Format String exploits

•••

192.092 SE Capture the Flag / Winter 2019

Hannes Hauer

Overview

Binary exploitation technique

Examples:

- lazy (SECCON 2019 Online)
- rot26 (RedpwnCTF 19)

Overview: lazy

```
┌─[user@parrot]─[/media/s
└──╼ $./lazy
1: Public contents
2: Login
3: Exit
```

```
┌─[user@parrot]─[/media/sf_ctf/
└──╼ $./lazy
1: Public contents
2: Login
3: Exit
2
username : Admin
Welcome, Admin

password : Password
Invalid username
Exit!
┌─[✗]─[user@parrot]─[/media/sf_
└──╼ $
```

3. Default (ssh)

```
┌─[user@parrot]─[/media/sf_ctf/presentation]
└──╼ $./lazy
1: Public contents
2: Login
3: Exit
1
Welcome to public directory
You can download contents in this directory
login_source.c
login_source.c
./login_source.c
Sending 1201 bytes#define BUFFER_LENGTH 32
#define PASSWORD "XXXXXXXXXX"
#define USERNAME "XXXXXXXX"

int login(void){
        char username[BUFFER_LENGTH];
        char password[BUFFER_LENGTH];
        char input_username[BUFFER_LENGTH];
        char input_password[BUFFER_LENGTH];

        memset(username,0x0,BUFFER_LENGTH);
        memset(password,0x0,BUFFER_LENGTH);
        memset(input_username,0x0,BUFFER_LENGTH);
        memset(input_password,0x0,BUFFER_LENGTH);
```

lazy

```c
#define BUFFER_LENGTH 32
#define PASSWORD "XXXXXXXXXX"
#define USERNAME "XXXXXXXX"

int login(void){
    char username[BUFFER_LENGTH];
    char password[BUFFER_LENGTH];
    char input_username[BUFFER_LENGTH];
    char input_password[BUFFER_LENGTH];

    memset(username,0x0,BUFFER_LENGTH);
    memset(password,0x0,BUFFER_LENGTH);
    memset(input_username,0x0,BUFFER_LENGTH);
    memset(input_password,0x0,BUFFER_LENGTH);

    strcpy(username,USERNAME);
    strcpy(password,PASSWORD);

    printf("username : ");
    input(input_username);
    printf("Welcome, %s\n",input_username);

    printf("password : ");
    input(input_password);


    if(strncmp(username,input_username,strlen(USERNAME)) != 0){
        puts("Invalid username");
        return 0;
    }

    if(strncmp(password,input_password,strlen(PASSWORD)) != 0){
        puts("Invalid password");
        return 0;
    }

    return 1;
}


void input(char *buf){
        int recv;
        int i = 0;
        while(1){
                recv = (int)read(STDIN_FILENO,&buf[i],1);
                if(recv == -1){
                        puts("ERROR!");
                        exit(-1);
                }
                if(buf[i] == '\n'){
                        return;
                }
                i++;
        }
}
```

```c
    char username[BUFFER_LENGTH];
    char password[BUFFER_LENGTH];
    char input_username[BUFFER_LENGTH];
    char input_password[BUFFER_LENGTH];

    memset(username,0x0,BUFFER_LENGTH);
    memset(password,0x0,BUFFER_LENGTH);
    memset(input_username,0x0,BUFFER_LENGTH);
    memset(input_password,0x0,BUFFER_LENGTH);

    strcpy(username,USERNAME);
    strcpy(password,PASSWORD);

    printf("username : ");
    input(input_username);
    printf("Welcome, %s\n",input_username);

    printf("password : ");
    input(input_password);
```

```c
        while(1){
                recv = (int)read(STDIN_FILENO,&buf[i],1);
                if(recv == -1){
                        puts("ERROR!");
                        exit(-1);
                }
                if(buf[i] == '\n'){
                        return;
                }
                i++;
        }
```

rtfm: printf

3. Default (ssh)

```
Format of the format string
    The format string is a character string, beg
    initial  shift state, if any.  The format st
    more directives: ordinary characters (not  %
    changed  to  the  output stream; and convers
    which results in fetching zero or more subse
    version specification is introduced by the c
    conversion specifier.  In between there may
    more  flags, an optional minimum field width
    an optional length modifier.

    The arguments must correspond properly (afte
    conversion  specifier.  By default, the argu
    given, where each '*' (see Field width and P
    conversion  specifier asks for the next argu
    insufficiently many arguments are given).  O
    itly  which  argument  is taken, at each pla
    quired, by writing "%m$" instead of '%' and
    the  decimal integer m denotes the position
    desired argument, indexed starting from 1.

        printf("%*d", width, num);
Manual page printf(3) line 66 (press h for help or
```

3. Default (ssh)

```
BUGS
       Because  sprintf()  and  vsprintf()  assume an arbitrarily long string,
       callers must be careful not to overflow the actual space; this is often
       impossible  to assure.  Note that the length of the strings produced is
       locale-dependent and difficult to  predict.  Use  snprintf()  and  vs-
       nprintf() instead (or asprintf(3) and vasprintf(3)).

       Code  such as printf(foo); often indicates a bug, since foo may contain
       a % character.


EXAMPLE
       To print Pi to five decimal places:

           #include <math.h>
           #include <stdio.h>
           fprintf(stdout, "pi = %.5f\n", 4 * atan(1.0));

       To print a date and time in the form "Sunday,  July  3,  10:02",  where
       weekday and month are pointers to strings:

           #include <stdio.h>
Manual page printf(3) line 431 (press h for help or q to quit)
```

lazy

```c
#define BUFFER_LENGTH 32
#define PASSWORD "XXXXXXXXXX"
#define USERNAME "XXXXXXXX"

int login(void){
    char username[BUFFER_LENGTH];
    char password[BUFFER_LENGTH];
    char input_username[BUFFER_LENGTH];
    char input_password[BUFFER_LENGTH];

    memset(username,0x0,BUFFER_LENGTH);
    memset(password,0x0,BUFFER_LENGTH);
    memset(input_username,0x0,BUFFER_LENGTH);
    memset(input_password,0x0,BUFFER_LENGTH);

    strcpy(username,USERNAME);
    strcpy(password,PASSWORD);

    printf("username : ");
    input(input_username);
    printf("Welcome, %s\n",input_username);

    printf("password : ");
    input(input_password);


    if(strncmp(username,input_username,strlen(USERNAME)) != 0){
        puts("Invalid username");
        return 0;
    }

    if(strncmp(password,input_password,strlen(PASSWORD)) != 0){
        puts("Invalid password");
        return 0;
    }

    return 1;
}


void input(char *buf){
        int recv;
        int i = 0;
        while(1){
                recv = (int)read(STDIN_FILENO,&buf[i],1);
                if(recv == -1){
                        puts("ERROR!");
                        exit(-1);
                }
                if(buf[i] == '\n'){
                        return;
                }
                i++;
        }
}
```

```c
    char username[BUFFER_LENGTH];
    char password[BUFFER_LENGTH];
    char input_username[BUFFER_LENGTH];
    char input_password[BUFFER_LENGTH];

    memset(username,0x0,BUFFER_LENGTH);
    memset(password,0x0,BUFFER_LENGTH);
    memset(input_username,0x0,BUFFER_LENGTH);
    memset(input_password,0x0,BUFFER_LENGTH);

    strcpy(username,USERNAME);
    strcpy(password,PASSWORD);

    printf("username : ");
    input(input_username);
    printf("Welcome, %s\n",input_username);

    printf("password : ");
    input(input_password);
```

```c
        while(1){
                recv = (int)read(STDIN_FILENO,&buf[i],1);
                if(recv == -1){
                        puts("ERROR!");
                        exit(-1);
                }
                if(buf[i] == '\n'){
                        return;
                }
                i++;
        }
```

lazy: Admin menu

3. ssh

```
┌─[user@parrot]─[/media/sf_ctf/presentation]
└──╼ $./lazy
1: Public contents
2: Login
3: Exit
2
username : _H4CK3R_
Welcome, _H4CK3R_

password : 3XPL01717
Logged in!
1: Public contents
2: Login
3: Exit
4: Manage
4
Welcome to private directory
You can download contents in this directory, but you c
h a dot in the name
lazy
core
Input file name
```

3. ssh

```
3: Exit
2
username : _H4CK3R_
Welcome, _H4CK3R_

password : 3XPL01717
Logged in!
1: Public contents
2: Login
3: Exit
4: Manage
4
Welcome to private directory
You can download contents in this directory, but you can't download contents wit
h a dot in the name
lazy
core
example
Input file name
example
Filename : example
OK! Downloading...
./example
Sending 3 bytesHi
┌─[user@parrot]─[/media/sf_ctf/presentation]
```

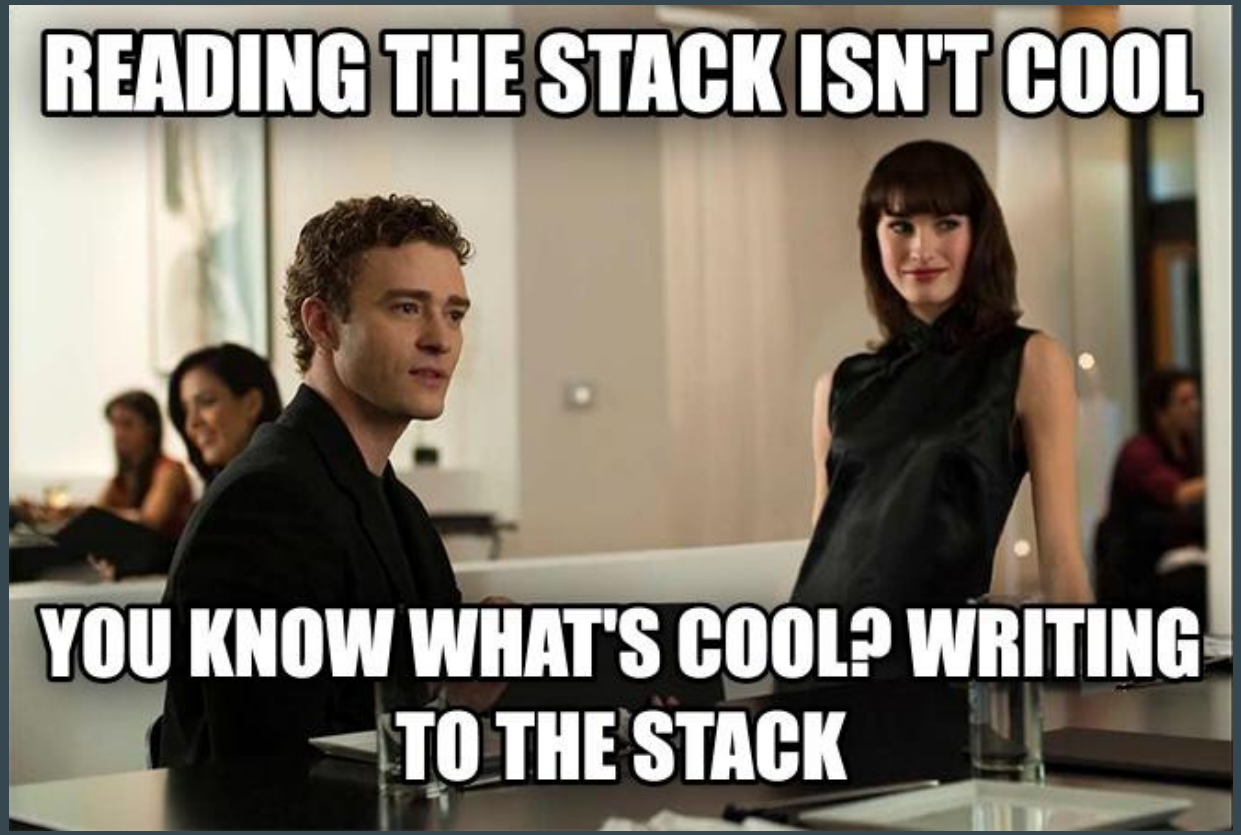
Demonstration lazy: leaking stack variables

rot26

```c
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char *ualphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
char *lalphabet = "abcdefghijklmnopqrstuvwxyz";

char *rot26(char *dst, char *src, size_t n)
{
    int i, x;

    for (i = 0; i < n; i++) {
        if (isupper(src[i])) {
            x = ualphabet[((src[i] - 'A') + 26) % strlen(ualphabet)];
        } else if (islower(src[i])) {
            x = lalphabet[((src[i] - 'a') + 26) % strlen(lalphabet)];
        } else {
            x = src[i];
        }
        dst[i] = x;
    }
}

void winners_room(void)
{
    puts("Please, take a shell!");
    system("/bin/sh");
    exit(EXIT_SUCCESS);
}

int main(void)
{
    char buf[4096];
    char sanitized[4096];

    setbuf(stdout, NULL);
    setbuf(stdin, NULL);
    setbuf(stderr, NULL);

    fgets(buf, sizeof(buf), stdin);
    rot26(sanitized, buf, sizeof(sanitized));
    printf(sanitized);
    exit(EXIT_FAILURE);
}

```


```c
void winners_room(void)
{
    puts("Please, take a shell!");
    system("/bin/sh");
    exit(EXIT_SUCCESS);
}
```

```c
    fgets(buf, sizeof(buf), stdin);
    rot26(sanitized, buf, sizeof(sanitized));
    printf(sanitized);
    exit(EXIT_FAILURE);
```



... or anywhere really.

Writing to memory

3. Default (ssh)

```
BUGS
       Because  sprintf()  and  vsprintf()  assume a
       callers must be careful not to overflow the a
       impossible  to assure.  Note that the length
       locale-dependent and difficult to  predict.
       nprintf() instead (or asprintf(3) and vaspri

       Code  such as printf(foo); often indicates a
       a % character.  If foo comes from untrusted u
       %n,  causing  the printf() call to write to m
       rity hole.

EXAMPLE
       To print Pi to five decimal places:

           #include <math.h>
           #include <stdio.h>
           fprintf(stdout, "pi = %.5f\n", 4 * atan(1

       To print a date and time in the form "Sunday,
       weekday and month are pointers to strings:

           #include <stdio.h>
 Manual page printf(3) line 431 (press h for help o
```

3. ssh

```
                for lc.  Don't use.

        S       (Not in C99 or C11, but in SUSv2, SUSv3,  and  SUSv4.)   Synonym
                for ls.  Don't use.

        p       The  void * pointer argument is printed in hexadecimal (as if by
                %#x or %#lx).

        n       The number of characters written so far is stored into the inte-
                ger  pointed  to  by  the corresponding argument.  That argument
                shall be an int *, or variant whose size  matches  the  (option-
                ally)  supplied  integer  length  modifier.  No argument is con-
                verted.  (This specifier is not supported by the  bionic  C  li-
                brary.)   The behavior is undefined if the conversion specifica-
                tion includes any flags, a field width, or a precision.

        m       (Glibc extension; supported by uClibc and musl.)   Print  output
                of strerror(errno).  No argument is required.

        %       A  '%' is written.  No argument is converted.  The complete con-
                version specification is '%%'.

RETURN VALUE
       Upon successful return, these functions return the number of characters
 Manual page printf(3) line 344 (press h for help or q to quit)
```

But where to?

Return address?

```
    fgets(buf, sizeof(buf), stdin);
    rot26(sanitized, buf, sizeof(sanitized));
    printf(sanitized);
    exit(EXIT_FAILURE);
}
```

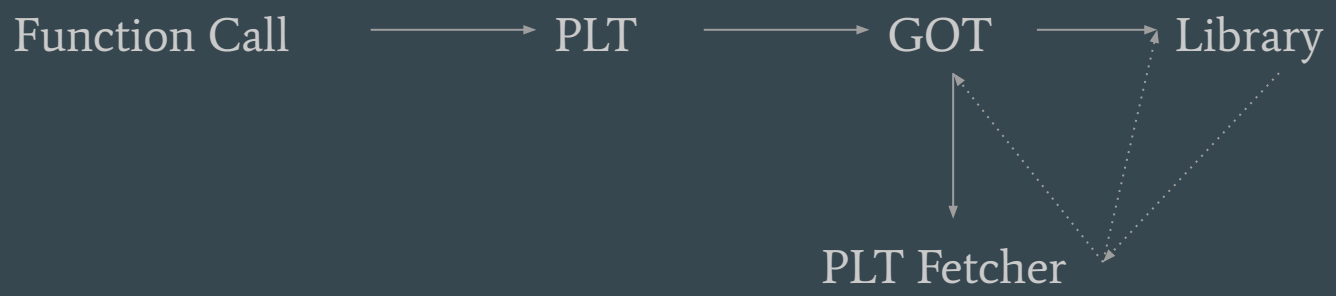
Maybe there’s something else…

```
   0x080487fc <+133>:   call   0x8048470 <fgets@plt>
   0x08048801 <+138>:   add    $0x10,%esp
   0x08048804 <+141>:   sub    $0x4,%esp
   0x08048807 <+144>:   push   $0x1000
   0x0804880c <+149>:   lea    -0x200c(%ebp),%eax
   0x08048812 <+155>:   push   %eax
--Type <RET> for more, q to quit, c to continue without paging--
   0x08048813 <+156>:   lea    -0x100c(%ebp),%eax
   0x08048819 <+162>:   push   %eax
   0x0804881a <+163>:   call   0x8048606 <rot26>
   0x0804881f <+168>:   add    $0x10,%esp
   0x08048822 <+171>:   sub    $0xc,%esp
   0x08048825 <+174>:   lea    -0x100c(%ebp),%eax
   0x0804882b <+180>:   push   %eax
   0x0804882c <+181>:   call   0x8048460 <printf@plt>
   0x08048831 <+186>:   add    $0x10,%esp
   0x08048834 <+189>:   sub    $0xc,%esp
   0x08048837 <+192>:   push   $0x1
   0x08048839 <+194>:   call   0x80484a0 <exit@plt>
End of assembler dump.
(gdb)
```

PLT? GOTcha!

PLT: Procedure Linking Table

GOT: Global Offset Table



Demo: rot26 Exploitation

Countermeasures

Listen to your compiler!

Full RELRO (& PIE)

3. ssh

```
┌─[user@parrot]─[/media/sf_ctf/presentation]
└──╼ $gcc -Wformat -Wformat-security -o my_rot26 rot26.c
rot26.c: In function 'main':
rot26.c:43:9: warning: format not a string literal and no format arguments [-Wfo
rmat-security]
   43 |  printf(sanitized);
      |         ^~~~~~~~~
```

3. ssh

```
┌─[user@parrot]─[/media/sf_ctf/presentation]
└──╼ $gcc -g -O0 -Wl,-z,relro,-z,now -o secure_rot26 rot26.c
┌─[user@parrot]─[/media/sf_ctf/presentation]
└──╼ $checksec secure_rot26
[*] '/media/sf_ctf/presentation/secure_rot26'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
┌─[user@parrot]─[/media/sf_ctf/presentation]
└──╼ $
```

How to use in challenges

How to spot:

- Source code? → Search for printf
- Binary? → Check file security. No RELRO/PIE? Hint at GOT-possibilities!
- Otherwise? → Pollute input strings with formatters

How to abuse:

- Leak data
- Arbitrary writes

Questions?

Sources & Further Information

- RedHat: Hardening ELF binaries using Relocation Read-Only (RELRO)
- LiveOverflow Pt. 1 + Pt. 2 + Pt. 3
- System Overlord: GOT and PLT for pwning.