A System of Persistent Baseline Automated Vulnerability Scanning and Response in a Distributed University Environment

GCIA Gold Certification

Author: Chet Langin, clangin@poofaccess.com

Adviser: Dominicus Adriyanto

Last Update: September 9, 2007

Table of Contents

1. Introduction

This paper describes and analyzes a persistent automated baseline vulnerability scanning procedure in a university ("The University"), including preparation, response, and follow up procedures. A Ruby script called run_nxscan.rb, written by the author, runs the nxscan* scanning tool (York University, 2007) and processes the output. But this paper is more about the overall system used than just about a script and the nxscan tool. See Wack, Tracy, and Souppaya (2003) for a list of some other vulnerability scanning tools.

1.1. The Size of the Problem

Practical experience shows that many vulnerable computers quickly get infected. Some statistics follow, along with an idea of the cost of recovering from infected vulnerable computers.

Houghton (2003) states that Antivirus company Symantec documented 2,524 vulnerabilities in 2002, and that the Sapphire Worm (also known as the SQL

* The nxscan program is unsupported and there is no guarantee of its performance or accuracy.

Slammer) doubled in size every 8.5 seconds and reached its peak in 3 minutes, at which time it was conducting 55 million scans per second to seek further victims.

Dacey (2003) states that over 11,000 security vulnerabilities were reported in software products from 1995 to 2003; the Morris Worm brought to a halt 10 percent of the systems connected to the Internet in November 1988; Code Red infected 20,000 systems in 10 minutes; Sapphire successfully attacked at least 75,000 systems; and, the Blaster Worm (also known as Lovsan) on August 11, 2003, infected more than 120,000 unpatched computers in 36 hours.

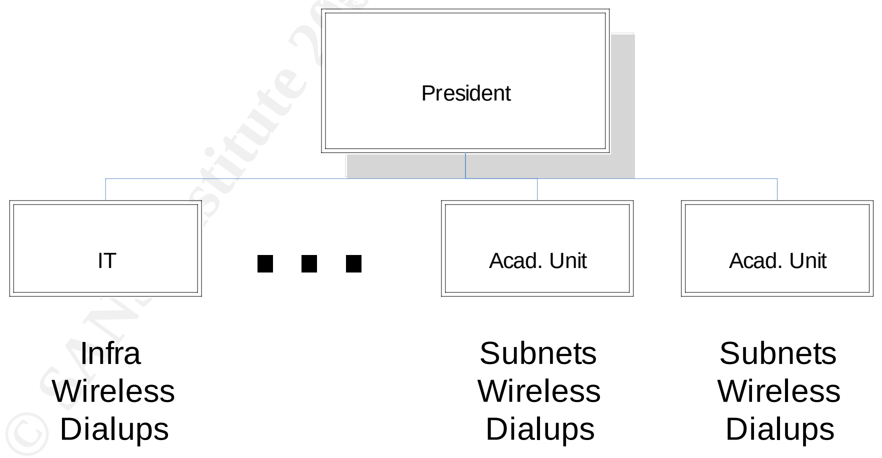
Dodge (2007) reports that 29 educational institutions reported 33 penetration incidents involving 2,209,237 sensitive records in 2006 (not including stolen laptops), and that penetration was the most common type of information security incident reported by these educational institutions.

Mell, Bergeron, and Henning (2005) have figured that the cost to recover from an attack is W * T * R, where W is the number of Workstations, T is the Time spent fixing one, and R is the hourly Rate of pay for someone to fix one. For example, if an organization has 1,000 computers to be fixed, each computer taking an average of 8 hours of downtime (4 hours for one person to rebuild plus 4 hours the user is unable to work), and the rate of pay is $70 per hour, then the cost is 1,000 * 8 * 70 = $560,000 cost to recover from the attack.

1.2. The Setting

The setting is a large distributed university with an estimated 25,000 users and more than 100 Local Area Network (LAN) administrators on a Class B network. The LAN administrators report to their respective schools and departments (i.e., “academic units”) instead of reporting to an existing Information Technology (IT) Department. The University has a diverse network with numerous subnets, dialup access points, and wireless access points, and a dormitory subnet which is a part of the campus area network.

The abilities of the LAN administrators in the academic units range from professional teams of LAN administrators to professors and clerical workers who also work part time in the roles of LAN administrators.



The above illustration demonstrates what is meant by a *distributed environment*. The IT Department provides infrastructure and university-wide wireless and dialup access. However, numerous other academic units, such as schools, colleges, and departments, provide their own LAN administrators with subnets that can include their own wireless and dialup accesses. No formal means of supervisory control exists from IT to the academic units except via the President of The University. The only direct control that IT has over the academic units is to disable network access via the infrastructure. The exact means of disabling access varies and can include disabling a switch port, using an Access Control List (ACL), disabling IT-controlled dialup or wireless access, or otherwise locking out network accounts. Persuasion is an indirect and limited means of control which IT has with the LAN administrators.

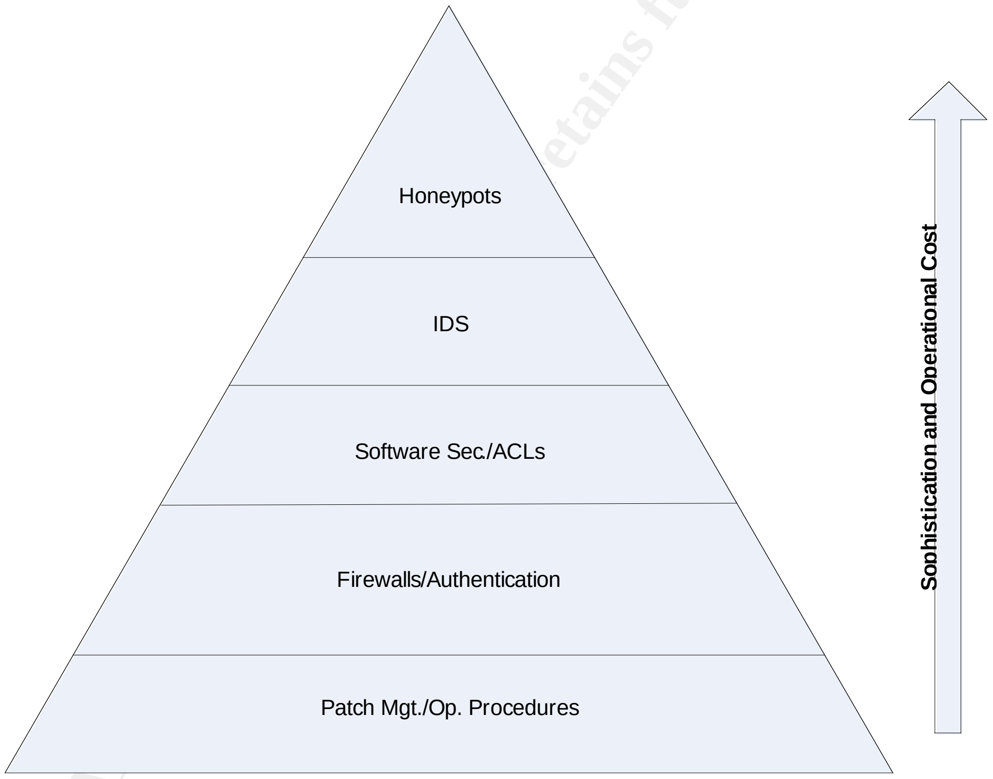
Rogue computers (Boyce, 2001) also characterize a distributed university environment. These include the computers of students who buy outdated equipment, professors that bring personal laptops in from home, and various types of university guests who need temporary network access.

The distributed network at The University necessitates that the Computer Security Incident Response Team (CSIRT) is the functioning Vulnerability Assessment (VA) group. A Patch and Vulnerability Group (PVG) does not exist as such at the university level. CSIRT finds the vulnerable computers and disables network access. LAN administrators and/or users must do the patching, themselves, although a help desk is available for assistance. Ideally, each academic unit would have its own Patch and Vulnerability Group (PVG). But, in reality, many academic units have relied on the CSIRT to find the vulnerable computers.

2. Background

The University has done nearly continuous automated vulnerability scans for over 18 months, repeatedly scanning every possible IP address on the network.

Potter's Pyramid of IT Security Needs


Honeypots
IDS
Software Sec./ACLs
Firewalls/Authentication
Patch Mgt./Op. Procedures
Sophistication and Operational Cost

Potter (2007) states that vulnerability scanning, as a part of patch management, is at the base of Potter's Pyramid of IT Security Needs, indicating that this is relatively low on the security scale in terms of sophistication and operational cost. Thus, this should be a basic part of security management at any facility.

Two programs provided by York University (2007) have been used at The University to accomplish the persistent automated vulnerability scanning: noxscan and nxscan. Other vulnerability assessment tools have also been used to supplement the above two tools. The noxscan and nxscan programs have been very fast, with each scan of The University's Class B network taking approximately 10 minutes. These programs were used instead of Nessus (Tenable Network Security, 2007) because they were faster and easier to set up. I have not closely examined the noxscan/nxscan code and I do not know exactly how noxscan/nxscan works, but noxscan/nxscan has been very accurate to the extent that The University has had no known false positives. The noxscan program was used at first and nxscan replaced it approximately one year later.

The noxscan program scans for vulnerabilities reported in Microsoft® Security Bulletin MS04-007 (2007) describing Microsoft Security Update KB828028 (2007) involving MITRE CVE-2003-0818 (2007); Microsoft Security Bulletin MS04-011 (2007) describing Microsoft Security Update KB835732 (2007) involving MITRE CVE-2003-0533 (2007); and, Microsoft Security Bulletin MS05-039 (2007) describing Microsoft Security Update KB899588 (2007) involving MITRE CVE-2005-1983 (2007).

The nxscan program added scanning for a vulnerability reported in Microsoft Bulletin MS06-040 (2007) describing Microsoft Security Update KB921883 (2007) involving MITRE CVE-2006-3439 (2007). The nxscan program also dropped the scanning for the vulnerability reported in MS05-039.

Network Access Control (NAC) for some parts of campus also contributes significantly to keeping vulnerable computers off of the network. NAC in some areas along with nxscan campus wide provide a baseline for The University for what types of vulnerabilities are never allowed on campus.

Below is sample output from an nxscan—note that “y’s” have replaced the actual subnet address and the prompt has purposely been obfuscated:

```
        [xxxxx@zzzzzzz /yourpath/nxscan]# ./nxscan yyy.yyy.100.0/25
        yyy.yyy.100.71 128 NO CONNECTION
        yyy.yyy.100.126 128 NO CONNECTION
        yyy.yyy.100.110 128 NO CONNECTION
        yyy.yyy.100.62 128 NO CONNECTION
        yyy.yyy.100.104 128 NO CONNECTION
        yyy.yyy.100.69 0 OS[Windows 5.1] MS04-007 SECURE, MS04-011 SECURE, MS06-040 SECURE
        yyy.yyy.100.67 0 OS[Windows 5.1] MS04-007 SECURE, MS04-011 SECURE, MS06-040 SECURE
        yyy.yyy.100.78 0 OS[Windows 5.1] MS04-007 SECURE, MS04-011 SECURE, MS06-040 SECURE
        yyy.yyy.100.72 0 OS[Windows 5.1] MS04-007 SECURE, MS04-011 SECURE, MS06-040 SECURE
        yyy.yyy.100.3 0 OS[Windows Server 2003 3790 Service Pack 1] MS04-007 SECURE, MS04-011
SECURE, MS06-040 VULNERABLE
        yyy.yyy.100.94 0 OS[Windows 5.1] MS04-007 SECURE, MS04-011 SECURE, MS06-040 SECURE
        yyy.yyy.100.70 0 OS[Windows 5.1] MS04-007 SECURE, MS04-011 SECURE, MS06-040 SECURE
        yyy.yyy.100.74 128 NO CONNECTION
        yyy.yyy.100.87 128 NO CONNECTION
        yyy.yyy.100.41 128 NO CONNECTION
        yyy.yyy.100.39 128 NO CONNECTION
        yyy.yyy.100.54 128 NO CONNECTION
        yyy.yyy.100.44 128 NO CONNECTION
        yyy.yyy.100.52 128 NO CONNECTION
        yyy.yyy.100.13 128 NO CONNECTION
        ...
```

An entire Class B network can be scanned at once (nxscan yyy.yyy.0.0/16), however, the output would include over 65,000 lines. Even so, one can filter this output with grep or with script like shown in Appendix B. An entire Class B network scan takes about 10 minutes.

The output above shows one computer at yyy.yyy.100.3 which is exposed to the vulnerability reported in MS06-040. CSIRT would take action to disable network access for the computer at this IP address.

Seven IP addresses are visible to nxscan as shown in the above output, the vulnerable one plus six more. Two operating systems have been recognized by nxscan in the above output, six computers running Windows 5.1 (Windows XP), and the vulnerable computer running Windows Server 2003 3790 Service Pack 1.

"NO CONNECTION" in the above output might mean that the connection is firewalled, the associated computer is turned off or offline, or that the IP address is not assigned—these are ignored.

Sometimes an "INCONCLUSIVE" designation is shown for a bulletin rather than "SECURE" or "VULNERABLE". The University CSIRT does not take action on "INCONCLUSIVE" IP addresses.

2.1. Rational – Broken Windows Policing

Broken Windows Policing is a controversial police practice that predates Microsoft Windows®. In this paper, it is more than a pun: It describes the baseline which noxscan/nxscan scanning provides. Broken Windows Policing (BWP) is generally attributed to James Q. Wilson and George L. Kelling in the article *Broken*

Windows in The Atlantic Monthly magazine (1982): "Social psychologists and police officers tend to agree that if a window in a building is broken *and is left unrepaired*, all the rest of the windows will soon be broken….Untended property becomes fair game for people out for fun or plunder."

The controversial part of BWP has to do with false security. Just fixing broken windows does not necessarily reduce other problems: It is necessary, but not sufficient. The procedure in this paper uses nxscan as a baseline for what vulnerabilities should never be allowed on the campus network. However, other types of vulnerability scanning should also be incorporated into the general process covered by this paper. There is already considerable literature available on other types of possible scanning which can be done.

According to CERT/CC (Dacey, 2003), about 95 percent of all network intrusions could be avoided by keeping systems up to date with appropriate patches—i.e., fixing the *broken windows* with baseline vulnerability scanning and follow up.

2.2. Three Levels of Vulnerability Detection

The level of vulnerability detection is defined in this paper as it relates to the amount of intrusion into a user's computer. There are three levels:

1. Version Checking. This implies no intrusion, whatsoever. The user and/or LAN administrator who have authorized access to the system can check to see what version of a program is installed, what services packs may be installed, and what patches are done and/or needed.

Individual users and LAN administrators should be doing this, but it is not yet practical for the IT department in a distributed university environment to do this for the entire university. Wack, Tracy, and Souppaya (2003) call this *host-based scanning*.

2. Vulnerability Scanning. This involves someone such as CSIRT using their own computers and the network to *look at* other computers, so to speak, to see if they are vulnerable. The CSIRT staff are typically not authorized users of the computers that they are scanning. The distinguishing characteristic of a scan is that the scan is not dangerous to the system that is being looked at. Wack, Tracy, and Souppaya (2003) call this *network-based scanning*.
3. Penetration Testing. This is involves someone such as CSIRT attempting to gain unauthorized access to a computer. The distinguishing characteristic of a penetration test is that, according to Wack, Tracy, and Souppaya (2003), it might damage the system that is being tested--including freezing the computer or causing the computer to reboot, as well as other things.

Deraison, et al, (2004) refers to the last two levels of detection as being non-intrusive and intrusive scanning, also referring to this as *exploiting*, and gives an example of each.

Vulnerability scanning is the level of detection being done in the processes referred to in this paper. It is assumed in this paper that CSIRT already has the necessary permission to do vulnerability scanning.

3. The Process

The detection of a vulnerable computer is considered to be an incident and action is taken. This section describes how scripts process the output of the vulnerability scans; how the users and LAN administrators are contacted; how network access is disabled, how information is documented; how the vulnerabilities are patched; and, how network access is restored.

3.1. Suggested Processes

Eight papers, enumerated below, have suggested incident response processes.

Williams (2003), for example, discusses vulnerability management of remote procedure calls in the *House of Windows*.

The other seven papers offer step-by-step procedures. These are reviewed one at a time, followed by a discussion and an explanation of the steps used by The University.

Boyce (2001) suggests a simplistic three-step process: (1) Conduct assessment; (2) identify exposures; and, (3) address exposures.

Dacey (2003) suggests these steps, which are more about patch management than they are about vulnerability detection and response: (1) Inventory computers and the software applications and patches installed; (2)

identify relevant patches and workarounds and gather them in one location; (3) group systems by departments, machine types, or other logical divisions to easily manage patch deployment; (4) scan a network to determine the status of the patches and other corrections made to network machines (hosts and/or clients); (5) assess the machines against set criteria; (6) access a database of patches; (7) test patches; (8) deploy effective patches; and, (9) report information to various levels of management about the status of the network.

Berg (2002) suggests four general steps, but the last step is a plan—there should also be implementation and follow up analysis: (1) Inventory your systems; (2) manage the flow of information; (3) assess the information; and, (4) plan for response.

Bracklin (2003) suggests six steps, the last of which is also a plan: (1) Maintain an asset inventory; (2) manage information dissemination; (3) assess risk level of assets and vulnerabilities; (4) perform vulnerability assessment; (5) track remediation and report status; and, (6) plan for response.

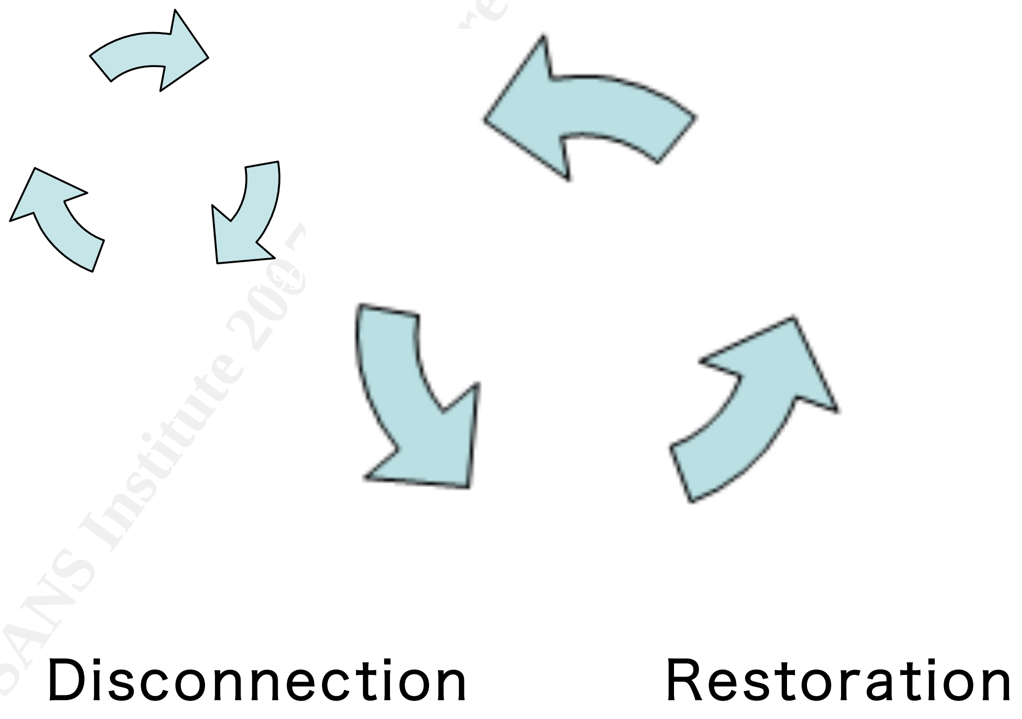
Mell, Bergeron, and Henning (2005) suggest 11 steps for vulnerability assessment and patch management, however, many of these steps appear to assume a large IT management role over the rest of the organization, which is not the case in a distributed university environment: (1) Inventory the organization's IT resources to determine which hardware equipment, operating systems, and software applications are used within the organization; (2) monitor security sources for vulnerability announcements, patch and non-patch remediations, and emerging threats that correspond to the software within the Patch and Vulnerability Group's (PVG's) system inventory; (3) prioritize the order in which the organization addresses remediating vulnerabilities; (4) create a database of remediations that need to be applied to the organization; (5) conduct testing of patches and non-patch remediations on IT devices that use standardized configurations; (6) oversee vulnerability remediation; (7) distribute vulnerability and remediation information to local administrators; (8) perform automated deployment of patches to IT devices using enterprise patch management tools; (9) configure automatic update of applications whenever possible and appropriate; (10) verify vulnerability remediation through network and host vulnerability scanning; and, (11) train administrators on how to apply vulnerability remediations.

VandenBrink (2006) gives a detailed explanation of how to do an automated vulnerability scan of users when they log on to VPN. Here is an overview: (1) Swatch monitors syslog from the VPN gateway, waiting for a successful VPN connection, then feeding that entire syslog event to a shell script; (2) a shell script parses out the command line arguments, then uses Nessus to scan the external (public internet IP) of the person who just VPN'd in; (3) a Nessus scan is saved to an HTML report file; and, (4) if the Nessus scan indicates a violation of company policy (i.e., if the remote firewall fails the scan on some or all identified tests), an alert e-mail is sent to the IT team responsible for security.

West-Brown, et al, (2003) suggests a CSIRT incident response decision tree which is simplified below to start with the discovery or report of a vulnerability: (1) Triage; (2) incident report; (3) analyze; (4) obtain contact information; (5) provide technical assistance; (6) coordinate Information and response; (7) repeat beginning at Step 3, if appropriate; and (8) resolution.

The West-Brown model is the one closest to what is used at The University. The others are either too general or, for the main part, assume a strictly controlled organizational structure which does not exist in a distributed university environment.

Consider the diagram below.



The diagram illustrates the overall process at The University from a top view: A smaller repetitive process of disconnection, on the left, drives a larger repetitive process of restoration, on the right. CSIRT does not have control over when either process begins. The initiation of the restoration process relies upon the user and/or LAN administrator and, in many cases, does not even occur. The restoration arrows are larger in the diagram to represent that the restoration process, when it occurs, takes approximately twice the CSIRT resources than the disconnection process does. The reasons for this will become clear as this paper progresses.

3.2. The Disconnection Process

The disconnection process at The University is the simpler process and typically takes 10 minutes per incident (after preparation has been done).

3.2.1. Preparation, Part 1—Monitor Vulnerability Announcements

From Mell, Bergeron, and Henning (2005): *Monitor security sources for vulnerability announcements, patch and non-patch remediations, and emerging threats that correspond to the software within the Patch and Vulnerability Group's (PVG's) system inventory*. CSIRT has to do this, anyway, so they might as well do it for the whole university. CSIRT should distribute this information to LAN administrators via a list server or by other means. Many LAN administrators will also be doing this—the more people doing it, the better.

3.2.2. Preparation, Part 2—Inventory Your Network (Create a Contact List)

Four of the above sources suggest that the network be inventoried. Specifically, CSIRT needs to know who to contact when a vulnerability is found. The Vulnerability Scanning Device (VSD) will return an IP address, which CSIRT must use to find who is responsible for the computer. The University associates one or more notification e-mail addresses with every IP address in the network. If the real contact person is unknown for an IP address, then a CSIRT e-mail address is used. For the remainder of this paper, this cross-reference between IP addresses and e-mail addresses is called The Contact List.

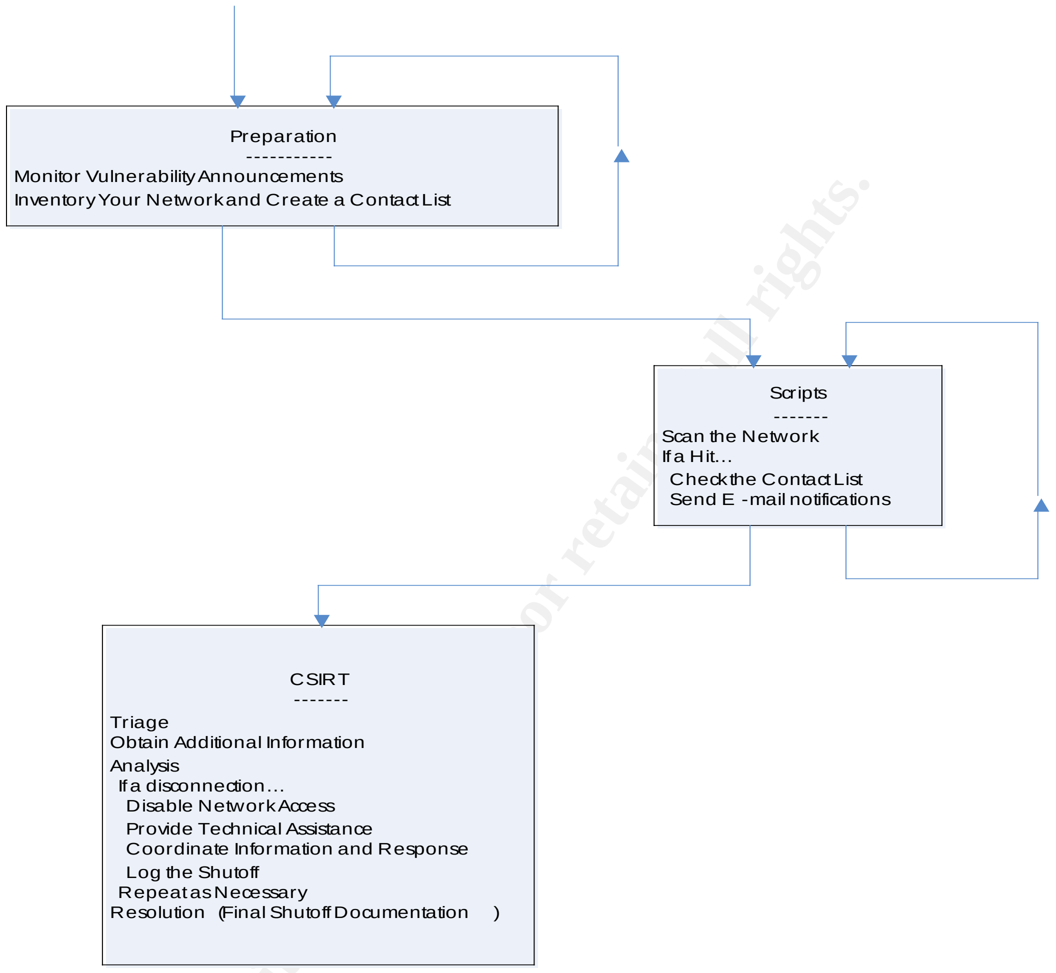
The University associates each e-mail address in The Contact List in a relational database with a real name, phone number, location, department, and comments. The result is an inventory of the active LAN administrators in The University. *Active* in this case means someone who hopefully can run down the hall and disconnect the network cable on a vulnerable computer.

When more than one LAN administrator works for an area in an academic unit, then all of the e-mail addresses for the LAN administrators are listed for the IP addresses in that area. As many appropriate LAN administrators for an area should be listed as possible, so that the first one who gets the message can take action. Also, if a LAN administrator is on vacation, hopefully a cohort who is on duty will still get notified.

A complete inventory cannot be done in advance, because IP addresses change frequently in a large environment—by the time one got through the list once, there would already be changes to it. So, start with whatever inventory can be put together in a reasonable amount of time and then make changes to it as necessary. Some LAN administrators will come to the attention of CSIRT many times, and others not at all—so the contact list over time will become updated for those that need to be contacted.

Many IP addresses will not be associated with LAN administrators, but rather will be associated with dialup users, wireless users, and students in dormitories. A CSIRT e-mail address is used on The Contact List for these groups so that CSIRT becomes aware of these special situations. DHCP IP addresses are also tagged with a CSIRT e-mail. An inventory of the IP addresses of notable computers on the network is also helpful. These preparation steps, of monitoring vulnerability announcements and inventorying your network, become an ongoing procedure even after the scanning begins.

The following diagram further illustrates these preliminary steps and leads into the scanning and following steps by showing the overall disconnection process. The below diagram ends with service being in a disconnected state.


Preparation

Monitor Vulnerability Announcements
Inventory Your Network and Create a Contact List
Scripts

Scan the Network
If a Hit...
Check the Contact List
Send E-mail notifications
CSIRT

Triage
Obtain Additional Information
Analysis
If a disconnection...
Disable Network Access
Provide Technical Assistance
Coordinate Information and Response
Log the Shutoff
Repeat as Necessary
Resolution (Final Shutoff Documentation)

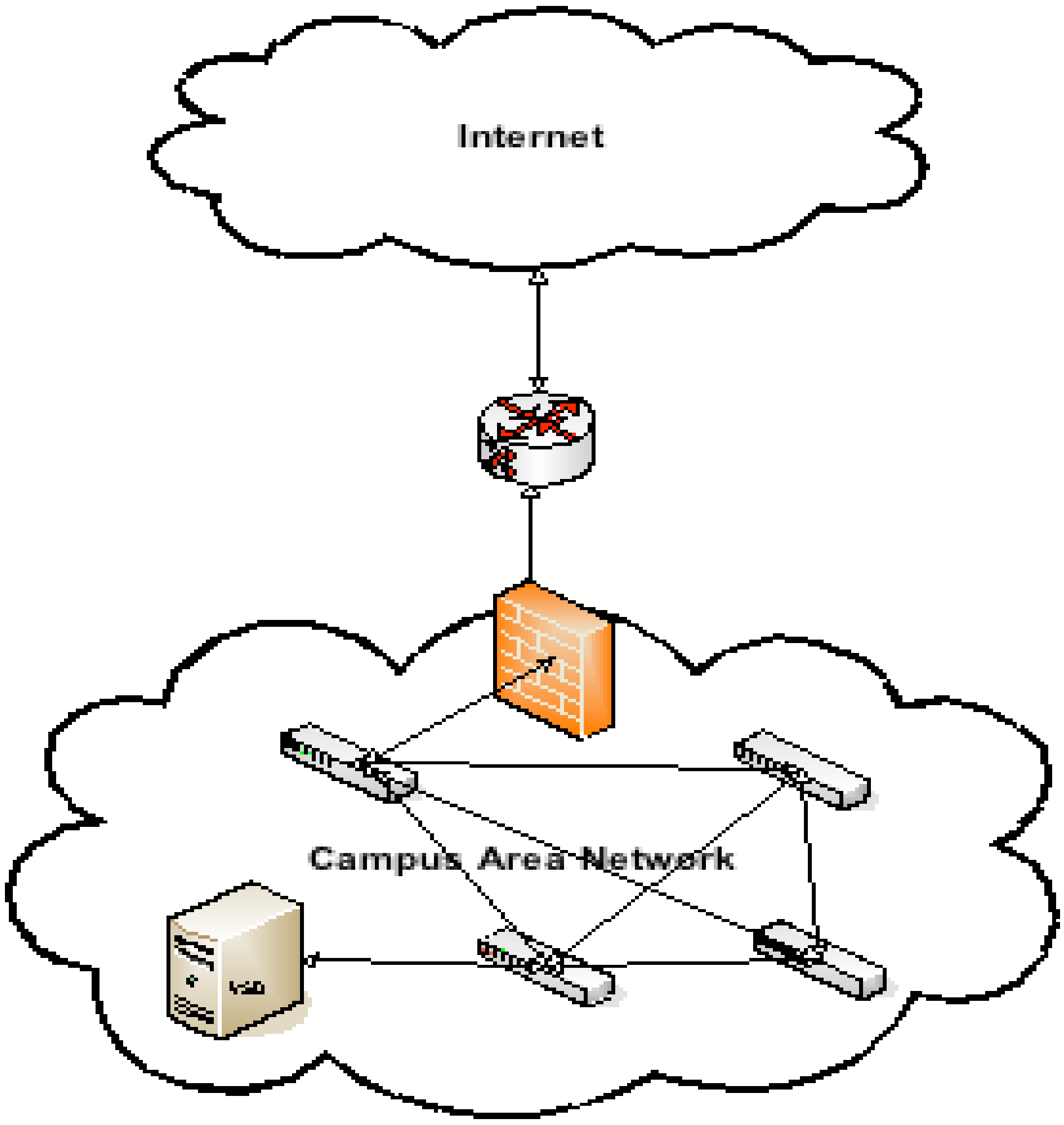
3.2.3. The Scripts, Part 1—Scan the Network

An automated job runs nearly continuous scans of the network. Various shell, Perl, and Ruby scripts process the output of the scans. These scripts are henceforth called The Scripts. Each accessible IP address produces one line of output during the scan. Here is an example of a line of output from noxscan:

```
aaa.bbb.ccc.ddd 445 MS04-007 SECURE:MS04-011 VULNERABLE:MS05-039 INCONCLUSIVE [00000000]
```

The pertinent information in the above output line is that IP Address aaa.bbb.ccc.ddd is vulnerable to the exploit described in MS04-011. The Scripts act on this information by searching for the string "VULNERABLE" on the scan output. When an IP address is scanned as being vulnerable, this paper will refer to it as *a hit*. An "INCONCLUSIVE" finding is not acted upon. A hit is considered to be an incident and causes incident response to be initiated. See the appendix for a representative listing of the script which calls and processes the vulnerability scanning.

The following diagram illustrates where to place your Vulnerability Scanning Device (VSD): I.e., plug it into your network mesh. (The switches in the diagram are intended to represent Layer 3 switches, and the mesh shown is not representative of the actual mesh at The University. The VSD is connected indirectly to one of the Layer 3 switches.)


Internet
Campus Area Network
VSD

3.2.4. The Scripts, Part 2—Check the Contact List

When The Scripts have a hit, they search The Contact List and obtain the e-mail addresses associated with the IP address involved in the hit. What happens next depends upon the e-mail addresses returned by The Contact List.

If the e-mail address indicates to The Scripts that the IP address is in a searchable log such as dialup, wireless, or DHCP, and if the appropriate log is available to The Scripts, then further information is obtained from the appropriate log by The Scripts. Beware of the temptation to send automated e-mails to users identified by dialup, wireless, and/or DHCP logs because these logs can become corrupted or unavailable in real time. The Scripts also check for notable IP addresses, in which case the hit is flagged. Note that this step does not have to be done by scripts. The Contact List and logs could be checked manually by CSIRT.

3.2.5. The Scripts, Part 3—Send E-mail Notifications

If the e-mails obtained from The Contact List represent at least one LAN administrator, then an automated e-mail notification is generated and sent to the LAN administrator/s. This automated notice contains a tracking number (West-Brown, et al, 2003), henceforth referred to as The Tracking Number. The e-mail also time stamps when the vulnerability was found. The e-mail states, among other things, that CSIRT intends to disconnect the vulnerable computer from the network and that the recipient should contact The Help Desk for restoration.

Here is an abbreviated sample automated e-mail notification:

```
From: CSIRT [mailto:csirt@abc.edu]
Sent: Thursday, June 14, 2007 8:50 AM
To: smith@abc.edu; DHCP
Cc: csirt@abc.edu
Subject: URGENT !!! Computer Security Vulnerability


To:  smith@abc.edu, DHCP <csirt@abc.edu>

You are receiving this automated message because
you are in the CSIRT database as being
the LAN Administrator for IP Address yyy.yyy.100.3
(subnet.abc.edu).

Please let us know at csirt@abc.edu if this IP
Address is not in your area.

Access to this IP Address is being disabled because
<your explanation here>.

The issues found are as follows:

MS04-007:  SECURE
MS04-011:  SECURE
MS06-040:  VULNERABLE

<Your warning and other messages go here.>

Please call the Help Desk at 555-5555 if you need
further assistance.  Refer to Tracking Number 99999
if you contact the Help Desk.

CONTACT THE HELP DESK TO RESTORE YOUR ACCESS.
```

The above e-mail indicates that this is a DHCP IP address, meaning that someone at CSIRT will probably have to look in the DHCP logs to trace the vulnerable computer. If the e-mail goes to more than one LAN administrator, then both are shown in the e-mail. This way each LAN administrator knows right away that the other one was also notified.

A copy of the LAN administrator e-mail should also be sent to The Help Desk or the Network Operations Center (NOC) or any other entity which needs immediate notification.

A copy of the LAN administrator e-mail is also sent to CSIRT. However, an additional e-mail is also sent to CSIRT with additional information. If the IP address of the hit is for a notable computer, then the CSIRT e-mail is flagged. In which case, CSIRT may telephone the LAN administrator or take other special action. Here is a representation of the additional e-mail which is sent to CSIRT:

```
        From: CSIRT Scanner [mailto:xxxxx@subnet.abc.edu]
        Sent: Thursday, June 14, 2007 8:58 AM
        To: csirt@abc.edu
        Subject: nxscan results 2007-06-14 08:57

        yyy.yyy.100.3 8 OS[Windows Server 2003 3790 Service Pack 1] MS04-007 SECURE, MS04-011
SECURE, MS06-040 VULNERABLE
        2007-06-14 08:50:03  New !!!  Not notable.  subnet.abc.edu.
        LAN admininistrator finder information:  For IP address range yyy.yyy.100.0 to
yyy.yyy.100.63 the listed name is Bob Smith at phone 555-4444 and e-mail address
smith@abc.edu.  Special message:  (None).
        LAN admininistrator finder information:  For IP address range yyy.yyy.100.0 to
yyy.yyy.100.63 the listed name is DHCP at phone (Blank) and e-mail address DHCP
<csirt@abc.edu>.  Special Message:  This IP Address appears to be DHCP.
        E-mail was sent to LAN admin/s.

        VPN lookup results:



        Visible IP Addresses:  3333.
        Unix = 33.
        Windows 5.0 = 333.
        Windows 5.1 = 333.
        Windows NT 4.0 = 33.
        Windows Server 2003 3790 = 3.
        Windows Server 2003 3790 Service Pack 1 = 33.
        Windows Server 2003 3790 Service Pack 2 = 33.
        Windows Server 2003 R2 3790 Service Pack 1 = 3.
        Windows Server 2003 R2 3790 Service Pack 2 = 3.
        Windows Vista (TM) Business 6000 = 3.
        Windows Vista (TM) Enterprise 6000 = 33.
        Windows Vista (TM) Ultimate 6000 = 3.
        Windows XP 3790 Service Pack 2 = 33.
```

The above e-mail tells CSIRT the following things:

- The IP address, operating system, if known, and what the computer at that IP address is vulnerable to.
- The time closest to when the detection was made (08:50:03 in the above example).
- That this is a new detection. Otherwise, it would be marked as being "historical." (Scans are frequent, and sometimes disconnections can not be made immediately, such as in the middle of the night.)
- This is not a notable computer.
- The domain name lookup.
- That a LAN administrator is assigned to this IP address, and what the range of the subnet is for this IP address.
- That this IP address is in a range assigned to DHCP, and what that range is. Note that the range assigned to the LAN administrator and the range assigned to DHCP both have to include this IP address, but the ranges do not otherwise have to be exactly the same.

- That an automated e-mail was sent to the LAN administrator.
- That VPN was not involved (the field is blank).
- A summary of visible IP addresses and operating systems is given. This information is readily available, anyway, and is included as a bonus.

Note that automated e-mails do not have to be sent; CSIRT can send them manually. CSIRT can still get the necessary information from a log file (see the appendix for a sample log file). Automated e-mails can be disruptive if not programmed properly. A limit should be established in The Scripts as to how many automated notifications can be made in a time period in order to prevent a potential self denial of service.

3.2.6. CSIRT, Part 1—Triage

Triage was suggested by West-Brown, et al (2003): *The triage function provides a single point of contact and the focal point for accepting, acknowledging, sorting, prioritizing, tracking, and passing on incoming information*. From this point on, somebody or a group owns The Tracking Number, with ownership beginning with CSIRT.

A vulnerability hit is considered to be an incident. The CSIRT Hotliner (West-Brown, et al, 2003) at The University accepts the Vulnerability Assessment (VA) incidents via the automated e-mail notices. The Hotliner starts an internal CSIRT worksheet for the incident. Although The University uses a database for the worksheet, this paper will refer to it as The Worksheet. A database does not have to be used as a worksheet; it could also be in the form of index cards, pads of printed forms, a spreadsheet, a text document, a journal entry, or another format. The Worksheet initially is filled out with all of the information from the e-mail plus other information that can be quickly obtained: The Tracking Number, the IP address, the type of vulnerability, the network hostname, the date and time of the offense, the tag for a notable computer, and a comments section. If The Scripts were able to obtain the username, the LAN administrator/s, or DHCP information, then this is also put in The Worksheet. The Worksheet also contains a checklist for further action. Acknowledgement for Vulnerability Assessment (VA) incidents is done by ownership of The Tracking Number being taken by or being assigned to an individual CSIRT member.

The Worksheet database also potentially contains other types of incidents besides vulnerabilities. The database sorts the active CSIRT incidents by type and also by network source, such as wireless, dialup, dormitory, or LAN. Any IP addresses of notable computers are also tagged in the sorting process. The incidents are prioritized from this sorted worksheet listing. Tracking is done via The Tracking Number. The Hotliner either accepts further ownership of The Tracking Number or assigns it to someone else.

3.2.7. CSIRT, Part 2—Incident Report (Obtain Additional Information)

West-Brown, et al, (2003) suggests an incident report as the next step. This implies the actual next step, which is to obtain more information. This additional information is placed into The Worksheet, i.e., a database, which can be used to produce an incident report, if desired.

Additional information to be obtained varies depending upon the type of network access, but could include a student's real name; the MAC address (and therefore manufacturer) of the offending computer; the building, room number, and jack number where the computer is located; the DHCP hostname of the computer; the network hostname of the computer; the switch and port number of the network connection; how many MAC addresses are using the same switch port; how long the computer has been using the current switch port; the actual usage of a notable computer (including what kind of data is on the computer); what academic unit the IP address is assigned to; previous vulnerability hits that have occurred with this computer and/or IP address; other types of incidents that have occurred with this computer and/or IP address; plus, any other information that CSIRT might come across while researching this incident. Sometimes, Network Engineering (NE) is contacted to trace a user or computer in order to gain more information. All of this information at this point remains internal to CSIRT and NE, if appropriate.

3.2.8. CSIRT, Part 3—Analysis

Not much analysis needs to be done about a vulnerable computer—it needs to be taken down. However, there are still things to consider:

- Was the proper e-mail notification sent?
- Is it a notable computer?
- Will shutting down the switch port shut down a classroom? Lab? Server room? LAN administrator's workshop? Other multiple units?
- Does the computer control special things such as an elevator? TV station? Airport activities? Medical equipment?
- Is the computer on a remote subnet that is off campus?

Additional procedures and notifications may kick into place with special situations such as the above. In many cases, the notification e-mail needs to be sent manually, either as the original notification, or with additional information for a LAN administrator.

3.2.9. CSIRT, Part 4—Disable Network Access

Network access can be disabled by any combination of types of network access controlled by IT including dialup, wireless, a switch port, network account, and other network restrictions. CSIRT can do this in most cases. The University disables the type of access on which the vulnerability was found: For example, wireless access is disabled when a vulnerability is found on a wireless connection. The user ID is determined based upon the IP address and time associated with where and when the vulnerability was detected. Then, access is disabled for that user ID on wireless. A similar procedure applies to dialup users. Thus, it is possible that a vulnerable user loses wireless access, and, then, subsequently loses dialup access.

The University typically disables direct network connections via switch ports. From the user's point of view, this usually means that the Ethernet jack in the wall goes dead. Network Engineering (NE) must become involved in order to trace an IP address to a switch port. CSIRT members with NE expertise may be granted access to the infrastructure to do this, or NE might provide one or more tools to CSIRT staff to assist with this. See your NE staff for more information on how to do this. If Network Engineering is involved in a shutoff, then CSIRT assigns The Tracking Number to Network Engineering, which then owns it.

Shutting off a jack often has collateral damage to the extent that other computers connected to the same jack will also be disabled from the network. If a user reconnects to another jack, then that jack will also be disabled once the vulnerability is detected, again.

NE might not have access to the pertinent switch port because of the distributed network environment (an entity besides NE might own the switch). In this type of case, NE can create an Access Control List (ACL) entry for the vulnerable IP address in the pertinent gateway device which NE does own which is closest to the problem. This does not completely solve the problem, but it can help to isolate the problem to a subnet.

3.2.10. CSIRT, Part 5—Provide Technical Assistance

Technical assistance (West-Brown, et al, 2003) means assisting LAN administrators in locating and disabling the vulnerable computer. It does not mean assistance in patching the computer—that duty falls on The Help Desk. Sometimes, the LAN administrator claims to not be able to find the vulnerable computer. In those cases, CSIRT often provides additional information to the LAN administrator to help find the computer.

For example, some vulnerable computers appear and disappear on the network in short time spans, and they may appear at different locations at different times. CSIRT can provide this timing and location information to the LAN administrator to help find the user. Sometimes, various jacks are disabled at various times in order to coerce a user to complain—at which time the user becomes identified.

3.2.11. CSIRT, Part 6—Coordinate Information and Response

CSIRT coordinates information and response (West-Brown, et al, 2003), i.e., The Worksheet and the checklist. When CSIRT assigns The Tracking Number to

NE, then NE owns it, but CSIRT maintains a listing of tracking numbers assigned to NE. After NE accomplishes a shutoff, The Tracking Number is reassigned to CSIRT, which checks the shutoff box on The Worksheet checklist. Any additional information obtained by NE is recorded by CSIRT on The Worksheet. Likewise, any additional information obtained from a LAN administrator is recorded by CSIRT on The Worksheet. Nothing prevents a LAN administrator from contacting a Network Engineer directly (except, maybe, the Network Engineer), but CSIRT controls the overall workflow.

3.2.12. CSIRT, Part 7—Log the Shutoff

The person who actually makes a network shutoff also makes an entry in a Shutoff Spreadsheet to notify others in the IT department that a shutoff has occurred. This entry specifies exactly what was disabled, a brief indication of why, and who did it. The shutoff could be a network account that was locked, dialup access disabled, wireless access disabled, the exact switch port which was disabled, a building, room number, and jack, if available. The MAC address is indicated, if available. The Tracking Number for the incident is also listed. Later, The Tracking Number can be used for the Help Desk, or others, to obtain more detailed information about the incident (more on this below).

From an office user's point of view, say, a clerical worker or a professor, the network connection just went dead for no apparent reason. Although CSIRT sent a notice to the LAN administrator, this information may have not made it to the user. A typical user will simply plug the same computer into a different network jack.

Usually after two or three jacks go dead, the user will complain to somebody. A detailed entry needs to be made in the Shutoff Spreadsheet for each disconnection for two reasons: 1) This makes it readily apparent that the same computer is involved in multiple instances; and, 2) this also makes it obvious that, when restoration is done, multiple jacks have to be restored.

The person who does a shutoff is the only person who has the information that needs to go in The Shutoff Spreadsheet. However, CSIRT is still responsible for verifying that the spreadsheet entry has been made. This might involve reminding the person to make the entry or it might involve obtaining the shutoff information from the individual and making the entry on behalf of the individual. This is still part of the coordination and response.

3.2.13. CSIRT, Part 8—Repeat as Necessary

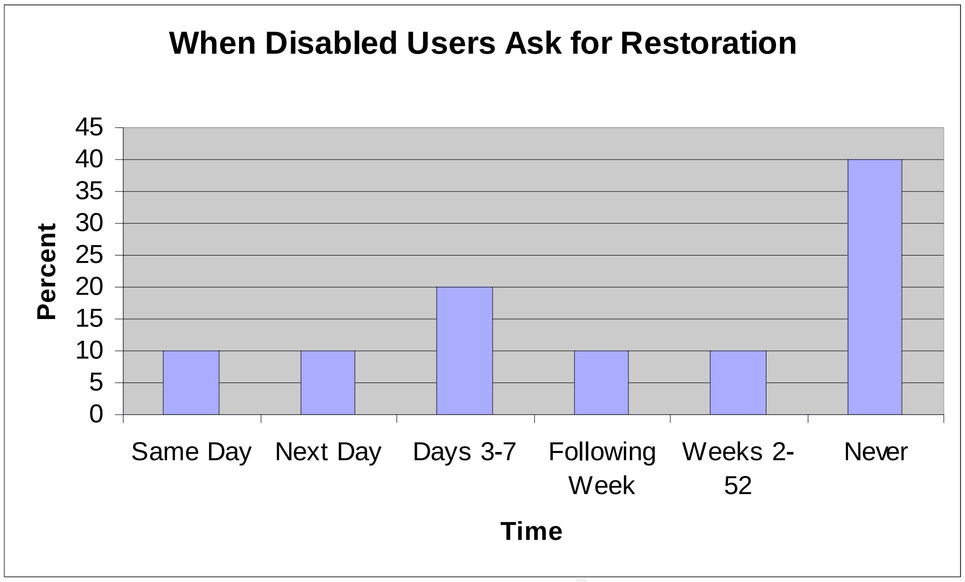
West-Brown, et al, (2003) sees this as being a repetitive process, and sometimes it is. For example, suppose that a user has configured a computer with the incorrect IP address, so that it is cross-referenced to one LAN administrator when it is actually located in another LAN administrator's department. Also, a user might just plug the vulnerable computer into another jack. Once one of these situations is discovered, then the process begins over at the analysis step.

3.2.14. CSIRT, Part 9—Resolution (Final Shutoff Documentation)

West-Brown, et al, (2003) suggests that resolution is the final step. *Resolution* means that the vulnerable computer has been taken off of the network. (The CSIRT goal is not to get the computer patched, but rather to get it off of the network.) CSIRT resolves the incident by assigning The Tracking Number to the Help Desk along with pertinent information about the shutoff. The University uses a separate database for this. The Help Desk does not need (or want) every detailed piece of information about the shutoff—just the information needed to respond when the complaint comes in (more on this below). At the end of the resolution step, The Help Desk owns The Tracking Number and CSIRT closes the case, sometimes never to see it, again.

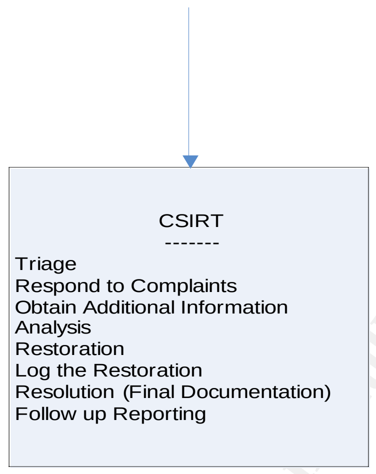
3.3. The Restoration Process

The restoration process at The University is often the more difficult process and typically takes 20 minutes per incident in CSIRT time. Often, this occurs because of poor communication between LAN administrators and the users, something which CSIRT has no control over. So, either a user initiates contact and does not understand what happened and cannot provide the information (IP address) necessary to easily restore access, or else a LAN administrator initiates contact having little or no idea who the user is or where the computer is located. Sometimes, no one ever calls to have network access restored.



The above chart gives a rule of thumb idea about how long it takes users/LAN administrators at The University to request restoration of network access after being disabled by CSIRT. Some 10 percent respond the same day, 10 percent the next day, 20 percent more before the end of the week, 10 percent the following week, and 10 more percent before the end of the year. Approximately 40 percent of the shutoffs never come back to the attention of CSIRT.

The following diagram illustrates the overall restoration process.



3.3.1. CSIRT, Part 1—Triage

Restorations have multiple potential input queues: Phone calls, walkups, Tracking Number referrals, and e-mails. The ideal situation is for all restoration requests to go through The Help Desk, in which case the tracking numbers become the input queue, where they can be more easily sorted and prioritized with the overall CSIRT workload.

3.3.2. CSIRT, Part 2—Respond to Complaints

CSIRT must respond carefully to students, LAN administrators, and the occasional professor who call and are upset because their network connection has been disabled. Sometimes this means responding to a CSIRT supervisor who has just gotten a telephone complaint from an upset user. These situations can be very time consuming, but are much easier if Worksheet documentation is impeccable.

3.3.3. CSIRT, Part 3—Obtain Additional Information

CSIRT attempts to obtain a piece of information that correctly identifies the original shutoff. The best case is when the Help Desk reassigns the original Tracking Number back to CSIRT. But, other times, the Help Desk creates a new Tracking Number. Sometimes, a LAN administrator sends an e-mail or telephones CSIRT directly. Sometimes, a Network Engineer contacts CSIRT asking about a switch port which has been disabled. CSIRT must document detailed information about each shutoff, so that any and all information can be cross-referenced when a complaint arrives. In the event that CSIRT did not make the shutoff, CSIRT must also be able to confidently state that a complaint is not related to a CSIRT incident.

3.3.4. CSIRT, Part 4—Analysis

CSIRT analyzes the likelihood that the computer has actually been patched. Usually CSIRT can take the recommendation of The Help Desk on this issue. However, the abuser might be a multiple offender, or have other circumstances to cause CSIRT to take special notice. Sometimes, for example, the abuser is required to physically bring the computer in to The Help Desk to be fixed.

3.3.5. CSIRT, Part 5—Restoration

CSIRT normally restores access in the manner in which it was disconnected: Network account, dialup, wireless, jack, multiple jacks, or whatever. Good recordkeeping in The Worksheet and The Shutoff Spreadsheet makes this easier. If Network Engineering was required to accomplish the shutoff, then Network Engineering might be required for the restoration—in which case The Tracking Number is assigned to Network Engineering, and reassigned back to CSIRT after restoration.

3.3.6. CSIRT, Part 6—Log the Restoration

The person who accomplishes the restoration is the person who logs the restoration in the Shutoff Spreadsheet. However, it is the responsibility of CSIRT to assure that this is done. Correctly marked restorations in the Shutoff Spreadsheet assist in future cases—for example, when CSIRT may be advised to *restore all disabled jacks in Brown Hall*.

3.3.7. CSIRT, Part 7—Resolution (Final Documentation)

CSIRT makes a final entry for The Tracking Number stating that access was restored and logged. The case is then closed.

3.3.8. CSIRT, Part 8—Follow up Reporting

The Worksheet and/or Tracking Number databases can be used periodically to create reports on criteria such as LAN administrators with the most problems, most common vulnerabilities found, days and times when vulnerabilities are most likely to be found, types of network connections with the most offenses, and other criteria as established by CSIRT.

3.4. Summary of Process

Here is a summary of the steps in the process used by The University, with some of the steps slightly renamed to more accurately describe the local situation:

Disconnection:

- Preparation, Part 1—Monitor Vulnerability Announcements
- Preparation, Part 2—Inventory Your Network and Create a Contact List
- The Scripts, Part 1—Scan the Network
- The Scripts, Part 2—Check the Contact List
- The Scripts, Part 3—Send E-mail Notifications
- CSIRT, Part 1—Triage
- CSIRT, Part 2—Obtain Additional Information
- CSIRT, Part 3—Analysis

- CSIRT, Part 4—Disable Network Access
- CSIRT, Part 5—Provide Technical Assistance
- CSIRT, Part 6—Coordinate Information and Response
- CSIRT, Part 7—Log the Shutoff
- CSIRT, Part 8—Repeat as Necessary
- CSIRT, Part 9—Resolution (Final Shutoff Documentation)

Restoration:

- CSIRT, Part 1—Triage
- CSIRT, Part 2—Respond to Complaints
- CSIRT, Part 3—Obtain Additional Information
- CSIRT, Part 4—Analysis
- CSIRT, Part 5—Restoration
- CSIRT, Part 6—Log the Restoration
- CSIRT, Part 7—Resolution (Final Documentation)
- CSIRT, Part 8—Follow up Reporting

Wack, Tracy, and Souppaya (2003) states V*ulnerablity scanning is a somewhat labor-intensive activity that requires a high degree of human involvement in interpreting the results*. The University experience corroborates that this is a labor-intensive activity: Even when a reliable tool is used, and CSIRT is confident of the correctness of the hits, the activity is labor-intensive in terms of documentation, communication with other departments, and the handling of complaints of the users and LAN administrators.

West-Brown, et al, (2003) has compared CSIRT to a firehouse to the extent that it has no control over its workload—it is continually on standby to react to both shutoffs and restorations at any time. However, the CSIRT *firemen* at The University are not *polishing the fire truck* waiting for calls, but, rather, are pro-actively searching for *fires*. Also, for most 10-minute shutoffs that CSIRT accomplishes, 20-minute restorations are stored up that can come back on CSIRT at any time. The *cleanups* take longer than putting out the *fires*.

Users and LAN administrators should be grateful when the *firemen* arrive to *save* a computer. However, in a distributed university environment, the users and LAN administrators are more likely to feel like they have been *caught*—sometimes fairly, other times unfairly. Therefore, CSIRT at The University is more like traffic policemen actively pursuing violators and making traffic stops.

4. Conclusions and Recommendations

Baseline automated vulnerability scanning is necessary but not sufficient for the network security of a distributed university environment. The nxscan tool appears to provide fast and reliable results in finding many unpatched computers. However, other tools are also necessary. Scripts can effectively automate baseline vulnerability scanning.

Dacey (2003) proposed the critical elements of patch management of management support, standardized policies, dedicated resources, risk assessment, and testing—for example on senior executive support: *Management recognition of information security risk and interest in taking steps to manage and understand risks, including ensuring that appropriate patches are deployed, is important to successfully implementing any information security-related process and ensuring that appropriate resources are applied.*

Essential elements of effective automated baseline vulnerability scanning in a distributed university environment are as follows:

- Administrators from the Information Technology Department up to the President of The University have to support the effort.
- Precise documentation is necessary in order for CSIRT to stand up to the inevitable complaints from users and LAN administrators when their network access is disconnected.
- The Help Desk staff has to understand why the shutoffs are necessary and be able to support CSIRT in discussions with users and LAN administrators.

- Good communication has to exist between CSIRT and other work groups in IT.

I recommend that automated baseline vulnerability scans should be done at universities where this is not taking place. Try nxscan manually and then check for updates on computers that are shown as being vulnerable to build confidence in the scanning program. Seek support of administrators from IT up to the president of the university. Be nice but convincing to recruit the good will of Network Engineering and The Help Desk in the process. Automate as much as is reasonably possible. Document completely and accurately.

Acknowledgements: Numerous administrators and staff at The University contributed significantly to the establishment of the procedure described in this paper.

Appendix A – run_nxscan.rb Screen Shot

The script which runs and processes the vulnerability scanning is run_nxscan.rb. (See Appendix B for a listing of the code for this script.) Here is a representative screenshot of this script when it runs:

```
[xxxxx@yyyyyyyy /yourpath/nxscan]# ./run_nxscan.rb
[xxxxx@yyyyyyyy /yourpath/nxscan]#
```

As you can see, there is nothing there. That is because the script was not intended to be run from the command line; it was intended to be run from /etc/crontab. See the documentation for your *nix operating system on how to run a script from crontab. No screen output is produced because in some operating systems, the screen output from crontab jobs is e-mailed to root, cluttering up the mailbox with un-useful messages.

Script messages are sent to a log. See Appendix C for sample output in this log file.

Appendix B – run_nxscan.rb Code

A representation of the Ruby script which runs the vulnerability scan follows below. This is the first Ruby script that I have ever written, so the script may not follow the best programming practices for Ruby. I took the production script, which I wrote, and changed a few things, such as directory paths and the subnet address, to disguise some details about the location of the script—the result is what appears below.

The script below refers to these other files:

- A log file. See Appendix C for a sample output of this log file.
- A history file. See Appendix D for a sample history file.
- A file with a list of subnets with notable computers. See Appendix E for a sample.
- A LAN admin finder file. See Appendix F.
- A tracking number file. See Appendix G.
- Three VPN files. A VPN log file and two scripts to manipulate this log file. See Appendix H.
- A temporary file for e-mailing CSIRT an alert. See Appendix I.

Enjoy…

```
#!/usr/local/bin/ruby
```

```
#################
# run_nxscan.rb #
#################
# This script runs and processes an nxscan.
# Requires extensive setup (see below).
# This script was tested on FreeBSD.
#
# Usage:  /yourpath/nxscan/run_nxscan.rb
#   Intended to be run from /etc/crontab.
#
# by Chet Langin
# Last update:  6/13/07


#############
# Copyright #
#############

# This code has the same copyright as the paper of which it is a part.


##############
# Disclaimer #
##############
# The author does not guarantee anything.  Use this
# code at your own risk.


#########
# Setup #
#########
# This is not a "click and go" program.
# Someone with extensive programming experience in
# Unix should set this up.  Plan on it taking weeks
# or months to do the preparation to make this work.
#
# Requires a history file (empty at the start)
# Requires a LAN administrator finder file.
# Requires a file of notable subnets.
#
# See the paper of which this code is a part on how
# to set up these files.


#####################
# Variable prefixes #
#####################
# These prefixes are used for some of the variables
# so that variable names are not repeated.
#   la_    "LAN Admin"
#   hist_  "History"
#   nota_  "Notable"


#########
# Timer #
#########

# Start timer.  This just times how long the script runs.
# It gives an idea of how long the scans are taking.
start_time = Time.now


###########
# Modules #
###########
# These are the modules required for the script to run
# For gethostbyname
require 'socket'
# For sending e-mails
require 'net/smtp'


######################
# Initiate variables #
######################
# Used to determine first scan of a new day.
# Real values will be set below.
old_day_of_year = 0
day_of_year = 0
# To enumerate the scans of a day.  Used to determine consecutive hits.
scan_of_day = 0
#  Means that a vulnerable computer is assigned to a LAN admin.
#  Default is true.
la_has_lan_admin = true
#  Keeps track of which vulnerabilities have been found.
# "Egg" is for debugging and means that this variable has not been used, yet.
ms04007_result = "Egg"
ms04011_result = "Egg"
ms06040_result = "Egg"


###################
# Initiate arrays #
###################
# Used to convert between octet to integer IP addresses
ip_address_int_array = Array.new
# The LAN admin contact/s for an IP Address.
# There has to be at least one contact, even if the contact is the CSIRT Team.
# There can be more than one contact.  This is the person or people best able
# to disconnect a computer or take care of it in an emergency.
lan_admin_array = Array.new
# This is a list of subnets with notable computers.  A "notable" computer
# might be one containting confidential information, a critical server,
# infrastructure, or anything else that makes it important.
# The list is obtained from a file which is created by the CSIRT Team.
notable_subnet_array = Array.new
# This is for a list of the visible IP Addresses.
visible_array = Array.new
# This is a list of the vulnerable IP addresses which were found,
# including any additional information which has been determined,
# including the time of detection.
vuln_array = Array.new
# This is a list of the e-mails addresses which will receive notice
# of the vulnerability.
la_to_array = Array.new


###################
# Initiate Hashes #
###################
# This is a list of IP addresses that have already been detected
# as being vulnerable during the same day.  This is used to prevent
# numerous e-mails being sent out for the same vulnerability.
# This history list is kept between scans in a file.
history_hash = Hash.new
# The scan reports operativing systems which it recognizes.
# These OS's are counted in this hash.  The visible_array (see
# above) turns these into a string for output.
os_hash = Hash.new
# This is used to keep track of IP addresses and detection
# times for subsequent lookup in a VPN log.  This way, the
# script can wait until the scan is over before it looks for
# users in VPN log files.
vpn_contact_hash = Hash.new


###################
# ip_addr_str2int #
###################
# A lamba procedure object to convert from IP Address string (4 octets) to integer.
#   Input parameter:  An IP address octet, such as 123.4.56.78.
#   Output:  The octet converted to an integer, such as 2063874126.
# Sanple usage:  ip_address_int = ip_addr_str2int.call("123.4.56.78")
#    ip_address_int => 2063874126
# The converted IP addresses can then be sorted and searched.
# This is used to find the LAN administrator/s for a given IP address.
ip_addr_str2int = lambda do |arg|
  ip_address_return_int = 0
  ip_address_string_array = arg.split(".")
  if (ip_address_string_array[0] and
      ip_address_string_array[1] and
      ip_address_string_array[2] and
      ip_address_string_array[3]    )
    ip_address_int_array[0] = ip_address_string_array[0].to_i * 256**3
    ip_address_int_array[1] = ip_address_string_array[1].to_i * 256**2
    ip_address_int_array[2] = ip_address_string_array[2].to_i * 256
    ip_address_int_array[3] = ip_address_string_array[3].to_i
    ip_address_int_array.each {|x| ip_address_return_int += x}
  end # if
  ip_address_return_int
end # ip_addr_str2int


############
# log file #
############
# time_string is used throughout this script to timestamp and report what is going on
# Since this script is intended to run in the background, nothing is sent to screen
# output.  Dependent upon your OS configuration, error messages are sent as e-mail to
# root.
# open a log file or not: set to "true" or "false"
log = true
if log
  log_file = File.new("/yourpath/nxscan/results/log.txt", "a")
  time_string = Time.now. \
      strftime("\n********** Starting run_nxscan.rb %Y-%m-%d %H:%M:%S **********\n")
  log_file.puts time_string
end # if log


################
# history file #
################
# Make a log entry, if appropriate.
if log
  time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Opening and loading history file.")
  log_file.puts time_string
end # if log
# Open and read the history file.
# Get the day of year and scan of day for the last time the scan was run.
# Get a list of all IP addresses which have been detected as being vulnerable this day.
File.open("/yourpath/nxscan/results/daily_history.txt", "r") do |hist_file|
  if hist_first_line = hist_file.gets # else
    hist_first_line.chomp!
    hist_first_line_array = hist_first_line.split
    if !old_day_of_year = hist_first_line_array[0]
      old_day_of_year = Time.now.yday
    end # if !old_day_of_year
    if !scan_of_day = hist_first_line_array[1]
      scan_of_day = 1
    end # if !scan_of_day
  else # if hist_first_line
    old_day_of_year = Time.now.yday
    scan_of_day = 1
  end # if/else hist_first_line
  while hist_line = hist_file.gets
    hist_line.chomp!
    history_hash[hist_line] = 1
  end # while hist_line
end # hist_file


#######################
# notable information #
#######################
# Make a log entry, if appropriate.
if log
    time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Loading notable information.")
    log_file.puts time_string
end # if log
# Load the information from the file list of subnets with notable computers.
notable_counter = 0
File.open("/yourpath/nxscan/externals/notable_list.txt", "r") do |nota_file|
  while nota_line = nota_file.gets
    nota_line.chomp!
    notable_subnet_array[notable_counter] = nota_line
    notable_counter += 1
  end # while
end # nota_file


################
# Run the scan #
################
# Make a log entry, if appropriate.
if log
    time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Starting scan.")
    log_file.puts time_string
    log_file.close                      # to flush the output
    log_file = File.new("/yourpath/nxscan/results/log.txt", "a")
end # if log
# Prepare some variables
vuln_counter = 0
visible_counter = 0
vpn_contact_hash.clear
# This is the actual scan.
# Ruby catches and processes the output while the scan is in progress.
# popen is for "pipe open"
# replace "y's" on line below with your subnet address.
scan_output = IO.popen("/yourpath/nxscan/nxscan yyy.yyy.0.0/16")
while line = scan_output.gets
  # This gets the time closest to when a vulnerability was detected.
  time_stamp = Time.now  # for searching logs
  line.chomp!

  # Look at the visible IP addresses
  # This includes both the ones that are, and are not, vulnerable
  if line !~ /NO CONNECTION/

    # Parse the line of scan output
    ip_address_array = line.split
    # Get the IP address as an octet
    ip_address = ip_address_array[0]

    # Get the results for each of the tests.
    # If there is a result for MS06-040, there is also a result for the others
    if line =~ /MS06-040/ # else
      ms04007_result = ip_address_array[-5]
      # Take off the comma
      ms04007_result.chop!
      ms04011_result = ip_address_array[-3]
      # Take off the comma
      ms04011_result.chop!
      ms06040_result = ip_address_array[-1]
    else # if line =~ /NS06-040/
      ms04007_result = "Not found"
      ms04011_result = "Not found"
      ms06040_result = "Not found"
    end # if/else line =~ /MS06-040/

    # Parse the original line, again, to get the OS.
    os_array = line.split(/[\[\]]/)
    if os_array[1]
      os = os_array[1]
      if os =~ /Unix|Windows/
        if os_hash[os] # else
          os_hash[os] += 1
        else # if os_hash[os]
          os_hash[os] = 1
        end # if/else os_hash[os]
      end # if os =~ /Unix|Windows/
    end # if os_arrary[1]

    # Create a list of visible IP addresses.
    time_string = Time.now.strftime("#{ip_address}*** %Y-%m-%d %H:%M:%S")
    visible_array[visible_counter] = time_string
    visible_counter += 1
  end # if !~ /NO CONNECTION/


  ########################################
  # Vulnerability processing starts here #
  ########################################

  # Some of the information has already been parsed above.
  domain_name = "(No domain name)"
  notable = "Not notable."
  if line =~ /VULN/
    # A vulnerable IP address has been detected.

    # Get the int value of the vulnerable IP address.
    # Used for finding the LAN admin/s.
    ip_address_int = ip_addr_str2int.call(ip_address)
```

```
# get the domain name
if socket_array = Socket.gethostbyname(ip_address)
  # This block is created to recover from socket errors.
  begin
    if(resolution_array = Socket.gethostbyaddr(socket_array[3], socket_array[2]))
      domain_name = resolution_array[0]
    end # if Socket.gethostbyaddr
    rescue SocketError
  end
end # if Socket.gethostbyname

# Get the subnet.
# Used to flag subnets with notable computers.
subnet = ""
if domain_name != "(No domain name)"
  subnet_array = domain_name.split('.')
  subnet = subnet_array[1]
end # if domain_name != "(No domain name)"
# Log it, if appropriate, for debugging.
if log
  time_string = Time.now. \
      strftime("%Y-%m-%d %H:%M:%S:  Domain name #{domain_name}.  subnet #{subnet}.")
  log_file.puts time_string
end # if log

# Check notability
notable_subnet_array.each {|x|
  if x == subnet
    notable = "NOTABLE."
  end # if
} # notable_subnet_array.each


##########################
# Check LAN Admin Finder #
##########################

# The LAN admin search also determines if an
# IP address represents a dialup user, a DHCP
# address, or a VPN user.

# Initialize variables
dhcp = false
dialup = false
vpn = false
number_of_lan_admins = 0
la_to_array.clear
lan_admin_array.clear

# Open and search the LAN admin finder file.
File.open("/yourpath/nxscan/externals/LAN_Admin_finder.txt", "r") \
    do |la_file|
  # Initialize variables.
  # "la_" = "lan admin" variables.
  # If the vulnerable IP address is for dialup, or VPN,
  # then there is no LAN Admin.  If it is DHCP, then
  # there may or may not be a LAN admin.  If the LAN
  # admin "name" is "CSIRT", then there is no
  # LAN admin.  Whether or not there is a LAN admin
  # determines whether or not an e-mail notification
  # is sent to the LAN admin (in addition to already
  # being sent to CSIRT).
  la_has_lan_admin = TRUE
  la_is_DHCP = FALSE
  la_is_Dialup = FALSE
  la_is_VPN = FALSE
  la_special_message = "Special message:  (None)."

  # Look at each line of the file.
  while la_line = la_file.gets
    la_line.chomp!

    # Parse the line.
    la_line_array = la_line.split("\t")

    # The data is organized in the file by a starting IP address and an
    # ending IP address for a subnet range that is assigned to a LAN
    # admin or a type of access, such as dialup or VPN.  If the LAN
    # admin is not known, then the range is assigned to CSIRT.
    # Index 0 is the starting IP address.
    # Index 1 is the ending IP address.
    # The data is sorted first by Index 0 and then by Index 1.

    # The starting IP address is determined and converted to an integer
    # in order to do the searching.
    la_starting_ip_address = la_line_array[0]
    la_starting_ip_address_int = ip_addr_str2int.call(la_starting_ip_address)

    # The vulnerable IP address has to be equal to or greater than the
    # starting IP address of the range being looked at.
    if ip_address_int >= la_starting_ip_address_int # else

      # If you are here, then the vulnerable IP address might be in
      # the current range, but the ending IP address of the range
      # must be looked at to know for sure.

      # Get the ending IP address of the range and convert it to an
      # integer so that it can be compared with the vulnerable IP address.
      la_ending_ip_address = la_line_array[1]
      la_ending_ip_address_int = ip_addr_str2int.call(la_ending_ip_address)

      # the vulnerable IP address must be less than or equal to the ending
      # IP address of the current range.  If it is, then this is a match.
      if ip_address_int <= la_ending_ip_address_int
        # If you are here, then the vulnerable IP address is in the current
        # range.  Get additional information about this subnet range from
        # the parsed line of information.
        la_name         = la_line_array[2]
        la_phone        = la_line_array[3]
        la_email        = la_line_array[4]

        # Determine type of connection

        # You have to plan ahead and make the "name" of the LAN admin "DHCP"
        # in the LAN admin finder file for all of the DHCP subnets, or this
        # won't work.
        if la_name == "DHCP"
          la_is_DHCP = TRUE
          la_special_message = "Special Message:  This IP Address appears to be DHCP."
        end

        # You have to make the LAN admin "name" "Dialup" in the LAN admin finder
        # file or this won't work.
        if la_name == "Dialup"
          la_is_Dialup = TRUE
          la_has_lan_admin = FALSE
        end

        # You have to make the LAN admin "name" "VPN" in the LAN admin finder
        # file or this won't work.
        if la_name == "VPN"
          la_is_VPN = TRUE
          la_has_lan_admin = FALSE
          vpn_contact_hash[ip_address] = time_stamp
        end

        # If the LAN admin "name" is "CSIRT" then there
        # is no LAN admin and no additional e-mail notification
        # is sent because CSIRT already gets an e-mail
        # notification.
        if la_name == "CSIRT"
          la_has_lan_admin = FALSE
        end

        # Create a string to store the LAN admin information.
        la_report_line = "LAN admininistrator finder information:  For IP " +
                          "address range #{la_starting_ip_address} to "     +
                          "#{la_ending_ip_address} the listed name is "     +
                          "#{la_name} at phone #{la_phone} and e-mail "     +
                          "address #{la_email}.  #{la_special_message}"
        lan_admin_array[number_of_lan_admins] = la_report_line

        # Add to the list of LAN admin e-mail addresses for this vulnerability.
        la_to_array[number_of_lan_admins] = la_email
        number_of_lan_admins += 1

      end # if ip_address_int <= la_ending_ip_address_int
      # If you are here, then the vulnerable IP address was greater
```

```
      # than the starting IP address of the range, but it was also
      # greater than the ending IP address of the range.  So, the
      # search continues by going through the while loop, again, and
      # getting the next line of the file to make the next comparison.
    else # if ip_address_int >= la_starting_ip_address_int
      # The vulnerable IP address is less than the starting
      # address of the range being looked at.  This means
      # that the vulnerable IP address will also not be in
      # any of the remaining ranges and that the search
      # is over:  All of the LAN Admins have been found.
      break
    end # if/else ip_address_int >= la_starting_ip_address_int
  end # while
end # File.open (LAN_Admin_finder.txt)


#################
# Check history #
#################
# It is a new find unless shown to be otherwise.
history_string = "New !!!"
if history_hash[ip_address] # else
  # This is a repeat
  history_string = "Historical."
else # if history_hash[ip_address]
  # This is new; it is stored for future reference.
  history_hash[ip_address] = 1
end # if/else history_hash[ip_address]


################
# Save and Log #
################

# Save the info to the array
lan_admin_info = ""
lan_admin_array.each {|x| lan_admin_info = lan_admin_info + x + "\n"}
if la_has_lan_admin # else
  lan_admin_info = lan_admin_info + "E-mail was sent to LAN admin/s.\n\n"
else
  lan_admin_info = lan_admin_info + "E-mail was NOT sent to LAN admin/s.\n\n"
end # if/else la_has_lan_admin

# Log LAN admin info
time_stamp_string = time_stamp.strftime("%Y-%m-%d %H:%M:%S")
time_string = "#{line}\n#{time_stamp_string}  #{history_string}  "  +
    "#{notable}  #{domain_name}.\n#{lan_admin_info}"
vuln_array[vuln_counter] = time_string
vuln_counter += 1

# Make a log entry, if appropriate.
if log
  log_file.puts time_string
```

```
      log_file.puts "MS04-007:  #{ms04007_result}."
      log_file.puts "MS04-011:  #{ms04011_result}."
      log_file.puts "MS06-040:  #{ms06040_result}."
    end # if log


    ########################
    # Send LAN admin e-mail #
    ########################
    # Send e-mail only if appropriate.
    if la_has_lan_admin and history_string =~ /New/

      # Create the "to" string
      la_to = ""
      la_to_array.each {|x| la_to = la_to + x + ", "}
      la_to.chop!
      la_to.chop!

      # Get the tracking number
      if log
          time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Getting the tracking number.")
          log_file.puts time_string
      end # if log
      tracking_number = ""
      File.open("/yourpath/nxscan/put_tracking_number_here.txt", "r") do |tracking_file|
        tracking_line = tracking_file.gets
        tracking_line.chomp!
        tracking_number = tracking_line.strip
      end # tracking_file


################################################
# This is where the actual e-mail message goes #
################################################

la_lan_admin_message = <<END_OF_LAN_ADMIN_MESSAGE

To:  #{la_to}

You are receiving this automated message because
you are in the university CSIRT database as being
the LAN Administrator for IP Address #{ip_address}
(#{domain_name}).

Please let us know at csirt\@abc.edu if this IP
Address is not in your area.

Access to this IP Address is being disabled because
<put your explanation here>.

The issues found are as follows:

MS04-007:  #{ms04007_result}
```

```
MS04-011:  #{ms04011_result}
MS06-040:  #{ms06040_result}

<Put your additional warnings and messages here.>

Please call the Help Desk at 555-5555 if you need
further assistance.  Refer to Case #{tracking_number}
if you contact the Help Desk.

CONTACT THE HELP DESK TO RESTORE YOUR ACCESS.

END_OF_LAN_ADMIN_MESSAGE


      ###################
      # Post the e-mail #
      ###################
      # cc the Network Operations Center (NOC), the Help Desk
      # and/ or anyone else who should also be notified.

      la_send_mail_to = la_to_array + ["csirt@abc.edu", "noc@abc.edu"]
      email_string = "From: CSIRT <csirt@abc.edu>\nTo: " +
                     "#{la_to}\nReply-to: CSIRT <csirt@" +
                     "abc.edu>\ncc:  csirt@abc.edu\nSubject: " +
                     "URGENT !!! Computer Security Vulnerability" +
                     "\n\n#{la_lan_admin_message}"
      Net::SMTP.start('localhost', 25) do |smtp|
        smtp.send_message email_string, 'csirt@abc.edu', la_send_mail_to
      end

    end # if la_has_lan_admin and history_string =~ /New/

  end # if line =~ /VULN/

end # while

# The scan is over


########################
# Get the VPN log file #
########################

# Make a log entry, if appropriate
if log
  time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Getting the VPN log file.")
  log_file.puts time_string
end # if log

# Call another script to pull down the latest VPN log file entries.
# The other script also combines the new and old log entries into
# a searchable file.
system("/yourpath/logs/vpn/get_latest_vpn_file.rb")
```

```
# If this is the first scan of a new day, wait a minute
# to be sure yesterday's VPN log file is created.  Then get it.
# These entries are converged with the other entries into a
# searchable file.
if scan_of_day == 1
  sleep 60
  system("/yourpath/logs/vpn/get_yesterdays_vpn_file.rb")
end # if scan_of_day ==1


###########################
# Search the VPN log file #
###########################

# Make a log entry, if appropriate.
if log
  time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Searching the VPN log file.")
  log_file.puts time_string
end # if log

# Initialize a variable.
vpn_contact_string = "VPN lookup results:\n"

# Search the VPN log file for each vulnerable VPN IP address.
# "key" is the IP address as an octet.
# "value" is a time object -- when the vulnerability was detected.
# Together they find the VPN user using that IP address at that time.
vpn_contact_hash.each do |key, value|

  # Convert the time object to a string which can be used to search the log.
  time_stamp_string = value.strftime("%Y-%m-%d %H:%M:%S")

  # This string will be used for output.
  vpn_contact_string = vpn_contact_string + "  #{key} at #{time_stamp_string}:"

  # Initialize two variables for holding the search results.
  # The user will be shown in one or both of these variables.
  # "Egg" means that the variable has not been set and is used for debugging.
  vpn_id1 = "Egg"
  vpn_id2 = "Egg"

  #  grep is your friend.  Only entries matching the IP address are returned.
  vpn_search = IO.popen("grep #{key}, /yourpath/logs/vpn/vpn_log.txt")

  # The search results are scanned and the last two entries for the
  # vulnerable IP address are kept.
  while vpn_line = vpn_search.gets
    vpn_line.chomp!
    vpn_id1 = vpn_id2
    vpn_id2 = vpn_line
  end # while vpn_line = vpn_search.gets
  # Make the VPN log entry easier to read by parsing each entry and saving
  # the pertinent information.
  vpn_id1_array = vpn_id1.split(",")
  vpn_id1 = vpn_id1_array[0] + ", " +
            vpn_id1_array[1] + ", " +
            vpn_id1_array[2] + ", " +   # this is the VPN ID
            vpn_id1_array[3] + ", " +
            vpn_id1_array[4] + ", " +
            vpn_id1_array[9]
  vpn_id2_array = vpn_id2.split(",")
  vpn_id2 = vpn_id2_array[0] + ", " +
            vpn_id2_array[1] + ", " +
            vpn_id2_array[2] + ", " +   # this is the VPN ID
            vpn_id2_array[3] + ", " +
            vpn_id2_array[4] + ", " +
            vpn_id2_array[9]

  # Save what you have found for later use.
  vpn_contact_string = vpn_contact_string + "\n     " + vpn_id1 + "\n     " + vpn_id2
end # vpn_contact_hash.each

# Make a log entry, if appropriate.
if log
  log_file.puts "   " + vpn_contact_string
end # if log
vpn_contact_string = vpn_contact_string + "\n\n"


############
# Clean up #
############

# Sort the vulnerability output array, logging, if appropriate.
if log
  time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Sorting vulnerability output.")
  log_file.puts time_string
end # if log
vuln_array.sort!

# Sort the visible output array, logging if appropriate
if log
  time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Sorting visible output.")
  log_file.puts time_string
end # if log
visible_array.sort!

# Save visible output to log, if appropriate
if log
  time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Saving visible output.")
  log_file.puts time_string
  # Uncomment below line for verbose output.
  #visible_array.each {|x| log_file.puts x}
  log_file.puts "  #{visible_counter} visible IP addresses."
```

```
        end # if log

        # Save os output to log, if appropriate.
        if log
          time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Saving OS output.")
          log_file.puts time_string
          os_hash.sort.each {|e| log_file.puts "  #{e[0]} = #{e[1]}."}
        end # if log

        # Save vulnerable output to a temporary file to be mailed to CSIRT.
        # Make a log entry, if appropriate.
        if log
            time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Saving vulnerability output.")
            log_file.puts time_string
        end # if log

        # Open the temporary file for output.
        File.open("/yourpath/nxscan/results/results_temp.txt", "w") do |file|

          # Save the vulnerability detection information.
          vuln_array.each {|x| file.puts x}

          # Save the VPN contact information, if any.
          file.puts  vpn_contact_string

          # Save information about the visible IP addresses.
          file.puts "Visible IP Addresses:  #{visible_counter}."

          # Save a summary of the operating systems identified.
          os_hash.sort.each {|e| file.puts "#{e[0]} = #{e[1]}."}

        end # file

        # Mail the temporary file
        # Make a log entry, if appropriate.
        if log
          time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Mailing report.")
          log_file.puts time_string
        end # if log

        # Only mail the temporary file if there has been a vulnerability detected.
        # "No e-mail is good e-mail."
        if vuln_counter > 0
          time_str = Time.now.strftime("%Y-%m-%d %H:%M")
          subject_line = "'nxscan results #{time_str}'"
          system("cat /yourpath/nxscan/results/results_temp.txt | mail -s #{subject_line}
csirt@abc.edu")
        end # if

        # Save the new history file
        # Recalculate day_of_year and scan_of_day
        day_of_year = Time.now.yday
        if day_of_year.to_i == old_day_of_year.to_i # else
```

```
          scan_of_day = scan_of_day.next
        else # if day_of_year.to_i == old_day_of_year.to_i
          scan_of_day = 1
          history_hash.clear
        end # if/else day_of_year.to_i == old_day_of_year.to_i
        # Log, if appropriate.
        if log
          time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S:  Saving history file:  #{day_of_year},
#{scan_of_day}.")
          log_file.puts time_string
        end # if log
        # Save the day, scan number, and vulnerable IP addresses.
        File.open("/yourpath/nxscan/results/daily_history.txt", "w") do |file|
          time_string = Time.now.strftime("%Y-%m-%d %H:%M:%S")
          file.puts "#{day_of_year} #{scan_of_day} #{time_string}"
          history_hash.sort.each {|e| file.puts "#{e[0]}"}
        end # file

        # Stop the timer
        end_time = Time.now
        elapsed_time = end_time - start_time
        elapsed_minutes = elapsed_time/60

        # Log time spent and close the log file, if appropriate.
        if log
          log_file.puts "#{elapsed_minutes} minutes elapsed time."
          log_file.close
        end # if log
```

Appendix C—log.txt Output

run_nxscan.rb normal output is written to the /yourpath/nxscan/results/log.txt file, a representative example of which follows. This logging can be turned on or off by setting the *log* variable in the code to *true* or *false*.

Numerous time stamps show the progress of the script. A write flush is done when the actually scanning starts to help the operator to be able to check the status of ongoing continuous scanning. Other information is presented to help with debugging. Also, a lot of information, such as operating system data, is also recorded, even though this information is also sometimes sent to CSIRT in an e-mail. The reason that this information is saved in this log file is that the e-mails are sent only when a vulnerability is detected. Whereas the information is saved in this log file during every single scan.

```
       ********** Starting run_nxscan.rb 2007-06-13 22:00:00 **********
       2007-06-13 22:00:00:  Opening and loading history file.
       2007-06-13 22:00:00:  Loading notable information.
       2007-06-13 22:00:00:  Starting scan.
       2007-06-13 22:03:32:  Domain name ws101125.vpn.abc.edu.  subnet vpn.
       yyy.yyy.101.125 8 OS[Windows 5.1] MS04-007 SECURE, MS04-011 SECURE, MS06-040
VULNERABLE
       2007-06-13 22:03:32  New !!!  Not notable.  ws101125.vpn.abc.edu.
       LAN admininistrator finder information:  For IP address range yyy.yyy.101.0 to
yyy.yyy.101.255 the listed name is VPN at phone (Blank) and e-mail address VPN
<csirt@abc.edu>.  S
       Special message:  (None).
       E-mail was NOT sent to LAN admin/s.

       MS04-007:  SECURE.
       MS04-011:  SECURE.
       MS06-040:  VULNERABLE.
       2007-06-13 22:12:34:  Getting the VPN log file.
       2007-06-13 22:12:34:  Searching the VPN log file.
         VPN lookup results:
         yyy.yyy.101.125 at 2007-06-13 22:03:32:
           06 13 2007, 21:59:00, panther, student, Stop, yyy.yyy.101.125
           06 13 2007, 22:00:38, nirvana, student, Start, yyy.yyy.101.125
       2007-06-13 22:12:34:  Sorting vulnerability output.
       2007-06-13 22:12:34:  Sorting visible output.
       2007-06-13 22:12:34:  Saving visible output.
         3333 visible IP addresses.
       2007-06-13 22:12:34:  Saving OS output.
         Unix = 33.
         Windows 5.0 = 333.
         Windows 5.1 = 333.
         Windows Server 2003 3790 = 3.
         Windows Server 2003 3790 Service Pack 1 = 33.
         Windows Server 2003 3790 Service Pack 2 = 33.
         Windows Server 2003 R2 3790 Service Pack 1 = 3.
```

```
  Windows Server 2003 R2 3790 Service Pack 2 = 3.
  Windows Vista (TM) Business 6000 = 3.
  Windows Vista (TM) Enterprise 6000 = 33.
  Windows Vista (TM) Ultimate 6000 = 3.
  Windows XP 3790 Service Pack 2 = 3.
2007-06-13 22:12:34:  Saving vulnerability output.
2007-06-13 22:12:34:  Mailing report.
2007-06-13 22:12:34:  Saving history file:  164, 89.
10.5739082 minutes elapsed time.
```

The above log shows that a vulnerability was detected at approximately 22:03:32. The last two VPN log entries for the vulnerable IP address were VPN user panther who signed off at 21:59:00 and VPN user nirvana who signed on at 22:00:38, so it is reasonable to conclude that nirvana is the one with the vulnerable computer. Showing the last two VPN log entries is usually sufficient for a human to identify the appropriate user. Keep in mind that live VPN log files can become corrupted or unavailable, so this information may not always be reliable.

Appendix D—The History File

A /yourpath/nxscan/results/daily_history.txt file is kept in order to keep track of when Midnight rolls over (for manipulation of the VPN log file) and to keep track of what IP addresses have been found vulnerable during a single day. Here is a representative sample of this file:

```
165 64 2007-06-14 13:47:28
yyy.yyy.100.3
```

The above file shows that this is the 165th day of the year and that this is the 64th scan of the day. The file was last written to on the date and time given. The IP address shown has already been detected as being vulnerable on this day. If it is detected, again, then a second e-mail will not be sent to the LAN administrator. An additional e-mail would be sent to CSIRT, but this e-mail would be marked "Historical".

Any additional IP addresses in this file would be listed on subsequent lines.

If an IP address is detected as being vulnerable and the address does not appear in this file, then the script treats the IP address as being new: An e-mail would be sent to the LAN administrator, if otherwise appropriate.

Appendix E—The Notable Computers File

This file contains a list of subnet names containing notable computers, one per line. It might look like this:

```
account
admin
business
chief
gold
records
```

If a vulnerable IP address resolves to a subnet containing one of the above names, then the automated e-mail is flagged so that CSIRT and the appropriate LAN administrator can give it special attention.

Appendix F—The LAN Administrator Finder File

The LAN administrator finder file is a list of IP addresses and ranges with their associated LAN administrators in a tab-delimited text file. One can keep this information in a database, export it to a spreadsheet, and export the spreadsheet to the text file. That way you can easily look up a phone number or e-mail address in the database, do a manual IP address lookup in the spreadsheet, or let the script do it automatically, in the text file. Only a representation of the text file will be shown here.

The file uses “from” and “to” IP addresses to express ranges because this allows the greatest flexibility: From a single IP address to a range across subnets. <tab> will be used to indicate a tab.

```
yyy.yyy.0.0<tab>yyy.yyy.0.255<tab>CSIRT<tab>555-1111<tab>csirt@abc.edu
yyy.yyy.1.0<tab>yyy.yyy.1.255<tab>John Smith<tab>555-2222<tab>jsmith@abc.edu
yyy.yyy.1.0<tab>yyy.yyy.2.255<tab>John Jones<tab>555-3333<tab>john@abc.edu
yyy.yyy.2.0<tab>yyy.yyy.2.255<tab>Betty Edwards<tab>555-6666<tab>better@abc.edu
yyy.yyy.2.64<tab>yyy.yyy.2.95<tab>DHCP<tab>(None)<tab>csirt@abc.edu
yyy.yyy.2.98<tab>yyy.yyy.2.98<tab>Printer<tab>(None)<tab>csirt@abc.edu
```

Note above how the ranges can overlap. In the above example, John Jones might be the supervisor of John Smith and Betty Edwards, explaining why his range overlaps both of their ranges. The file is presorted. The starting IP addresses are sorted first, and the ending IP addresses are sorted second. Then, the search just compares each line as the file is read in. Once the vulnerable IP address is less than the starting address of a line entry, then the file read is terminated and the search is over.

(I have not done a time analysis on this, but this method is probably about as quick on the average as reading the entire file in and then doing, say, a binary search.)

Appendix G—The Tracking Number File

This file just has the tracking number at the start of the file, like this:

```
99999
```

The reason for this is so that the tracking number can be updated while the script runs continuously. Then, the script reads the file for the new tracking number when a new number is needed.

Appendix H—The Three VPN-Related Files

The three VPN-related files are the VPN log file, itself, and two scripts that are used to set up the VPN log file: vpn_log.txt, get_latest_vpn_file.rb, and get_yesterdays_vpn_file.rb.

get_latest_vpn_file.rb is a script that pulls down the active VPN log file and sets up a custom log file, vpn_log.txt, for searching. The code for get_latest_vpn_file.rb is not shown in this paper because the code depends upon how the active VPN log file is set up in an organization and where the active VPN log file is located. *Active* means the live VPN log file that is actively recording VPN logons and logoffs. If you are going to do VPN lookups in real time then you need access to this active VPN log file.

Before starting any scanning, manually copy the active VPN log file to /yourpath/logs/vpn/vpn_log.txt. Then, write get_latest_vpn_file.rb to 1) pull down the active VPN log file, 2) concatenate the active file to vpn_log.txt, and 3) do a unique sort on vpn_log.txt. A unique sort will sort the vpn_log.txt file in time order without any duplicate entries. The .rb in get_latest_vpnfile.rb denotes a Ruby script, but this could just as well be a Perl or shell script.

Your active VPN log file possibly rolls over at midnight, in which case you need another script, get_yesterdays_vpn_file.rb to be sure that you catch all of the log entries just before midnight. (Otherwise, you might miss VPN log entries between your last scan and Midnight.) run_nxscan.rb watches for the first scan of the day, i.e., the first scan after Midnight, waits a minute to be sure the log rollover has time to finish, and then runs get_yesterdays_vpn_file.rb, which should pull down yesterday's file, concatenate this file to vpn_log.txt, and do a unique sort on vpn_log.txt.

Once all of the above is done, then run_nxscan.rb searchs vpn_log.txt for one or more offenders, if appropriate.

A nice side benefit of get_latest_vpn.file.rb is that it can be run manually any time that you need to search the active VPN log file—just run this script then grep vpn_log.txt.

A disadvantage to having scripts access your VPN log file is that there are authentication issues involved with scripts accessing this type of data. You will have to decide if you want to risk accessing data with trusted systems, or stored passwords and/or keys.

If you decide to go this route, then you could also extend the scripts discussed in this paper to do automatic lookups for dialup, DHCP, and other log files which you may have.

Appendix I—The Temporary File

The temporary file is a workspace to create the body of the e-mail that will go to CSIRT. This temporary file is recreated each time a scan detects one or more vulnerabilities. Here is what the temporary file would look like for an example e-mail shown earlier in this paper:

```
        yyy.yyy.100.3 8 OS[Windows Server 2003 3790 Service Pack 1] MS04-007 SECURE, MS04-011
SECURE, MS06-040 VULNERABLE
        2007-06-14 08:50:03  New !!!  Not notable.  subnet.abc.edu.
        LAN admininistrator finder information:  For IP address range yyy.yyy.100.0 to
yyy.yyy.100.63 the listed name is Bob Smith at phone 555-4444 and e-mail address
smith@abc.edu.  Special message:  (None).
        LAN admininistrator finder information:  For IP address range yyy.yyy.100.0 to
yyy.yyy.100.63 the listed name is DHCP at phone (Blank) and e-mail address DHCP
<csirt@abc.edu>.  Special Message:  This IP Address appears to be DHCP.
        E-mail was sent to LAN admin/s.

        VPN lookup results:
```

```
Visible IP Addresses:  3333.
Unix = 33.
Windows 5.0 = 333.
Windows 5.1 = 333.
Windows NT 4.0 = 33.
Windows Server 2003 3790 = 3.
Windows Server 2003 3790 Service Pack 1 = 33.
Windows Server 2003 3790 Service Pack 2 = 33.
Windows Server 2003 R2 3790 Service Pack 1 = 3.
Windows Server 2003 R2 3790 Service Pack 2 = 3.
Windows Vista (TM) Business 6000 = 3.
Windows Vista (TM) Enterprise 6000 = 33.
Windows Vista (TM) Ultimate 6000 = 3.
Windows XP 3790 Service Pack 2 = 33.
```

The contents of the above file are sent as an e-mail to CSIRT.

Appendix J—References

Berg, A. (2002, February). Feeling Vulnerable? Information Security Magazine [Online] Available: http://infosecuritymag.techtarget.com/2002/feb/features_vulnerable.shtml

Boyce, R. (2001). Vulnerability Assessments: The Pro-active Steps to Secure Your Organization. SANS Institute.

Brackin, C. (2003, October 15). Vulnerability Management: Tools, Challenges and Best Practices. Sans Institute.

Dacey, R.F. (2003, September 10). Effective Patch Management is Critical to Mitigating Software Vulnerabilities. United States General Accounting Office (GAO).

Deraison, R., Rathaus, N., Moore, H.D., Alder, R., Theall, G.A., Johnston, A., &

Alderson, J. (2004). Nessus Network Scanning. 217. Rockland, MA: Syngress Publishing, Inc.

Dodge, A. (2007). Educational Security Incidents (ESI) Year in Review—2006 [Online]. Available: http://www.adamdodge.com/esi/yir_2006

Houghton, K. (2003, July). Vulnerabilities and Vulnerability Scanning. SANS Institutute.

Mell, P., Bergeron, T., & Henning, D. (2005, November). Creating a Patch and Vulnerability Management Program. National Institute of Standards and Technology (NIST).

Microsoft. (2007, February 28). MS04-007: An ASN.1 vulnerability could allow code execution [Online]. Available: http://support.microsoft.com/kb/828028

Microsoft. (2007, February 28). MS04-011: Security Update for Microsoft Windows [Online]. Available: http://support.microsoft.com/kb/835732

Microsoft. (2007, February 28). MS05-039: Vulnerability in Plug and Play could allow remote code execution and elevation of privilege [Online]. Available: http://support.microsoft.com/kb/899588

Microsoft. (2007, February 28). MS06-040: Vulnerability in Server service could allow remote code execution [Online]. Available: http://support.microsoft.com/kb/921883

Microsoft. (2007, February 28). Microsoft Security Bulletin MS04-007 [Online]. Available: http://www.microsoft.com/technet/security/bulletin/MS04-007.mspx

Microsoft. (2007, February 28). Microsoft Security Bulletin MS04-011 [Online]. Available: http://www.microsoft.com/technet/security/bulletin/MS04-011.mspx

Microsoft. (2007, February 28). Microsoft Security Bulletin MS05-039 [Online].

Available: http://www.microsoft.com/technet/security/Bulletin/MS05-039.mspx

Microsoft. (2007, February 28). Microsoft Security Bulletin MS06-040 [Online]. Available: http://www.microsoft.com/technet/security/bulletin/MS06-040.mspx

MITRE. (2007, February 28). CVE-2003-0818 [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0818

MITRE. (2007, February 28). CVE-2003-0533 [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0533

MITRE. (2007, February 28). CVE-2005-1983 [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-1983

MITRE. (2007, February 28). CVE-2006-3439 [Online]. Available: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3439

Potter, Bruce. (2007, February 28). Host Integrity Monitoring (Slide 8) [Online]. Available: http://www.shmoo.com/~gdead/pres/Potter-LinuxWorld06.ppt

Tenable Network Security. (2007, June 20). Nessus Vulnerability Scanner [Online]. Available: http://www.nessus.org

VandenBrink, R. (2006, September 12). VPNScan: Extending the Audit and Compliance Perimeter. SANS Institute.

Wack, J., Tracy, M., Souppaya, M. (2003, October). Guideline on Network Security Testing. C-6. National Institute of Standards and Technology (NIST).

West-Brown, M.J., Stikvoort, D., Kossakowski, K-P., Killcrece, G., Ruefle, R., & Zajicek, M. (2003, April). Handbook for Computer Security Incident Response Teams (CSIRT), 2nd Ed. Carnegie Mellon Software Engineering Institute [Online]. Available: http://www.sei.cmu.edu/publications/documents/03.reports/03hb002.html

Williams, J.R. (2003, August 28). Managing vulnerabilities exposed by Windows services. SANS Institute.

Wilson, J.R., & Kelling, G.L. (1982, March). Broken Windows. The Atlantic Monthly, Vol. 249, No. 3, 29-38.

York University. (2007, February 28). Information Security Unsupported Tools [Online]. Available: http://infosec.yorku.ca/tools/