2000 system then the 100th
worm thread is also used to infect other systems.
5. Each worm thread checks for c:\notworm
	-If the file c:\notworm is found, the worm goes dormant.
	-If the file is not found then each thread will continue to attempt to infect more systems.
6. Each worm thread will now check the infected computers time.
	-If the time is between 20:00 UTC and 23:59 UTC then the worm will proceed
to use this thread to attack www.whitehouse.gov. The attack consists of the infected system sending 100k bytes of data to port 80 of www.whitehouse.gov therefore potentially performing a denial of service attack against www.whitehouse.gov.
	-If the time is below 20:00 UTC then this worm thread will try to find and
infect new web servers.

In testing we have calculated that the worm can attempt to infect roughly half a million IP addresses a day and that was a ruff estimate made from

using a very slow network.

As of writing this document (July 18 6:49pm) we have had reports from administrators that have been probed by over 12 thousand unique hosts. That basically means at least 12 thousand hosts have been infected by this worm.

In testing we have seen that sometimes the worm does not execute correctly and will continue to spawn new threads until the infected machine crashes and has to be rebooted. We have not been able to isolate the cause of this behavior.

Commentary
==========
As we sat and watched this worm spread itself through the Internet we were somewhat dumbfounded as to the number of reported systems being infected. The main reason for that is that this worm is using a, what we thought to be, very well known vulnerability. A patch for this vulnerability has been available for over a month now and at the time of the release of the vulnerability many news agencies ran stories about it therefore trying to spread awareness. Even with all of that though there have still been, at the very least, 12 thousand web servers infected by this worm. So we once again encourage administrators to pay attention to security issues and patch their systems as soon as patches are available for holes.

All of this research would not be available if we were not a Full Disclosure company and if the many people who helped us research this worm were against full disclosure. Although we did not release any exploit code for the .ida vulnerability, the computer underground still had no problem creating a devastating worm. In the past we have seen a much higher percentage of patched systems when example exploit code was provided. System administrators do not believe vulnerabilities exist without a public exploit, or until something like this worm happens forcing them to patch their systems. We feel that if the security community moves away from Full Disclosure more incidents like this will happen and everyone will be left in the dark as to what is going on.

An example of why full disclosure is so important would be this worm itself. Chances are this worm would have not been discovered if we had not fully disclosed details about the .ida vulnerability, allowing Intrusion Detection System vendors to create signatures for the .ida buffer overflow attack. Because of the way that IIS performs logging (logging a request after processing it), servers hit with this worm will actually have no traces of information left in the IIS log files making tracking this worm (via IIS logs) virtually impossible. The first instances of _anyone_ learning anything useful about this worm was via Intrusion Detection Systems that had signatures based on our .ida vulnerability research which would not have been there if not for full disclosure. The computer underground continues to find vulnerabilities everyday. If full disclosure goes away, then the general security community is left in the dark, and therefore so are administrators.

More and more we are seeing an increase in worms attacking the Internet. There have been about 2 or 3 worms in as many months. Worms force computer security to become a global issue. With the rise of the Internet, not only do you have to worry about securing your own systems, you must also be

careful of your "neighbors" systems. For instance with this worm even if you have done everything right (installed the .ida patch etc...) your network connection could still be taken down because of the amount of data flooding you from all of the "other guys" who hadn't applied patches and were infected by this worm. It is important to not only maintain your own security but make sure that other people you come in contact with secure their sites. Only with global cooperation to maintain security will the threat of worms start disappear.

We received information from "friends" that leads us to believe that this worm is actually a derivative of another worm which was written to exploit the first ever remote IIS buffer overflow (also discovered by eEye Digital Security), the .htr buffer overflow. From reports, the .htr worm had a much smaller infection base and seemed to go under the radar probably because the .htr vulnerability was/is very old.

Deep Analysis

The following is a very detailed analysis of what the worm is doing at each step of its infection. Full disassembled and commented worm code is available at our website at http://www.eeye.com/html/advisories/codered.zip. It will be provided as both a assembly text dump and as a IDA (Interactive Disassembler) database file. We have chosen not distribute the worm code in this eMail as to not clog up your bandwidth anymore then we already are >:-]

Note: We will use CODEREF comments in the analysis to reference where we are in the disassembled code so you can "follow along" with us.

The tools that we used to perform this detailed analysis were:
IDA (Interactive Disassembler) from www.datarescue.com. IDA is an advanced disassembler that made this analysis possible.
MS VC++ debugging environment. We used this to monitor modified pieces of the worm as they interacted with IIS.
Ryan's brain + Marc's brain + private eEye tools. No... you cant have any of those.

We will be heavily referencing the disassembled worm code in the following analysis.

In an attempt to make this easier to understand we have broken the functionality of the worm into 3 parts. The core worm functionality, the worm hack web page functionality and the attack www.whitehouse.gov functionality.

Core worm functionality

1. Initial infection vector (i.e. host is vulnerable to the .ida attack and gets hit with this worm).

The initial infection starts to take place when a web server, vulnerable to the .ida attack, is hit with a HTTP get request that contains the necessary code to exploit the .ida attack and uses this worm as its payload.

At the time of the .ida overflow a systems stack memory will look like the

following:

```
        <MORE 4E 00>
        4E 00 4E 00 4E 00 4E 00
        4E 00 4E 00 4E 00 4E 00
        4E 00 4E 00 4E 00 4E 00
        92 90 58 68 4E 00 4E 00
        4E 00 4E 00 4E 00 4E 00
        FA 00 00 00 90 90 58 68
        D3 CB 01 78 90 90 58 68
        D3 CB 01 78 90 90 58 68
        D3 CB 01 78 90 90 90 90
        90 81 C3 00 03 00 00 8B
        1B 53 FF 53 78
```

EIP is overwritten with 0x7801CBD3 which an address within msvcrt.dll. The code at 0x7801CBD3 disassembles to:

```
        call ebx
```

When EIP is overwritten with call ebx it then causes program flow to divert back to the stack. The code on the stack jumps into the worm code that's held in the body of the initial HTTP request.

2. Sets up some initial stack variables
CODEREF: seg000:000001D6 WORM

At this point we are executing the initial code of the worm. The first thing to happen is that the worm sets up a new stack for its own use. The new stack is 218h bytes, filled with CCh. The worm code then moves on to initialize its function jump table.

The entire worm heavily uses an EBP stack based memory offset system. This means that all variables are referenced as EBP-X values. On our website we have a document called worm-ebp.txt that attempts to track stack usage throughout the course of the worm code.

3. Load functions (create the "jump table")
CODEREF: seg000:00000203 DataSetup

The first thing the worm code does is reference the data portion of the exploit code at EBP-198h. The worm then needs to setup its internal function jump table. A function jump table is a stack based table used to store function addresses. It allows the worm to generate the function addresses at run time (This makes the worm have a better chance of executing cleanly on more systems).

The technique used by this worm is what is called an RVA (Relative Virtual Addresses) lookup. Basically this means that all functions, or specifically GetProcAddress, are found within IIS itself. For more details on RVA please consult any good PE (Portable Executable, the executable file format for Microsoft platforms) documentation, or read through the assembly code of this worm.

In a nutshell, RVA techniques are used to get the address of GetProcAddress. GetProcAddress is then used to get the address of LoadLibraryA. Between these two functions all other functions that the worm may need can be easily found. The worm uses these two functions to load the following functions:

```
>From kernel32.dll:
GetSystemTime
CreateThread
CreateFileA
Sleep
GetSystemDefaultLangID
VirtualProtect

>From infocomm.dll:
TcpSockSend

>From WS2_32.dll:
socket
connect
send
recv
closesocket
```

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

4. Check the number of threads the worm has created.
CODEREF: seg000:00000512 FUNC_LOAD_DONE

Here the worm seems to perform a WriteClient (Part of the ISAPI Extension API), sending "GET" back to the attacking worm. This possibly could be a way of telling attacking worms that they have successfully infected a new host.

Next the worm code will count the number of worm threads already in action. If the number of threads is 100 then control is shifted to the Worm hack web page functionality.

If the number of threads is below 100 then the worm creates a new thread. Each new thread is an exact replica of the worm (Using the same code base).

The worm now continues its path of execution.

6. Checks for the existence of c:\notworm
CODEREF: seg000:0000079D DO_THE_WORK

There seems to be a to be built in "lysine deficiency" (See Jurassic Park, or Caesar's paper on this at www.rootkit.com). A "lysine deficiency" is a built in check to keep malicious code from spreading further.

In this case the "lysine deficiency" is a check for the existence of the file c:\notworm. If this file exists then the worm will become dormant. This means it will not attempt to make connections out to other IP addresses to try to infect.

If this file does not exist then the worm continues onto the next step.

7. Check the infected systems time (computer clock)
CODEREF: seg000:00000803 NOTWORM_NO

The worm will now check the infected systems local time (in UTC). If the hour is greater then 20:00 UTC then the worm will proceed to goto the first

step of the attack www.whitehouse.gov functionality.

If the time is less than 20:00 UTC then the worm will attempt to continue to try to infect new systems.

8. Infect a new host (send .ida worm to a "random" IP address on port 80).

At this point the worm will resend itself to any IP addresses which it can connect to port 80 on. It uses multiple send()'s so packet traffic may be broken up. On a successful completion of send, it closes the socket and goes to step 6... therefore repeating this loop infinitely.

Worm hack webpage functionality

This functionality is called after a hundred threads are spawned within the worm.

1. Check if local system default language is English us then goto step 6 of core worm functionality.
CODEREF: seg000:000005FE TOO_MANY_THREADS

The first thing the worm does is get the local codepage. A codepage specifies the local operating system language (I.E. English (US), Chinese, German etc...). It then compares the local codepage against 0x409. 0x409 is the codepage for English (US) systems. If the infected system is an English (US) system then the worm will proceed to deface the local systems webpage. If the local codepage is not English (US) then this worm thread will goto step 6 of core worm functionality.

2. Sleep for 2 hours.
CODEREF: seg000:00000636 IS_AMERICAN

This worm thread now sleeps for 2 hours. We anticipate that this is to allow the other worm threads to attempt to spread the infection before making a presence known via defacing the infected systems webpage.

3. Attempt to modify infected systems webpages in memory.
CODEREF: seg000:0000064F HACK_PAGE

This worm uses an interesting technique called "hooking" to effectively deface (alter) an infected systems webpages. Hooking is modifying code in memory to point to code that the worm provides. In this case the worm is modifying w3svc.dll to change the normal operation of a function called TcpSockSend. TcpSockSend is what w3svc.dll (IIS core engine) uses to send information back to the client. By modifying this, the worm is able to change data being written back to clients who request web pages of an infected server.

To perform hooking, first the worm makes the first 4000h bytes of w3svc.dll's memory writable. In a normal situation the memory for w3svc.dll (and basically all mapped dll's) is read-only. It uses the function VirtualProtect to change the memory of w3svc.dll to be writable, saving the old state to a stack variable.

It then uses the saved codebase of w3svc.dll (from step 3 of core worm

functionality) as a start point to search the import table (again see PE header documentation) for the address of TcpSockSend. Once the address for TcpSockSend is located the worm then replaces TcpSockSend's actual address with an address within the worm.

The address that TcpSockSend now points to is a function within the worm that will return the "Hacked by Chinese !" webpage. The CODEREF for this function is seg000:00000C9A FAKE_TCPSOCKSEND.

This thread of the worm now sleeps for 10 hours. During this 10 hours all web requests to the infected server will return the "Hacked by chinese !" webpage.

After the 10 hours is up this thread will return w3svc.dll to its original state, including re-protecting memory.

Execution after this proceeds to step 6 of the core worm functionality.

Attack www.whitehouse.gov functionality

Sooner or later every thread within the worm seems to shift its attacking focus to www.whitehouse.gov.

1. create socket and connect to www.whitehouse.gov on port 80 and send 100k byes of data
CODEREF: seg000:000008AD WHITEHOUSE_SOCKET_SETUP

Initially the worm will create a socket and connect to 198.137.240.91 (www.whitehouse.gov/www1.whitehouse.gov) on port 80.

CODEREF: seg000:0000092F WHITEHOUSE_SOCKET_SEND
If this connection is made then the worm will create a loop that performs 18000h single byte send()'s to www.whitehouse.gov.

CODEREF: seg000:00000972 WHITEHOUSE_SLEEP_LOOP
After 18000h send()'s the worm will sleep for about 4 and a half hours. It will then repeat the attack against www.whitehouse.gov (goto step one of Attack www.whitehouse.gov functionality).

Appendix
========
This is associated information about the "Code Red" worm including how to stop the worm, commentary on the worm, and dispelling common misconceptions about this worm.

How to secure your system from this .ida "Code Red" worm?

Microsoft patch for this .ida vulnerability
http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-033.asp
The worm spreads itself to new vulnerable systems via the .ida vulnerability. Applying this patch will keep your server from being infected. However, as stated earlier, because of the way the worm creates

its list of "random" IP addresses to attack, you could still be affected by a high traffic overload denial of service.

eYe Digital Security Advisory for .ida vulnerability
http://www.eeye.com/html/Research/Advisories/AD20010618.html
We initially discovered the .ida vulnerability which is being used by this worm as its infection vector. The above advisory details our research of that specific vulnerability. We worked with Microsoft to help them create a patch for the .ida vulnerability.

SecureIIS - Application firewall, stops known and unknown IIS vulnerabilities.
http://wwww.eeye.com/secureiis
We do produce a product that protects IIS web servers from attack which is one of the reasons that we were so quick to research this worm. Funny enough in our initial testing we couldn't get the worm to work because we forgot we had SecureIIS enabled on the lab web server. heh.

I have been infected by this worm what can I do?

The first thing you must do is goto the Microsoft security site, as referenced above, and install the .ida patch ASAP. The worm will remain in memory until you reboot your server so make sure to reboot after installing the .ida patch.

I think I am infected, how can I tell?

An infected system will show an increase in load (processor/network). It will also show a number of external connections (or attempts) to port 80 of random IP addresses. You can see this by doing a "netstat -an" from a MS-DOS prompt. Either way do not take any chances... if your system is missing the .ida patch then install it ASAP and reboot.

How to setup your IDS to detect this specific worm?

The following is part of the packet data that is sent for this .ida "Code Red" worm attack:

```
GET
/default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNN%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%
u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a
HTTP/1.0
```

Just add that to your IDS signature database.

What are some common misconceptions about the "Code Red" worm?
----

1. It connects to worm.com.
 This worm only specifies www.worm.com in the initial HTTP GET request HOST:

header and in the defaced page show on English (US) systems. This worm does _not_ connect to www.worm.com. This worm operates completely independent and can spread and infect systems without having a single point of failure. What that means is that this worm will be wild on the Internet until there is a _VERY_ high degree of systems that go and install the .ida patch.

2. This worm is based off of hsj's "proof of concept" .ida exploit.
This worm is _NOT_ based off of hsj's "proof of concept" .ida exploit. His
exploit code had no worm functionality. It was a simple exploit shell that had little to no implicit functionality. It was designed to prove to administrators the seriousness of this vulnerability so that they would install patches ASAP.

Credits
=======
Ken Eichman of Chemical Abstracts Service
Matthew Asham of Left Coast Systems Corp
and a large handful of administrators who gave us much needed data to piece this together.

Signed,
eEye Digital Security
T.949.349.9062
F.949.349.9538
http://eEye.com/Retina - Network Security Scanner
http://eEye.com/Iris - Network Traffic Analyzer
http://eEye.com/SecureIIS - Stop known and unknown IIS vulnerabilities

"Its not a virus! Its a worm!" - z3r0 c00l
"Whats this one eat?" - l0rd n1k0n
"th1s 0n3 34ts 11S s3rv3rs!" - ch4m3l30n
h4ck3rs

APPENDIX E. REFERENCES

SANS Computer Security Incident Handling Step by Step Version 1.5 of May 1998

http://www.cisecurity.org/patchwork.html

http://www.microsoft.com/security

http://www.cert.org

http://www.eeye.com

http://www.incidents.org

http://www.fish.com/forensics

http://www.securityfocus.com/archive/1/197828

http://www.whitehats.com

http://www.packetstormsecurity.com