Global Information Assurance Certification Paper

“The Blind Leading The Blind”
Sadmind/IIS Worm

SANS GCIH Practical
V2.1a
Option 1

Rich Barger
September 30, 2003

Table of Contents

"The Blind Leading The Blind" 1
Table of Contents 2
Abstract 3

Part –1 The Exploit 4
Name 5
Operating Systems Affected 5
Protocol & Services 5
A Brief Description of the sadmind/IIS Worm 5
Variants 5
Also Known As 6
References 6
Executables and Individual Exploit Source Code 7

Part 2 – The Attack 8
Diagram of Network 8
Network Description 9
Protocol Description 10
How the Exploit works 10
Diagram of the attack 16
Signature of the attack 17
How to protect against the sadmind/IIS 20

Part 3 – The Incident Handling Process 21
Preparation 21
Identification 23
Containment 24
Eradication 25
Recovery 26
Lessons Learned 27
References 30

Abstract

Way back in the day, before I had contracted my “Left Click Finger” and “IP Eye,” two rare conditions that I began to suffer from after the second Code Red attack, I used to be an Administrator. Both maladies were aggravated by countless hours in front of Intrusion Detection Sensors.

You know those guys who get lunch breaks, and usually get to go home at normal hours on Fridays? Needless to say, my first experience with a security related incident was as an Administrator, and a Junior Administrator at that.

Doesn’t that take you back? Junior Administration, where working as root was a forbidden land and the “grep” command was a new and exciting revelation.

I decided to select this incident for my practical topic, not only because it is an example of one of my most favorite war stories to tell over a beer with my closest of geeks, but also as an example of what not to do.

This paper will describe the steps I took in handling and incident when I was very new to the IT industry and had very little information system security training. Looking back, I can see the mistakes I took when I “handled” this sadmind/IIS infection. In part three I will break down the failures of the incident handling process, hopefully one will be able to get an idea of what it will be like in the trenches when things are moving fast and your fearless leader or Chief Information Officer (CIO) is either wetting his pants one minute or breathing fire the next.

I’ve always seemed to be fortunate when learning from the mistakes of others, not only does it afford you the opportunity to point and laugh at them, but you can contrast the example given to the scenario that you maybe facing at that time. This tends to give you quite an advantage when you no longer have a /dev/rmt directory.

Hopefully, this story will find its way to the front lines. To the ears of other aspiring Junior Administrators, all huddled around a flickering flat screen somewhere late at night, shaking in their Dockers as an old grizzled Security Engineer spreads the gospel of properly configured firewalls and keeping current patches.

Part –1 The Exploit

There I was again. In sheer euphoria, those poor suckers didn't even see me leave the server room; I chuckled to myself as I sunk my teeth into my Chalupa. Suddenly, the shrieking sounds of my Nextel filled the overly festive Taco Bell.

"Hello," I sighed as to punish the individual that interrupted my meal.

"Yes, this is Sergeant Smith from the US Army National Guard. We have reason to believe one of your systems have defaced one of our websites." "Umm, how do you know for sure?" I replied. "Is SUN_TEST1 a hostname of one of your systems?" My confusion then turned to curiosity. "Yeah, it is…who is this again?" I asked, trying to buy time for a sip of soda.

"This is Sergeant Smith US Army National Guard," he replied. The good sergeant then began to patiently explain that sometime that morning a system from my organizations network had defaced an Army web server. Unsure of what to do, the sergeant and I exchanged contact information somewhat reminiscent of an "e-traffic accident." However, this scenario was more like a hit and run. As I understood, FBI agents usually frown on government web defacements.

As I drove back to work, I tried to figure out how a dust covered Ultra20 test Sun Solaris 2.6 system hacked an Army website. This test workstation was nested back in the communications closet. The only time someone ever physically touched it was when it needed to be physically rebooted.

After I parked my car and shot up the elevator, I contemplated just how I would inform Lenny, the CIO. Lenny claimed to be an old VAX administrator, who just received his MBA. Just in my few months of working with him I realized that his technical competence was somewhat questionable. When in a planning meeting about implementing new database architectures, He was the first to rule out a growing file system because "disks were expensive."

Lenny never really understood why it was that incremental backups were just as beneficial as doing "fulls" everyday and establishing an FTP session seemed to be the backbone of his technical expertise.

I ducked my head into Lenny's office. "Lenny you got a minute?" He was listening to his voice mail over speakerphone. A man had left a livid message claming that his website had been defaced and he wanted somebody to call his lawyer back to explain. Lenny informed me that this is the second phone call today, where someone had claimed that we hacked a website

"Actually, Lenny it's the third, I just got a call from the Army."

Lenny became visibly upset as his Mont Blanc went sent sailing across the room followed by his choice of expletives. "I want every contractor to meet me in the conference room and in the meantime, you make this go away!" So off I went to catch a hacker.

Name

Sadmind/IIS Worm

CVE-1999-0977

Operating Systems Affected

Solaris 2.3*, 2.4* 2.5, 2.5.1, 2.6, and 7
**Note: Solaris 2.3 and 2.4 are applicable if sadmind was installed as part of the Sun Solstice Adminsuite.*

Microsoft Windows NT / Windows 2000 running Microsoft IIS 4.0 / 5.0

Protocol & Services

111/TCP & UDP – Sun Remote Procedure Call (sunrpc)

32771/UDP (*Solaris 2.3 – 2.5.1) - Sun RPC portmapper / rpcbind

80/TCP – Hyper Text Transfer Protocol (HTTP)

600/TCP – sadmind/IIS worm (/bin/sh)

A Brief Description of the sadmind/IIS Worm

The sadmind/IIS Worm is based on two vulnerabilities. The first of which would be unpatched versions of Solaris (Solaris 2.3 through Solaris 7), and secondly unpatched versions of Microsoft IIS 4.0 and 5.0.
Initially the worm exploits a buffer overflow in Solaris Solstice sadmind process. The worm then installs a payload onto the vulnerable Solaris system. Not only does the software installed then begin to actively locate and infect other vulnerable Solaris systems, but it also begins to locate and attempt to deface Microsoft IIS 4.0 and 5.0 web servers by attempting to exploit an extended UNICODE directory traversal vulnerability.
Once 2,000 Microsoft IIS servers have been compromised the worm finally turns on the host Solaris system and modifies the index.html file. Additionally, the sadmind/IIS Worm will append "+ +" to the .rhost file. This subsequently allows all other systems on the same network of the compromised Solaris system to be trusted for remote authentication. A backdoor root shell is also executed and TCP port 600 is subsequently activated and begins to listen.

Variants

Worm.SadMind.b - The Worm.SadMind.b variant is very similar to the sadmind/IIS version. However, a few executable utilities had been recompiled.

Worm.SadMind.c – The Worm.SadMind.c variant only differs from the .A and .B variants of the worm where the "index.html" file that is used to overwrite local "index.html" files on compromised Solaris systems after defacing 2000 IIS servers has been modified. However the defaced Microsoft IIS 4.0 and 5.0 servers will include the same defacement as those hacked by version .A and .B.

Also Known As

ELF_SADMIND.A

Backdoor.Sadmind.dr

Sadmin-iis

Unix/Sadmind

References

CERT/CC Advisories (Solaris):

CERT® Advisory CA-2001-11

Sadmind/IIS Worm
http://www.cert.org/advisories/CA-2001-11.html

CERT® Advisory CA-1999-16

Buffer Overflow in Sun Solstice AdminSuite Daemon sadmind
http://www.cert.org/advisories/CA-1999-16.html

Vulnerability Notes (Solaris):

Vulnerability Note VU#28934

Sun Solaris sadmind buffer overflow in amsl_verify when requesting NETMGT_PROC_SERVICE
http://www.kb.cert.org/vuls/id/28934

Vulnerability Notes (Microsoft IIS):

Vulnerability Note VU#111677

Microsoft IIS 4.0 / 5.0 vulnerable to directory traversal via extended unicode in url (MS00-078)
http://www.kb.cert.org/vuls/id/111677

Common Vulnerability Exposure (CVE) Analysis

CVE-1999-0977
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-1999-0977

CVE-2000-0884
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=2000-0884

Antivirus Vendors

Symantec – Backdoor Sadmind
http://www.symantec.com/avcenter/venc/data/sadmind-iis.html

Sophos - Unix/SadMind
http://www.sophos.com/virusinfo/analyses/unixsadmind.html

Executables and Individual Exploit Source Code

Digitaloffense.net -
http://www.digitaloffense.net/worms/sadmind-unicode/
***Please exercise caution when downloading any files from this website.**

Securiteam –
http://www.securiteam.com/exploits/3P5Q1Q0QAO.html

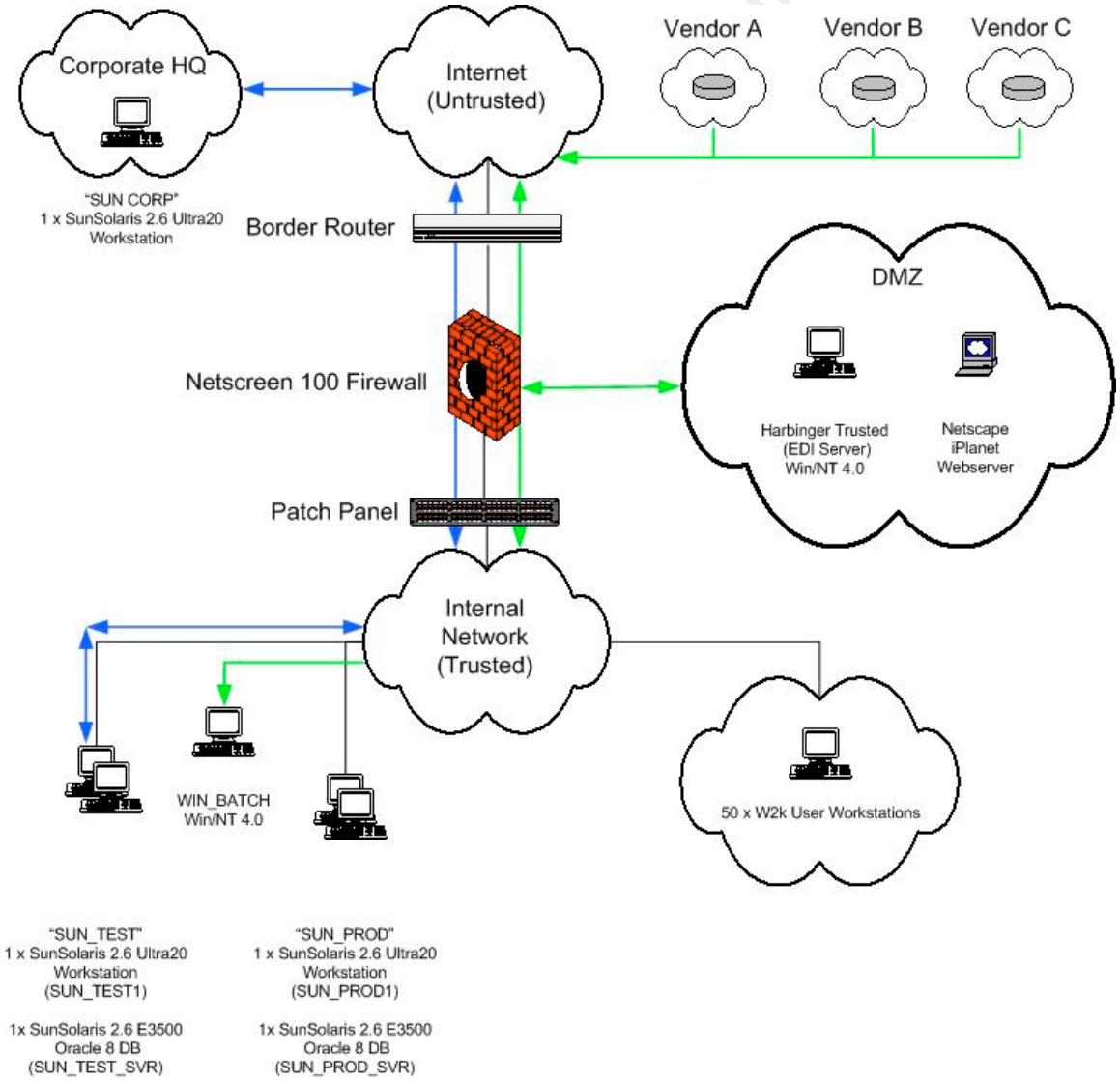
http://www.securiteam.com/windowsntfocus/6U00B2000A.html

Part 2 – The Attack

Diagram of Network

Figure 1 is a simple representation of the network of the time when the incident occurred. It is very simple layout and may be similar to networks that many other small and medium sized businesses currently operate. With thin IT budgets the productivity requirement usually far outweighs the security requirement. The "bare minimum" usually suffices to organizations that exercise enough due diligence to maintain operations and avoid lawsuits, but still choose to operate vulnerable and risk valuable data on a daily basis.

Figure 1



Network Description

The network depicted in Figure 1 is a representation of a network that I inherited. The backbone connection to the Internet was a Cisco 3600 Series Router (IOS version unknown) with a very liberal ACL (Access Control List) rule set.

A Netscreen Firewall running ScreenOS v1.66 is configured with an explicit deny all ACL from any untrusted sources to any inbound destinations. The logging function of the firewall, coupled with an alert feature, would send emails to the administrator group's inbox. This process served as our "poor mans" Intrusion Detection System (IDS).

The Demilitarized Zone (DMZ) housed a Netscape iPlanet Win/NT 4.0 Web server. Additionally, a second Win/NT 4.0 Server (Harbinger Trusted) provided Electronic Data Interchange (EDI) where several vendors would deposit encrypted customer records to be processed against trusted billing databases.

Although in this instance the EDI service was located in the DMZ, a FTP proxy was still required to enable customer records to be sent from a Harbinger Trusted EDI server to WIN_BATCH a trusted Win/NT 4.0 Server. Once WIN_BATCH received EDI data from the Harbinger Trusted system, the customer records were populated to Oracle 8 databases via homegrown Perl scripts.

However, additional exceptions were implemented for systems located at Corporate HQ (Headquarters). A team of offsite Oracle Database Administrators and Sr. UNIX Administrators would require unfettered access into the local test database to run various test Perl scripts against the SUN_TEST and SUN_PROD environments. Of all ACL's in the firewall, this rule set was the most liberal. It would simply allow all services from the IP address of the external port of the firewall at Corporate HQ to the inbound IP address of the firewall at our satellite office.

50 user Windows 2000 Professional workstations were required for the sales department, billing department, human resources, etc. Anti-virus engines were installed on each workstation and were centrally managed by a corporate anti-virus server.

Below is a brief description of each "Sun environment":

SUN_TEST: A testing environment consisting of:
One SunUltra20 Workstation (SUN_TEST1) running Sun Solaris 2.6.
One Sun Enterprise 3500 (SUN_TEST_SVR) running Sun Solaris 2.6 and Oracle 8.

SUN_PROD: A production environment consisting of:
One SunUltra20 Workstation (SUN_PROD1) running Sun Solaris 2.6.
One Sun Enterprise 3500 (SUN_PROD_SVR) running Sun Solaris 2.6 and Oracle 8.

SUN_CORP: A testing environment consisting of:
One SunUltra20 Workstation (SUN_CORP) running Sun Solaris 2.6.

Protocol Description

The protocols used to execute the sadmind/IIS worm are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). The services used are as follows:

111/TCP & UDP – Sun Remote Procedure Call (sunrpc)
"Sun RPC portmapper / rpcbind"
Sun Remote Procedure Call (sunrpc) was originally designed for client /server functions with the Sun Network File System (NFS). Because Sun RPC is not a transport protocol it is reliant on portmapper / rpcbind. Portmapper / rpcbind are two names for the same type of service. Once sunrpc initializes and makes an RPC call to a specific program number, sunrpc then contacts the local portmapper daemon to figure out what port number RPC packets should be sent to. Thus, Sun RPC dynamically maps the RPC program numbers to actual transfer specific TCP or UDP port numbers.

80/TCP – Hyper Text Transfer Protocol (HTTP)
Hyper Text Transfer Protocol is the protocol a web server uses to communicate with a web browser such as Internet Explorer or Netscape.

600/TCP – sadmind/IIS worm (/dev/cuc/nc)
The sadmind/IIS worm may attempt to execute a shell with root level permissions on the Solaris system. When executed the shell listens on port 600/TCP as a backdoor into the compromised host.

How the Exploit works

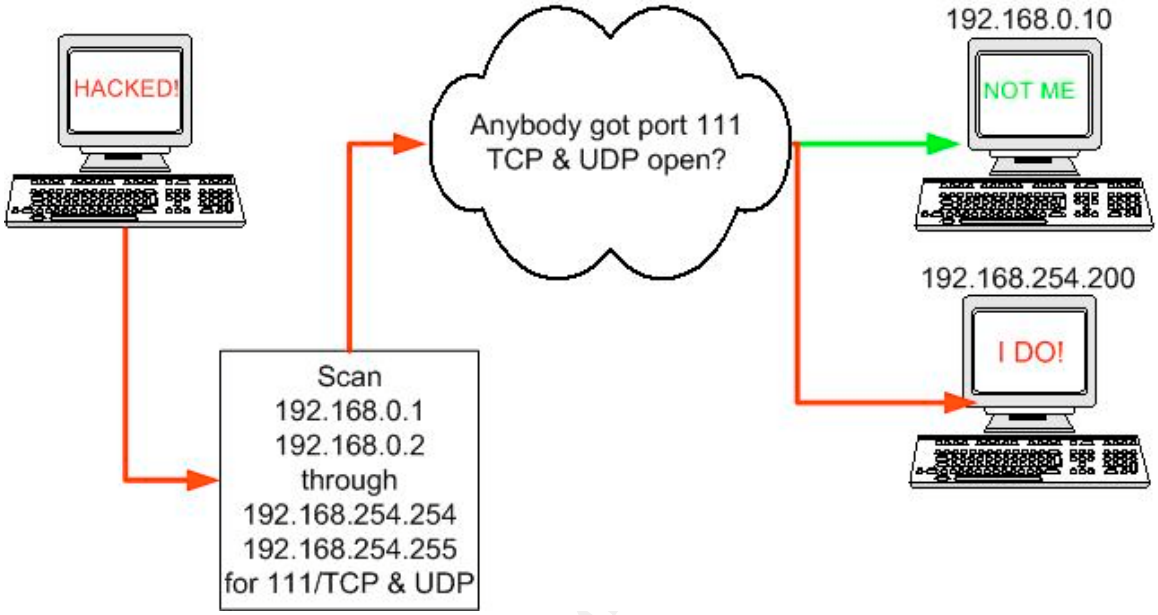
As we found out earlier the sadmind/IIS worm works in several stages by exploiting multiple vulnerabilities. Bill Hayes at the University of Nebraska Lincoln [1]describes the individual steps the worm takes as it begins to propagate throughout a network (as shown below):

Step 1.
Initially a compromised Solaris system will begin by executing /dev/cuc/start.sh. This file is appended to the /dev/rc.d/S71rpc file to ensure the worm is started when the system initializes. At this time the /dev/cuc/start.sh executes the worm without requiring a system reboot. The start.sh script then selects a random class "b" network looking for additional vulnerable Solaris systems. Afterward the /dev/cuc/ranip.pl begins generating random IP addresses within the random class "b" network. Once this process begins the worm checks all possible IP

[1] http://www.unl.edu/security/virus_alerts/sadmind.htm

address with in a class “b” network, e.g. a.b.0.1 to a.b.0.2 through a.b.254.254 to a.b.254.255. All of the addresses that fall within the generated range will be tested for a running Sun RPC service (port 111 TCP/UDP).

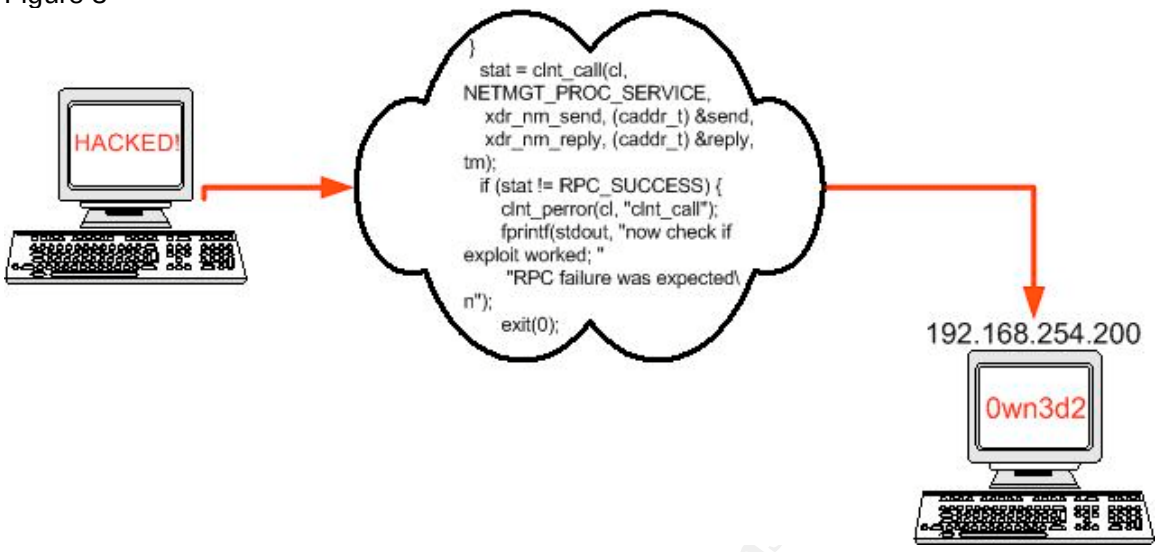
Figure 2



Step 2.
When a system running the Sun RPC process is located, the worm uses the /dev/cuc/grabb binary to test if sadmind service is running. If sadmind is detected the /dev/cuc/brute binary determines the platform type. Depending if the hardware is a Sparc system or X86, the worm exploits the appropriate platform specific vulnerability and attempts to employ a sadmind buffer overflow against the amsl_verify when requesting the NETMGT_PROC_SERVICE in vulnerable versions of sadmind.
Alfred Huger and Marcy Abene who provided information detailing this exploit at Securiteam.com, indicate that the initial sadmind buffer overflow is successful once a large buffer is sent to the NETMGT_PROC_SERVICE request (called via clnt_call()). They state that it is then possible to overwrite the stack pointer and execute arbitrary code, and the actual buffer in question appears to hold the client's domain name. The overflow in sadmind takes place in the amsl_verify() function. Because sadmind runs as root any code launched as a result will run as with root privileges, therefore resulting in a root compromise.[2]

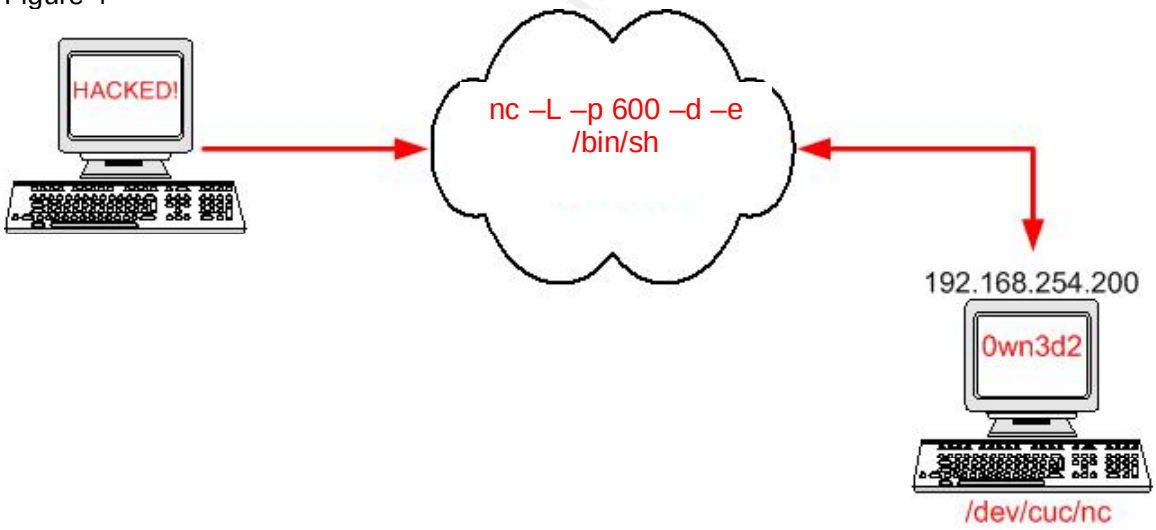
[2] http://www.securiteam.com/exploits/3P5Q1Q0QAO.html

Figure 3



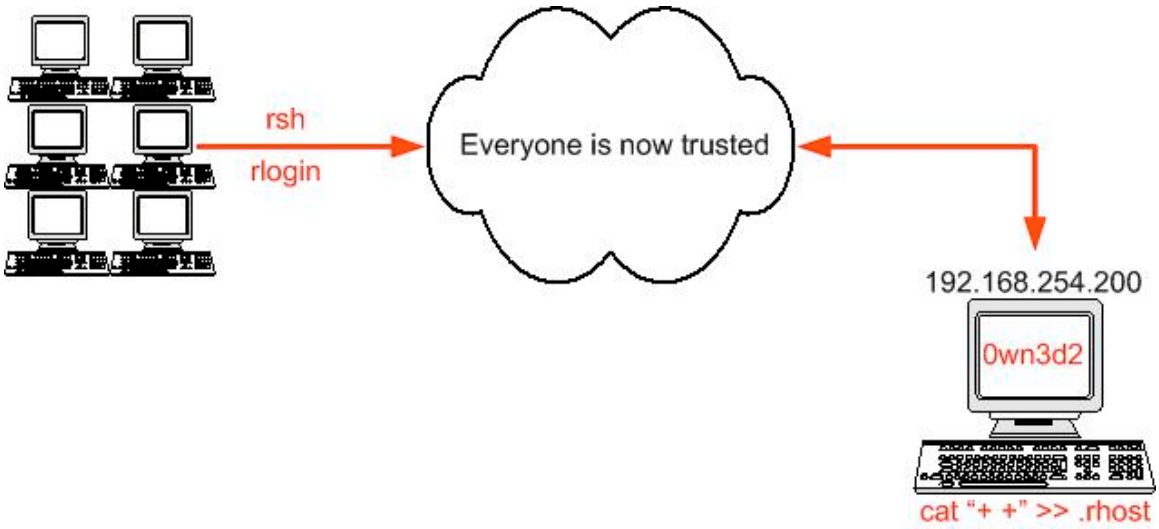
Step 3.
Figure four indicates that the worm will then execute /dev/cuc/nc which creates a backdoor root level shell listening on port 600/TCP of the compromised machine.

Figure 4



Step 4.
In Figure 5 the worm appends "+ +" to the .rhost file, subsequently nullifying any remote authentication in remote connections such as rsh or rlogin. The attacking Solaris system then copies the worm's payload via "rcp" to the vulnerable Solaris system and executes the worm. The infection cycle continues as the victim machine becomes an attacker and begins to actively search for additional vulnerable Solaris hosts.

Figure 5



Step 5.
A list of IP addresses of compromised systems are kept in two separate log files in /dev/cub/ directory. The dev/cub/result.txt file catalogs the IP addresses of all compromised IIS servers and the dev/cub/sadminhack.txt file contains the IP addresses of all compromised Solaris systems.

Step 6.
The worm then attempts to download and install Perl 5.005 from a Chinese FTP site (ftp://bak-px.online.sh.cn).

Step 7.
Now that a Sun Solaris system has been successfully compromised, the worm changes gears and now has a staging area to conduct further attacks against Microsoft IIS 4.0 / 5.0 web servers. The worm begins its second stage of attacks by executing the /ev/cuc/grabbb binary to identify vulnerable IIS systems. Once a list of systems has been generated the worm executes the /dev/cuc/uniattack.pl perl script to attempt to exploit an Extended UNICODE Directory Traversal exploit.

First, let's dig a little deeper to fully understand this second leg of the sadmind/IIS worm and how it is used to modify web content. By default Microsoft IIS 4.0 and 5.0 provide the ability for web administrators to place executable files in a set of default folders on the web server for visitors to execute when they visit the site. However, when malicious users run a specially crafted GET request using extended UNICODE representations for "/" or "\" against the vulnerable IIS server, read, write and execute privileges of the IUSR_*systemname* are granted to files and folders that sit on the same logical drive that contain the default web folders, (i.e. the C:\inetpub). The IUSR_*systemname* serves as a default

anonymous user account, however it is granted “Everybody” and “User” permissions.

Here are a few examples of a specially crafted GET and HTTP requests:

GET scripts/..%c0%af../winnt/system32/cmd.exe?

http://target_IP_address/scripts/..%c0%af../winnt/system32/cmd.exe?

The *..%c0%af..* portion of this string specifically addresses the Extended UNICODE vulnerability. The “/” was represented by “..%c0%af..”, therefore the vulnerable IIS system would interpret this portion of the string as “../..”
By default IIS includes executable directories in the web folder, as well as the *scripts* directory that is used in this example.
So as we see in these examples, if the C:*/winnt/system32* directory is located on the same logical drive as the target hosts vulnerable *scripts* directory, the attacker who exploits this vulnerability should be able to obtain and command line interface (cmd.exe).

Several other examples of Extended UNICODE characters that can be used as well are:

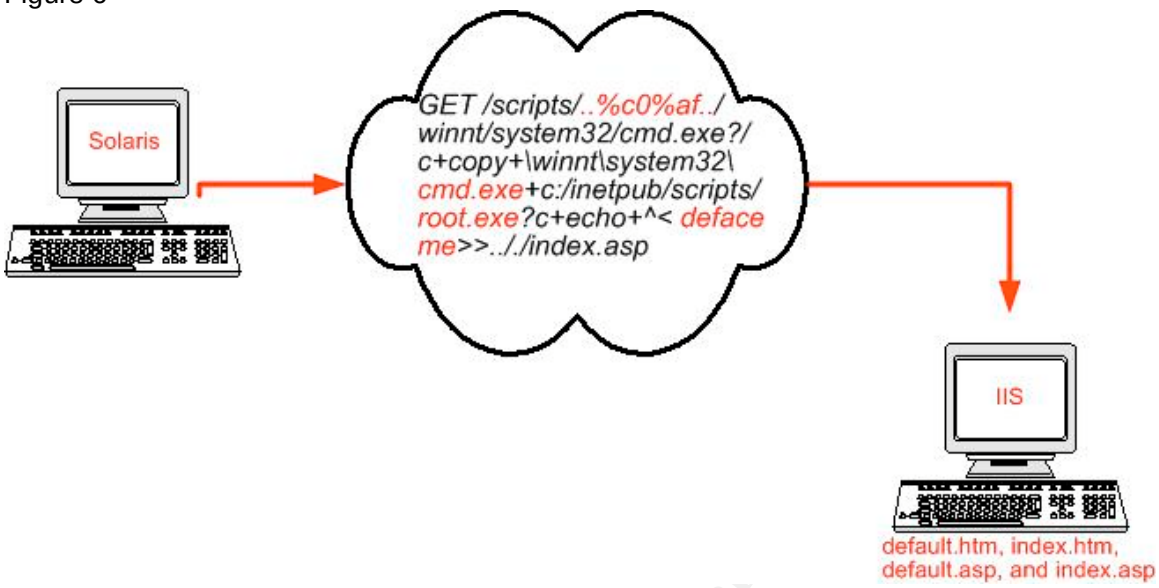
http://target_IP_address/scripts/..**%c1%1c**../winnt/system32/cmd.exe?/c+dir
http://target_IP_address/scripts/..**%c0%9v**../winnt/system32/cmd.exe?/c+dir
http://target_IP_address/scripts/..**%c0%qf**../winnt/system32/cmd.exe?/c+dir
http://target_IP_address/scripts/..**%c1%8s**../winnt/system32/cmd.exe?/c+dir
http://target_IP_address/scripts/..**%c1%9c**../winnt/system32/cmd.exe?/c+dir
http://target_IP_address/scripts/..**%c1%pc**../winnt/system32/cmd.exe?/c+dir

*http://target_IP_address/**msadc**/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
*Note: By entering msadc in the URL, it is possible to traverse from the root web directory to other directories that are not normally available within standard HTTP requests.

Step 8.

In figure 6 the IIS portion of this attack begins with the compromised Solaris system sending a specially crafted GET request that copies the cmd.exe from the c:/winnt/system32 folder to c:/inetpub/scripts and renames cmd.exe to root.exe. The root.exe will now carry the date of modification for cmd.exe; this will be a helpful clue later in determining the date of compromise.

Figure 6

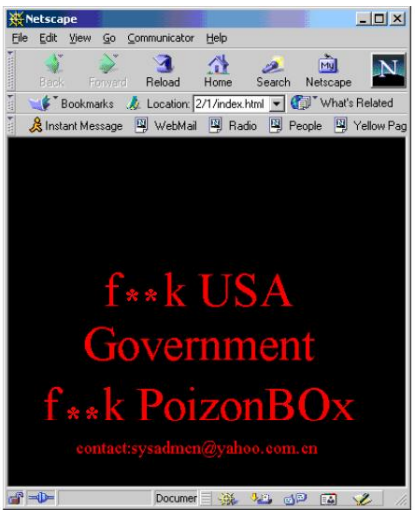


The compromised Solaris system initially moves /winnt/system32/cmd.exe to c:/inetpub/scripts/root.exe, the worm then passes the following GET request to the vulnerable Microsoft IIS system. Where <*deface me*> is the html code for the web defacement (see Figure 7):

GET /scripts/..%c0%af../winnt/system32/cmd.exe?/ c+copy+\winnt\system32\cmd.exe+c:/inetpub/scripts/root.exe?c+echo+^< deface me>>../../index.asp

Figure 7 is a depiction of the defaced web page once the sadmind/IIS worm had successfully exploited the extended UNICODE vulnerability. In this example the offensive message content has been edited from the original defacement.

Figure 7



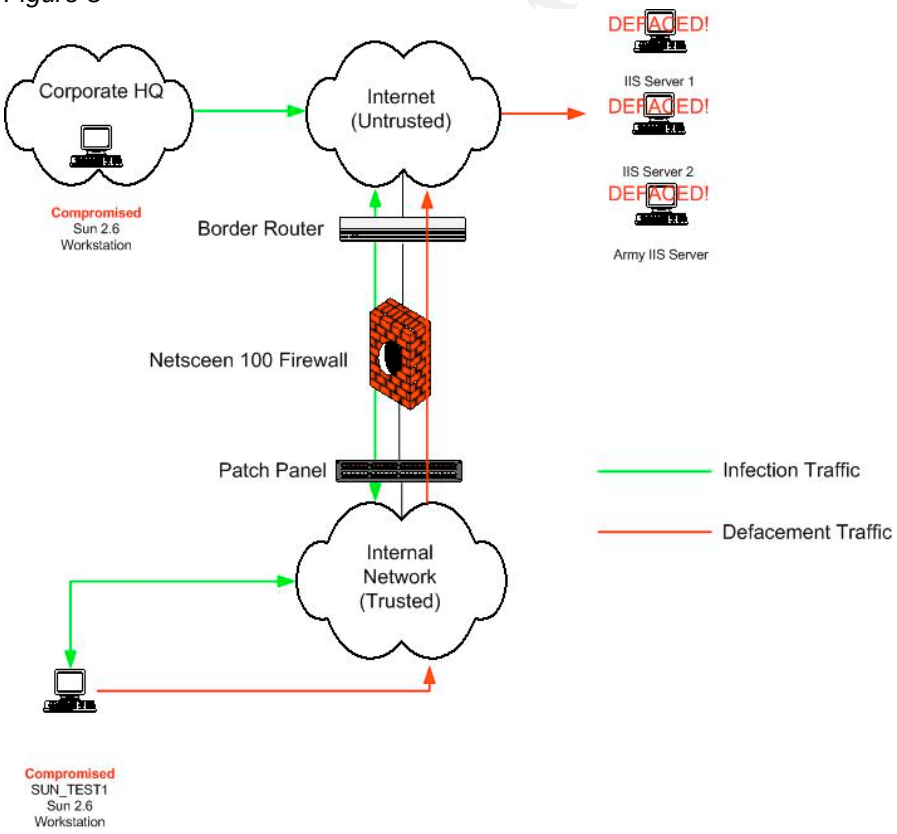
Original graphic Courtesy: Kaspersky Lab[3]

[3] http://www.avp.ch/avpve/worms/net/sadmind.stm

Diagram of the attack

Figure 8 is a diagram of what most likely occurred that day. As Lenny corralled all the contractors that were onsite to begin his “hacker interrogations” a sadmind/IIS worm had been wreaking havoc inside our network. A compromised Sun Solaris 2.6 workstation located at Corporate HQ had traversed a misconfigured firewall rule set that allowed SunRPC traffic (Port 111 TCP/UDP) from Corporate HQ into our SUN_TEST environment. SUN_TEST1 was the only Sun Solaris system running the sadmind process, all other Sun Solaris systems had the sadmind process commented out of the inetd.conf file, which nullified any subsequent worm propagation attempts.
Once the worm shifted into the IIS extended UNICODE attack phase, several IIS servers (version unknown) belonging to outside third parties were attacked and defaced from the SUN_TEST1 system. Thus the numerous phone calls from angry web administrators. Shared or connected networks are common requirements for organizations that operate business-to-business (B2B) commerce. Very often organizations make the mistake of inherently trusting other organizations, vendors or client networks. Networks that are running systems and services they have never seen and have never configured. However, the most disturbing is these networks are run by engineers and administrators they have never met. These users, engineers and administrators that are now considered “trusted” because of a single Access Control List.

Figure 8



In such a scenario network accreditations should be conducted for each network involved. Additionally, memorandums of agreements should be in place prior to committing to a B2B network configuration. If and when an incident occurs all of the chiefs and indians have been identified and everyone has a function and knows what their responsibilities are.

Signature of the attack

There are many ways to detect a sadmind/IIS attack, syslog messages, Snort Intrusion Detection Sensors as well as the unconventional methods such as having another organization give you call to complain. Then there is my personal favorite, opening a web browser to discover that you have a new web page. However, we will begin looking at the individual systems and analyzing the small clues the worm leaves on compromised hosts as it propagates from network to network.

Compromised Solaris Signatures

Initially the compromised Solaris systems /var/adm/syslog file may look similar to the output below:

May 7 02:40:01 carrier.example.com inetd[139]: /usr/sbin/sadmind: Bus Error - core dumped
May 7 02:40:01 carrier.example.com last message repeated 1 time
May 7 02:40:03 carrier.example.com last message repeated 1 time
May 7 02:40:06 carrier.example.com inetd[139]: /usr/sbin/sadmind: Segmentation Fault - core dumped
May 7 02:40:03 carrier.example.com last message repeated 1 time
May 7 02:40:06 carrier.example.com inetd[139]: /usr/sbin/sadmind: Segmentation Fault - core dumped
May 7 02:40:08 carrier.example.com inetd[139]: /usr/sbin/sadmind: Hangup
May 7 02:40:08 carrier.example.com last message repeated 1 time
May 7 02:44:14 carrier.example.com inetd[139]: /usr/sbin/sadmind: Killed

**syslog file example provided by: http://www.cert.org/advisories/CA-2001-11.html*

In the syslog example above, provided by the CERT (www.cert.org) we can see that *carrier.example.com* is checking to see if /usr/sbin/sadmind is running. A Segmentation Fault occurs where the worm attempts to access memory that is not allocated to it thus “core” or memory dumps with a segmentation violation error.

The presence of /dev/cuc or /dev/cub directories and the files listed below are also indicative of a sadmind compromise.

/dev/cuc/brute
/dev/cuc/cmd1.txt
/dev/cuc/cmd2.txt
/dev/cuc/grabbb
/dev/cuc/index.html
/dev/cuc/nc
/dev/cuc/pkgadd.txt
/dev/cuc/sadmin.sh
/dev/cuc/sadmindex-sparc
/dev/cuc/sadmindex-x86
/dev/cuc/start.sh
/dev/cuc/time.sh
/dev/cuc/uniattack.sh
/dev/cuc/uniattack.pl
/dev/cuc/ranip.pl
/dev/cuc/wget

System processes as well as the processes of the sadmind/IIS worm may be listed out in a terminal with the "ps" command. Commands like "netstat" will also prove useful in determining listening or idle services to include existing connections. Below is a list of processes of the sadmind/IIS worm's scripts:

/dev/cuc/grabbb -t 3 -a .yyy.yyy -b .xxx.xxx 111
/dev/cuc/grabbb -t 3 -a .yyy.yyy -b .xxx.xxx 80
/bin/sh /dev/cuc/sadmin.sh
/bin/sh /dev/cuc/uniattack.sh
/bin/sh /dev/cuc/time.sh
/usr/sbin/inetd -s /tmp/.f
/bin/sleep 300

If found the /dev/cuc/nc is another process that should receive additional attention per your incident handling process standard operating procedure. If there is not already an established connection, TCP port 600 may have a backdoor root level shell listening. Additionally as stated before, the presence of root.exe is also a very good hint that the system has been compromised.

Finally once /dev/cuc/result.txt file collates 2000 compromised IIS systems, any instance of index.html on the compromised Solaris system will be replaced with the /dev/cuc/index.html thus displaying the same defaced webpage that the compromised IIS systems display.

Snort Intrusion Detection System Signatures for sadmind/Worm Attack

The following is a Snort signature that was provided at http://www.whitehats.com. When this rule is applied Snort will inspect each inbound UDP packet with a destination port of 111 UDP. The rpc option checks for all processes calling100232 (the sadmind application) that are trying to pass content matching 018788.

alert UDP $EXTERNAL any -> $INTERNAL 111 (msg: "IDS20/rpc_portmap-request-sadmind"; rpc: 100232,*,*; content: "|018788|"; classtype: info-attempt; reference: arachnids,20;)[4]

[4] http://www.whitehats.com/info/IDS20

Compromised IIS Signatures

By checking the c:\winnt\system32\logfiles\W3SVC1 directory on an IIS host will show additional evidence of a sadmind/IIS attack. The log file will look similar to the following Microsoft IIS log sample provided CERT (www.cert.org).

```
2001-05-06 12:20:19 10.10.10.10 - 10.20.20.20 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir 200 -
2001-05-06 12:20:19 10.10.10.10 - 10.20.20.20 80 GET /scripts/../../winnt/system32/cmd.exe
/c+dir+..\ 200 -
2001-05-06 12:20:19 10.10.10.10 - 10.20.20.20 80 \
      GET /scripts/../../winnt/system32/cmd.exe /c+copy+\winnt\system32\cmd.exe+root.exe 502 -
2001-05-06 12:20:19 10.10.10.10 - 10.20.20.20 80 \
      GET /scripts/root.exe /c+echo+<HTML code inserted here>../../index.asp 502 -
```

On compromised windows hosts the worm will replace or create default.htm, index.htm, default.asp, and index.asp with the defaced page to ensure that the page is posted. Instances of these defaced pages have been found in the following directories:

c:\
c:\inetpub
c:\inetpub\mailroot
c:\inetpub\scripts
c:\inetpub\wwwroot_private
c:\inetpub\wwwroot_vti_log
c:\program files\common files\system\msadc

By looking for file names with “default.htm or default.asp” and “index.html or index.asp”, the windows “find” utility will locate the defaced web page files. Another way of locating infected files would to be to search for files with a creation or modification date matching the date that the root.exe file was created.

Snort Intrusion Detection System Signatures for IIS UNICODE Exploit

The following Snort signatures were provided at www.whitehats.com. When applied these Snort rules inspect all inbound web traffic on port TCP port 80. The A+ flag checks to see if any packet has an ACK flag set in addition to any other flags. The Uniform Resource Identifier Content or uricontent allows Snort to search the URI section of a request rather than the entire packet payload. The nocase option will specify that case sensitivity is not applicable.
In the examples provided |25|c1|25|1c and |25|c0|25|af are translations for %c1%1c and %c0%af. Additional rules with matching uricontent should be implemented for all extended UNICODE characters to properly detect IIS UNICODE exploits.

alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS432/web-iis_http-iis-unicode-traversal"; flags: A+; uricontent: "..|25|c1|25|1c"; nocase; classtype: system-attempt; reference: arachnids,432;)

alert TCP $EXTERNAL any -> $INTERNAL 80 (msg: "IDS433/web-iis_http-iis-unicode-traversal-optyx"; flags: A+; uricontent: "..|25|c0|25|af"; nocase; classtype: system-attempt; reference: arachnids,433;)[5]

How to protect against the sadmind/IIS

If your organization is running Solaris systems we can begin with simply determining if the /usr/sbin/sadmind is a required process. If the process is required, adding authentication requests with a –S 2 flag will give an additional layer of security to this process. To do this the/etc/inetd.conf file will need to be edited.

100232/10 tli rpc/udp wait root /usr/sbin/sadmind sadmind -S 2
**Note this change will not take place until inetd.conf receives a signal hang-up (kill –HUP procid or init 6 for reboot)*

If the sadmind is not a required process then the service can be excluded during system startup by simply commenting out the following line in the /etc/inetd.conf.

100232/10 tli rpc/udp wait root /usr/sbin/sadmind sadmind\
**Note this change will not take place until inetd.conf receives a signal hang-up (kill –HUP procid or init 6 for reboot)*

Additionally, vendor patches are provided at the following website:

http://sunsolve.sun.com/pub-cgi/retrieve.pl?doctype=coll&doc=secbull/191&type=0&nav=sec.sba

If the IIS 4.0 or 5.0 services are a required part of your organizations infrastructure, Microsoft recommends several fixes for the IIS extended UNICODE exploit:

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-044.asp

http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS00-078.asp

Administrators may also download the URLScan Security Tool:

http://www.microsoft.com/windows2000/downloads/recommended/urlscan/default.asp

[5] http://www.whitehats.com/ids/index.html

The process of Securing Windows IIS is another paper in itself. The SANS GIAC Certified Windows Security Administrator (GCWN) offers a specific block of instruction on Securing Microsoft IIS.
However, to guard against the sadmind/IIS worm Web Administrators may also do a combination of the following steps to prevent a successful sadmind/IIS attack.

- Reviewing the permissions of the IUSR_*systemname* account, or disabling it if is not required.
- Partition the system so globally accessible files are stored on a separate partition than critical system files.
- Install recommended patches.

Part 3 – The Incident Handling Process

Unfortunately, in this scenario there was not an Emergency Action Plan in place to provide a roadmap to recovery should an event or incident occur. The organization I worked for had changed hands several times and many people had come and gone since it was established. Many lessons were learned through failures during my time at this organization. From non-existent redundant power sources to faulty air conditioning and humidity systems poor planning had left its mark at this organization.

Preparation

It is quite simple to see how an unprepared organization is at a disadvantage from the beginning of incident, unless a battle tested and refined Emergency Action Plan is developed and that the individuals that will handle the incident are familiar with the Incident Handling Process.
In this scenario security tasks were not centralized, and multiple administrators with limited lines of communication, as well as instances of one administrator making a system change and another administrator coming along and changing the configuration back made for a very dynamic atmosphere.
The only person that had been pegged to lead should any crisis occur was Lenny, and that was just an appointment by default not by policy.
However, we saw how easy he was willing to sacrifice the organizations well being on the behalf of his image of a competent CIO by downplaying and keeping the incident from corporate leadership.
Very little preparation had been conducted prior to my employment. A corporate edition anti-virus server ensured that viruses and trojans were identified and contained on end user workstations.
A Netscreen 100 firewall was deployed and configured and an independent security consultant was even hired when security issues were raised.
However, many things had changed prior to my first week on the job. Many of these systems had been forgotten, administrators had come and gone, tasks had been ignored and licenses had expired.

So one of my initial tasks were to secure the both Sun Solaris environments. So I began by reading a few white papers about UNIX security and hardening Solaris systems.
I conducted simple Jr. Administration tasks such as reviewing system accounts, changing root passwords and reviewing and editing individual system configuration files to include /etc/inetd.conf file to enable used services and disable unused services in each of the SUN environments (TEST & PROD). To include editing all of the .rhosts files to require a username and password for all remote authentication attempts. This effort may have helped but it was not complete and comprehensive. At one point I even recommended researching and deploying an anti-virus engines for both Sun Solaris environments.
An outsourced independent security consultant was in charge of maintaining the Netscreen 100 firewall and the ACL's required between cooperate headquarters and our remote office. Anti-Virus engines were only enabled on the individual Windows 2000 workstations and the Corporate Anti-Virus Server had an expired license. So at the time of this incident there were no anti-virus countermeasures deployed on the Solaris systems and up to date signatures were not making it down to the individual workstation.
Needless to say this organization was not prepared to handle an incident, and in some cases once events were identified they were often dismissed.
Properly preparing our satellite office would have included implementing policies that specified what system services would be used for specific tasks, establishing accreditation baselines would have ensured that all arms of the organization were configured the same way or would allow other organizations to plan accordingly if there was ever a deviation from that baseline.
Memorandums of agreement would have also been developed and employed so that predefined leadership would have been selected and in place prior to any incident occurring. Leadership could then identify senior staff and task them with developing and Emergency Action Plan (EAP).
This process would have eventually included a comprehensive review of all firewall ACL's against services required during daily operations.
For example, the off site administrators at Corporate HQ would usually TELNET into our organization 99% of the time. Most of their time was spent testing Perl scripts or maintaining the systems. There were not any shared drives or mount points on any of the Sun Solaris systems and very rarely did they need to create one to connect to. The Netscreen 100 was pretty much a big expensive NIC between Corporate HQ and our satellite office.
Establishing a policy that all inter-organization systems administration would be though an encrypted means would have required that we install a VPN (Virtual Private Network) or had used SSH (Secure Shell) on each systems. The firewall ACL could have been changed from:

"Allow ANY service from the external_IP at Corporate HQ to our internal_IP of our Netscreen 100"

To

"Allow TCP/UDP port 22 from SUN_CORP_IP to SUN_TEST1_IP"

This change would have allowed the administrators to test their Perl scripts and to conduct remote administration tasks in a Secure Shell, but better yet would have mitigated unnecessary risk by blocking services our satellite office didn't even use.
Additionally, deploying a Snort Intrusion Detection System would have been a key step to the preparation phase. The data gathered by the IDS would assist later in the early stages of the identification phase, this free IDS would have been quite cost effective to a small organization like our satellite office with limited resources. Time and money spent identifying the suspicious network traffic that we were experiencing could have been limited.

Identification

Several days prior to being contacted directly by the US Army National Guard we had received numerous calls from individuals claiming to be web administrators or owners of web sites all of which stated that defacements had occurred on their web sites and that the defacements had originated from our organization. These claims were sent over to Lenny and the onsite lawyer, where later in a meeting they had dismissed the claims as social engineering attempts. However, both Lenny and the onsite lawyer refused to take any calls from these victims.
I had informed Lenny that the increased alarms from the firewall indicated numerous traceroute attempts and although, traceroute activity was not uncommon, the volume of activity coupled with the numerous phone calls should have been identified as an indicator and taken more seriously when validating these defacement claims and then investigated further.
If a Snort IDS had been deployed during the preparation phase this attack and if the signatures were kept up to date that IDS would have most likely alarmed on either the sadmind request (TCP or UDP) or the Extended UNICODE directory traversal phase of the attack. Even though a predefined signature for the actual sadmind/IIS worm would not have been available yet, the signatures for the individual known exploits were months old at the time of this incident.
If the organization had deployed an Incident Response team the EAP would have used by the first responders so they could take initial steps identifying these events, as well as determining if these events were enough to declare an incident and decide whether or not to investigate.
Either way lines of communication would have been opened up between Corporate HQ and our satellite office, the technical staff at Corporate HQ would

have been able to react and verify our claims at their end, by checking for anomalies on their Solaris system (SUN_CORP) or their router logs for example. Instead of blowing off the people that called and claimed to have defaced web sites, a brief telephone conversation would have quickly identified the indicators of a sadmind/IIS worm compromise rather than assuming it was a disgruntled employee or contractor using company resources to stage web attacks.
An open source Ethernet sniffer such as Ethereal or the "snoop" command on SUN_TEST1 could have been used to collect just web traffic, or all traffic to and from SUN_TEST1.
By getting eyes on the system and simply verifying that SUN_TEST1 was actively searching for vulnerable IIS servers would have been all the evidence needed to confirm or deny whether SUN_TEST1 was indeed compromised. Since SUN_TEST1 was a testing box for Perl scripts, it was never used to browse the web. Information gained from these steps could have saved time and resources to identify a true compromise.

Containment

In this scenario the infected SUN_TEST1 was left connected to the network. The compromised system was never contained per se.
Inexperience and ignorance of the Incident Handling Process kept this incident from being properly contained. The administrators and leadership at Corporate HQ were not warned of a compromised system so that appropriate steps dealing with systems that had trust relationships could be taken.
The system was never taken off of the network or isolated from the rest of the network in anyway.
In hindsight, since the system that was compromised was a testing system, the organization could have determined through risk analysis that they risked less by removing the system completely from the network than leaving it online to potentially attack. Business continuity and daily operations did not hinge on the uptime of a single testing system.
If this scenario were to occur again, I would recommend completely removing this system from the network or moving it to a closed segment and copying the disk via either a third party application or a simple Solaris command "dd" to duplicate the compromised disk to another disk or "ufsdump" to a magnetic medium.
A copy of the compromised SUN_TEST1 could have been made available if the legal department needed outside expertise to analyze the disk should one of the organizations with a defaced web site chose to take legal action.
Separate copies of a compromised SUN_TEST1 could be made and loaded on a standalone system, where the system could then be analyzed to see if any additional software was installed on the system or if a backdoor had been used via a listening shell on TCP port 600.
The Netscreen 100 firewall ACL's would be reviewed again and TCP port 600 and TCP/UDP 111 from all external sources would have been blocked until the compromise was eradicated from the network.

The administrators and engineers at Corporate HQ that had systems with trust relationships would have been contacted to inform them of a compromise and that all applicable systems needed to be checked for applicable signatures of infection.
At the time of the compromise very little was known about this worm, just imagine if an attacker had used the TCP port 600 to backdoor into the system, additional attacks could have been staged against our valuable customer database. Thousands of credit card numbers and the integrity of our organization were at risk just by leaving a testing box online.
To me it seemed as if leadership was attempting to hide the fact that a serious incident had occurred, therefore I was in a difficult position. I still needed to follow Lenny's direction, but I also needed to protect the rest of the organization as well as myself.
By failing to properly document the steps I took, the organization had nothing to go on should any of these instances of defacement go to a legal setting.
When things go wrong, desperate leaders sometimes find it easier to blame someone else rather than their past judgments. By failing to properly document everything, from times and dates of log files, to commands I entered into SUN_TEST1, conversations or the bullets from meetings, anything I found noteworthy would have been helpful should fingers start to have pointed my way or if evidence had been needed for the courts.

Eradication

Prior to working at this organization in particular, I spent time working as a defense contractor for the Defense Information Systems Agency (DISA). Somewhat familiar with the Department of Defense Computer Emergency Response Team (DoD CERT) mission, I found it quite strange to be contacted directly by the web administrator of the actual National Guard Unit that had been compromised. Usually the DoD CERT is contacted directly from the compromised organization or work with the service specific CERT as an intermediately.
As I followed up with the idea that it was some type of social engineering attempt I contacted the DoD CERT and spoke with an incident handler to see if they could corroborate Staff Sergeant Smith's claim with the service specific CERT. After briefly explaining the events that were occurring on my network, the incident handler informed me that SSG Smith may have meant well but should not have contacted me directly. After describing the symptoms we had been experiencing on our network, the incident handler later referred me to a sadmind/IIS link on CERT www.cert.org, and gave me some helpful hints on how to identify and remove the traces of infection.
I began by collecting all the information regarding the worm as I could. After reviewing the collected information I downloaded the appropriate vendor patches from the Sun Microsystems website.
Later I spoke with Lenny and informed him that it was very unlikely that we had a hacker defacing websites from within our organization. I showed him the

information regarding the sadmind/IIS worm and the recommended using the removal instructions. Lenny then directed me to patch SUN_TEST1 and monitor the system prior to deploying the patches to the SUN_PROD environment. After patching the SUN_TEST1 system I began checking the system for the existence of /dev/cuc/ and /dev/cub/ directories.
Once the /dev/cuc/ and /dev/cub directories were discovered they were immediately removed with the "rm –r" command by direction of the CIO who was hovering over my shoulder.
The /etc/inetd.conf was reviewed and edited per recommendations from the vendor, and the system was then rebooted. During this step I made a very strange discovery, the inetd.conf had been changed since my initial lockdown. I would later find out that the AdminSuite package had been re-installed remotely by Administrators at corporate headquarters.
By failing to copy the files I deleted or document their existence, my organization had nothing but our memories to go on should this incident had gone into a legal setting. All Lenny and I were interested in at the time was removing this infection from the network. Without any formal training of the eradication phase of the incident handling process, valuable data and evidence was destroyed.

Recovery

Later that afternoon all of the Solaris systems were eventually patched with the recommended vendor patches, per Lenny's direction. The modified .rhosts was edited to require authentication and inetd.conf file was edited and the sadmind daemon was commented out. Once the /dev/cub and /dev/cuc directories were removed the system was rebooted.
Once the systems were back online I reviewed the syslog entries and /var/adm/messages files to ensure that everything was running smoothly. The current running processes were reviewed and monitored with "ps –elf" and "netstat -an" combinations.
I continued to review all of the Solaris systems in both test and production environments, I later informed Lenny that SUN_TEST1 was the only system on our network that seemed to had been compromised, and that the Sr. Administrators at Corporate HQ should review their Solaris systems as well. I added that the independent security consultant should be contacted and informed of the incident to make appropriate firewall changes. Lenny assured me he would handle everything else, but asked me to review the system periodically and keep an eye on it for the next couple weeks.
For the next several weeks I continued to monitor both Sun environments for any system anomalies or instances of sadmind/IIS worm re-infection.
One of the largest blunders, and something I still have trouble understanding was the fact that the Netscreen 100 firewall ACL's were never changed. Even after learning more about the worm I recommended to management that we review all the ACL's especially inbound TCP & UDP port 111. My requests fell on deaf ears; I had even explained that Network File Sharing (NFS) or remote mount points were not even used from corporate headquarters to our satellite office.

The Sun RPC service was not needed in any of the Business Applications that we used.
Eventually I was able to convince management into researching VPN (Virtual Private Network) solutions. However after a few weeks everyone fell back into the usual state of complacency, and eventually the sadmind/IIS compromise was shrugged off.

Lessons Learned

A meeting regarding this incident occurred over a lunch between Lenny and I the next day.
There was no documentation indicating that an incident ever occurred on the network. The fact that the sadmind/IIS worm seemed to be successfully removed from SUN_TEST1 was all Lenny and I needed to forget about this incident.
However, this practical has contrasted what my organization and I did that day, to what my organization and I should have done or considered.
If I were asked to provide input for a lessons learned meeting regarding this incident today I would begin highlighting where when leadership lost composure, in any successful operation knowledgeable leadership and sound decisions are key to success. As scary as it is to have an unknown person or program operating in your network, staying calm and making sound decisions is the first step towards success when handling an incident.
The old saying, you catch more bees than with honey than vinegar may be applicable in this scenario as well. If Lenny had gathered all the contractors to pool their knowledge, and explain what he could about the incident, he could have killed two birds with one stone.
As the contractors sat in the conference room they were away from their systems, which Lenny wanted, he would also be able to tactfully have a group of the smartest folks with an intimate knowledge of his network work as a team to solve a common problem.
Corporate HQ was a larger organization and had owned and operated several small startup organizations; they most likely were unaware of their firewall misconfigurations. Usually where you find one you find others, so if Corporate HQ had been made aware of the incident from the beginning, rooting out the worm would have been team effort opposed to the every man for himself approach.
Although the worm was successfully removed from SUN_TEST1, the proverbial bullet was dodged. What would the outcome have been if the worm would have been more advanced and would have propagated through Windows 2000 clients, of which we had 50 systems all key to satisfying customer service and administrative requirements. Days of cleanup would be required; customer service and billing departments would have been completely incapacitated.
Many Anti-virus solutions are made available for Solaris environments, the most valuable assets when considering data in this example would have been the customer data, however they were left completely unprotected to viruses and

Trojans. The cost of an anti-virus application is usually far less than the cost of downtime of a mission critical database and the cost of labor hours to rebuild it. Additionally, there was no positive control regarding configuration management of the individual systems. Several administrators onsite and at corporate had access to mission critical workstations and database servers. This was a simple issue of the left hand not knowing what the right was doing.

Once the compromised system was identified reviewing the /var/adm/log/syslog or /var/adm/messages files would have indicated the source and day of compromise. In any event the compromise had been ongoing for several days under our noses; it wouldn't have hurt to remove the system from the network to copy the system to disk or tape.

When first discovered we knew very little about the worm, or how wide spread it was. Backing up the system to cover your tail is never a bad idea; this step at least would have taken about an hour to conduct. The organizations success did not hinge on SUN_TEST1 and its operational status or uptime.

Collecting evidence would have helped if the Army or any of the compromised sites had taken legal action against our organization, and disks or tapes could have been made readily available should the data ever be subpoenaed.

If better lines of communication and policy had been implemented between corporate headquarters and our remote office this attack would have most likely failed.

By discussing required services and implementing proper firewall ACL's then finally checking our work by scanning or conducting permitted self audits we would have most likely prevented unwanted malicious traffic on our network.

References

Internet Security Systems –

http://www.iss.net/security_center/advice/Services/SunRPC/default.htm

Network Associates Technology Inc. –

http://vil.nai.com/vil/content/v_99085.htm

University of Nebraska – Bill Hayes

http://www.unl.edu/security/virus_alerts/sadmind.htm

http://www.avp.ch/ - KasperskyLab

http://www.avp.ch/avpve/worms/net/sadmind.stm

SANS – David P. Reece “RPCbind and Portmapper”

http://www.sans.org/resources/idfaq/blocking.php

Incidents.org –

http://www.incidents.org/archives/intrusions/msg03805.html

Securityfocus.com –

http://www.securityfocus.com/bid/1806/info/

http://www.securityfocus.com/bid/1806/solution/

Securiteam.com – Microsoft Product Security

http://www.securiteam.com/windowsntfocus/6U00B2000A.html

Beale, J. Foster, J.C. Posluns, J. Caswell, B ed. (2003) *Snort 2.0* Rockland, MA: Syngress