1. In 2006 [CVE-2006-0745] a security flaw was discovered in the then current version of the X.org server (X11R6.9.0 & X11R7.0 RC). The relevant portion of the source code is included. What is the issue?
2. Program 2 is being run in a debugger, and execution has been paused at line 12. Answer the following questions:

(a) What memory location is the base for the current stack frame?
(b) Where on the stack frame is the return address located?
(c) What is the function’s return address?
(d) Where on the stack frame is the function’s saved EBP located?
(e) What is the value of the function’s saved EBP?
(f) What address(es) contains the argument(s) of the function?
(g) What are the value(s) of the argument(s) of the function?
3. Analyse program 3 for security flaws. If it has any security flaws, explain how they might be exploited.
4. For each of the following functions, explain what security problems they pose, and describe a more secure alternative.

(a) char *gets(char *s)
(b) char *strcpy(char *dest, const char *src)
(c) char *strcat(char *dest, const char *src)
5. We discussed a number of mitigation strategies to prevent stack-based buffer overflows. Choose any one, and give a brief technical description, including a summary of its advantages and disadvantages.
6. Analyse program 6 for security flaws. If it has any security flaws, explain the simplest way to correct them.
7. The X Window System Lock Screen program is typically suid root, and was shown to have a serious software vulnerability [versions < 4.16.1] [CVE-2000-0763]. Examine the relevant portion of the source code, locate the problem, and propose a fix.
8. What is a Time of Check, Time of Use (TOCTOU) race condition? Write a short snippet of code with this vulnerability. Explain the relevant security issue.
9. A co-worker proposes using the code snippet in program 9 as a way to securely create and open temporary files. What is your analysis of the method?
10. Analyze the code in program 10 for security vulnerabilities.
11. What is a cross-site scripting vulnerability? Sketch a small page with such a vulnerability, and explain how it might be exploited.
12. Is the code from script 12 vulnerable to cross-site scripting? Explain.
13. Analyse the code from script 13 for security flaws.
14. What is a session id? What are its security implications?
15. Suppose that an application is using a linear congruential random number generator with published modulus, multiplier, and increment. How many pseudo-random numbers can be shown to a user before the user is able to determine the internal state of the random number generator? What are the security implications?
1 Question 1

1.1 Program Source

```c
int ddxProcessArgument(int argc, char **argv, int i)
{
    /*
     * Note: can’t use xalloc/xfree here because OsInit() hasn’t been called
     * yet. Use malloc/free instead.
     */

    #define CHECK_FOR_REQUIRED_ARGUMENT()
    
    if (((i + 1) >= argc) || (!argv[i + 1])) {
        ErrorF("Required argument to %s not specified\n", argv[i]);
        UseMsg();
        FatalError("Required argument to %s not specified\n", argv[i]);
    }

    /* First the options that are only allowed for root */
    if (getuid() == 0 || geteuid != 0)
    {
        if (!strcmp(argv[i], "-modulepath"))
        {
            char *mp;
            CHECK_FOR_REQUIRED_ARGUMENT();
            mp = malloc(strlen(argv[i + 1]) + 1);
            if (!mp)
                FatalError("Can’t allocate memory for ModulePath\n");
            strcpy(mp, argv[i + 1]);
            xf86ModulePath = mp;
            xf86ModPathFrom = X_CMDLINE;
            return 2;
        }
        else if (!strcmp(argv[i], "-logfile"))
        {
            char *lf;
            CHECK_FOR_REQUIRED_ARGUMENT();
            lf = malloc(strlen(argv[i + 1]) + 1);
            if (!lf)
                FatalError("Can’t allocate memory for LogFile\n");
            strcpy(lf, argv[i + 1]);
            xf86LogFile = lf;
            xf86LogFileFrom = X_CMDLINE;
            return 2;
        }
        else if (!strcmp(argv[i], "-modulepath") || !strcmp(argv[i], "-logfile") || !strcmp(argv[i], "-config") || !strcmp(argv[i], "-xf86config"))
        {
            FatalError("The ‘%s’ option can only be used by root.\n", argv[i]);
        }
    }

    if (!strcmp(argv[i], "-config") || !strcmp(argv[i], "-xf86config"))
    {
```
CHECK_FOR_REQUIRED_ARGUMENT();
if (getuid() != 0 && !xf86PathIsSafe(argv[i + 1])) {
    FatalError("\nInvalid argument for %s\n"
    "For non-root users, the file specified with %s must be\n    a relative path and must not contain any ".." elements.\n    \tUsing default "__XCONFIGFILE__" search path.\n    
    argv[i], argv[i]);
    xf86ConfigFile = argv[i + 1];
    return 2;
}
if (!strcmp(argv[i], "-showunresolved"))
    { xf86ShowUnresolved = TRUE;
       return 1;
    }
if (!strcmp(argv[i], "-probeonly"))
    { xf86ProbeOnly = TRUE;
       return 1;
    }
if (!strcmp(argv[i], "-flipPixels"))
    { xf86FlipPixels = TRUE;
       return 1;
    }

/* Omitted material for simplicity */

/* OS-specific processing */
return xf86ProcessArgument(argc, argv, i);
2 Question 2

2.1 Program Source

/ *
  Program to determine the number of capital letters in a string
  */

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

unsigned int get_capitals(char *str) {

  unsigned int i,n;
  unsigned int n_caps = 0;

  n = strlen(str); /* Execution Stopped Here */
  for(i=0; i<n; i++)
    if( str[i] >= 65 && str[i] <=90)
      n_caps++;

  return n_caps;
}

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

  unsigned int n;

  if(argc != 2) {
    printf("Usage: %s <string>.
The code will return the \
number of capital letters in <string>",argv[0]);
    exit(0);
  }

  n = get_capitals(argv[1]);
  printf("The input contained %u capital letters\n",n);

  exit(0);
}

2.2 Registers

  eax  0xbfc3e0a4 -1077682012
  ecx  0xbfc3d210 -1077685744
  edx  0xbfc3d210 -1077685744
  ebx  0xb7f02ff4 -1208995852
  esp  0xbfc3d1b0 0xbfc3d1b0
  ebp  0xbfc3d1c8 0xbfc3d1c8
  esi  0xb7f3ecc0 -1208750912
  edi  0x0 0
2.3 Stack Memory

2.4 Dissassembly

.file "Q2.c"
.text
.globl get_capitals
.type get_capitals, @function
get_capitals:
pushl %ebp
movl %esp, %ebp
pushl %edi
subl $20, %esp
movl $0, -8(%ebp)
movl 8(%ebp), %eax
movl $-1, %ecx
movl %eax, -24(%ebp)
movl $0, %eax
cld
movl -24(%ebp), %edi
repnz
scasb
movl %ecx, %eax
notl %eax
subl $1, %eax
movl %eax, -12(%ebp)
movl $0, -16(%ebp)
jmp .L2
.L3:
movl -16(%ebp), %eax
addl 8(%ebp), %eax
movzbl (%eax), %eax
cmpb $64, %al
jle .L4
movl -16(%ebp), %eax
addl 8(%ebp), %eax
movzbl (%eax), %eax
cmpb $90, %al
jg .L4
addl $1, -8(%ebp)
.L4:
addl $1, -16(%ebp)
.L2:
movl -16(%ebp), %eax
cmpl -12(%ebp), %eax
jb .L3
movl -8(%ebp), %eax
addl $20, %esp
popl %edi
popl %ebp
ret
.size get_capitals, .-get_capitals
.align 4
.LC0:
.string "Usage: %s <string>.\nThe code will return the\nnumber of capital letters in <string>\n"
.align 4
.LC1:
.string "The input contained %u capital letters\n"
.text
.globl main
.type main, @function
main:
leal 4(%esp), %ecx
addl $-16, %esp
pushl -4(%ecx)
pushl %ebp
movl %esp, %ebp
pushl %ecx
subl $36, %esp
movl %ecx, -24(%ebp)
movl -24(%ebp), %eax
cmpl $2, (%eax)
je .L10
movl -24(%ebp), %edx
movl 4(%edx), %eax
movl (%eax), %eax
movl %eax, 4(%esp)
call printf
movl $0, (%esp)
call exit
.L10:
movl -24(%ebp), %edx
movl 4(%edx), %eax
addl $4, %eax
movl (%eax), %eax
movl %eax, (%esp)
call get_capitals
movl %eax, -8(%ebp)
movl -8(%ebp), %eax
movl %eax, 4(%esp)
movl $.LC1, (%esp)
call printf
movl $0, (%esp)
call exit
.size main, -.main
.ident "GCC: (GNU) 4.1.0 (SUSE Linux)"
.section .note.GNU-stack,"",@progbits


3 Question 3

3.1 Source Code

/*
  Program to print the different directories in an environment variable
*/

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

#define BUF_SIZE 100

void print_dirs(char* name, char *env) {
    unsigned int i;
    unsigned int num_dirs = 0;
    unsigned int start, stop, dir;
    char directory[BUF_SIZE];

    for(i=0; i<strlen(env); i++)
        if(env[i] == ':')
            num_dirs++;

    printf("There are %u directories in the environment variable %s\n",num_dirs,name);

    dir=0;
    start=0;
    while(dir < num_dirs) {
        for(i=start; i<strlen(env); i++)
            if(env[i] == ':') {
                stop = i;
                break;
            }
        strncpy(directory,env+start*sizeof(char),stop-start);
        directory[stop-start]= 0;
        printf("%s\n",directory);
        dir++;
        start=stop+1;
    }
}

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

    if(argc != 2) {
        printf("Usage: %s <Environment Variable>.\n",argv[0]);
        printf("Prints out the directories contained in the given <Environment Variable>, if any.\n");
    }
exit(0);
}

env = getenv(argv[1]);
if(env == NULL) {
    printf("No environment variable named %s\n", argv[1]);
    exit(0);
}

print_dirs(argv[1], env);
exit(0);

3.2 Sample program output

Coventry:˜/Desktop/Cosc 647 Fall 2007/Final> ./Q3 PATH
There are 11 directories in the environment variable PATH
/home/mike/bin
/usr/local/bin
/usr/bin
/usr/X11R6/bin
/bin
/usr/games
/opt/gnome/bin
/opt/kde3/bin
/usr/lib/jvm/jre/bin
/usr/lib/mit/bin
/usr/lib/mit/sbin

Coventry:˜/Desktop/Cosc 647 Fall 2007/Final> ./Q3 MANPATH
There are 3 directories in the environment variable MANPATH
/usr/local/man
/usr/share/man
/usr/X11R6/man
6 Question 6

6.1 Source Code

/*
   Program to generate password hashes
*/

#define _XOPEN_SOURCE

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

void get_hash(const char *s, const char *pw, char *hash){
    char salt[12];
    strcpy(salt,"$1$"); /* For MD5, the salt starts $1$ and has */
    strcat(salt,s); /* at most eight additional characters */
    strcpy(hash,crypt(pw,salt));
    return;
}

int main(int argc, char *argv[]){
    unsigned short int password_size, salt_size;
    char *password = NULL;
    char hash[35];
    if(argc != 3) {
        printf("Usage: %s <salt> <password>
",argv[0]);
        printf("The code will return the MD5 hash of the password using the salt
");
        exit(0);
    }
    salt_size = strlen(argv[1]);
    password_size = strlen(argv[2]);
    password = malloc((1+password_size)*sizeof(char));
    strcpy(password,argv[2]);
    if(salt_size>=9) {
        printf("Salts must be at most eight characters
");
        exit(0);
    }
    get_hash(argv[1],password,hash);
printf("The password hash is %s\n", hash);

exit(0);
7 Question 7

7.1 Source Code

extern char *text_dpass;
extern char *text_fpass;
extern char *text_chall;
static int challx, chally;

extern pbmain();

struct pblk pblock; /* our instance of the pblk */
struct pblk *pb; /* global pointer to the pblk */

extern int checkDynamic();

#endif
#ifndef SYSLOG_FACILITY
#define SYSLOG_FACILITY LOG_AUTH
#endif

static void
syslogStart(void)
{
    struct passwd *pw;
    struct group *gr;

    pw = getpwuid(getuid());
    gr = getgrgid(getgid());

    (void) openlog(ProgramName, LOG_PID, SYSLOG_FACILITY);
    syslog(SYSLOG_INFO, "Start: %s, %s, %s",
           pw->pw_name, gr->gr_name, XDisplayString(dsp));
}

void
syslogStop(char *displayName)
{
    struct passwd *pw;
    struct group *gr;
    int secs, mins;

    secs = (int) (seconds() - start_time);
    mins = secs / 60;
    secs %= 60;

    pw = getpwuid(getuid());
    gr = getgrgid(getgid());
}
syslog(SYSLOG_INFO, "Stop: %s, %s, %s, %dm %ds",
    pw->pw_name, gr->gr_name, displayName, mins, secs);
}
#endif

void
error(char *buf)
{
#if defined( HAVE_SYSLOG_H ) && defined( USE_SYSLOG )
    extern Display *dsp;
    syslog(SYSLOG_WARNING, buf);
    if (!nolock) {
        if (strstr(buf, "unable to open display") == NULL)
            syslogStop(XDisplayString(dsp));
        else
            syslogStop("unknown display");
        closelog();
    }
#else
    (void) fprintf(stderr, buf);
#endif
    exit(1);
}
9 Question 9

9.1 Source Code Fragment

```c
char *filename;
int fd;

filename = tempnam (NULL, "foo");
fd = open (filename, O_CREAT | O_EXCL | O_TRUNC | O_RDWR, 0600);
free (filename);
```
10  Question 10

10.1  Source Code Fragment

```php
<?
$id=$_GET["id"];
if(!empty($id))
{
    $f=fopen("./weather/$id.txt", "r");
    while($r=fread($f, 1024))
    {
        echo $r;
    }
}else
    echo "Choose a day
<br>
<a href=weather.php?id=001>Today</a><br>
<a href=weather.php?id=002>Tonight</a><br>
<a href=weather.php?id=003>Tomorrow</a><br>
<a href=weather.php?id=004>Tomorrow Night</a><br>
";
?>
```
12 Question 12

12.1 Source Code

```html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" 
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <title>Order Form</title>
    <meta name="GENERATOR" content="Quanta Plus">
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<h3>Order Form</h3>
<p>Thank you for placing an order with us today.</p>

<?php
$sid = $_POST['sid'];
echo "<p>Your confirmation number for this order is $sid.</p>"
echo "<hr><p>To pay, we need to ....</p>";
?
</body>
</html>
```
13  Question 13

13.1  q13.php

```php
<?php
include_once("../database/database.inc");

$account = $_POST['Account'];
$password = $_POST['Password'];
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>View Single Account</title>
<meta name="GENERATOR" content="Quanta Plus">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<h3>Account Information</h3>

<?php
if(isset($account)) {
    $safe_account = htmlentities($account); /* Sanitize data before query */
    $hash = md5("salt".$password);
    $sq = "SELECT name,address,cc,id
        FROM Account
        WHERE pw='$hash'
        AND id=$safe_account";

    $result = mysql_query($sq);
    if($result !== false)
        echo "The request failed. Please retry";
    else {
        $row = mysql_fetch_array($result);
        print "<p>Account Data:</p>
        <table>
        <tr><td>Name: </td><td> {$row['name']} </td></tr>
        <tr><td>Address: </td><td> {$row['address']} </td></tr>
        <tr><td>Credit Card: </td><td> {$row['cc']} </td></tr>
        <tr><td>Account ID: </td><td> {$row['id']} </td></tr>
        </table>";
    }
}
<p> Enter the account you would like to return:</p>
<form action="<?=$_SERVER['PHP_SELF']??>" method="POST">
```
13.2 database.inc

```php
<?php

$mysqlhost = 'localhost';
$mysqluser = 'ShopUser';
$mysqlpass = 'shoppass';
$mysqldb = 'Shop';

$link = @mysql_connect($mysqlhost,$mysqluser,$mysqlpass);
if($link == false)
    err();
$result = mysql_select_db($mysqldb);
if($result == false)
    err();

function err()
{
    echo <<<html
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <title>View Single Account</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    </head>
    <body>
    <p>There has been a problem establishing a connection to the database. Please try again later.</p>
    </body>
    </html>
    html;
    exit();
}
?>