Format String Dangers

Shachar Shemesh
Security Consultant
http://www.shemesh.biz

Schedule

- Reintroduction to printf (as if you don't already know…).
- Some reflection about common uses of printf.
- Dangers of letting attackers supply format strings.
- Step by step demonstration of exploit.

Printf manual

```
PRINTF(3)                Linux Programmer's Manual                PRINTF(3)

NAME
       printf,  fprintf,  sprintf,  snprintf,  vprintf,  vfprintf,  vsprintf,
       vsnprintf - formatted output conversion

SYNOPSIS
       #include <stdio.h>

       int printf(const char *format, ...);
       int fprintf(FILE *stream, const char *format, ...);
       int sprintf(char *str, const char *format, ...);
       int snprintf(char *str, size_t size, const char *format, ...);
```

The Format String

- Most characters are simply echoed.
- A “%” indicates a special field (unless followed by another “%”).
- It is up to the programmer to make sure the parameters number and type match the format string.
  - Some compilers will verify this for static format strings.

Format String Functions

- (v)printf
- (v)sprintf
- (v)snprintf
- (v)fprintf
- syslog
- (v)wprintf
- (v)fwprintf
- (v)swprintf
- (v)dprintf
- (v)asprintf

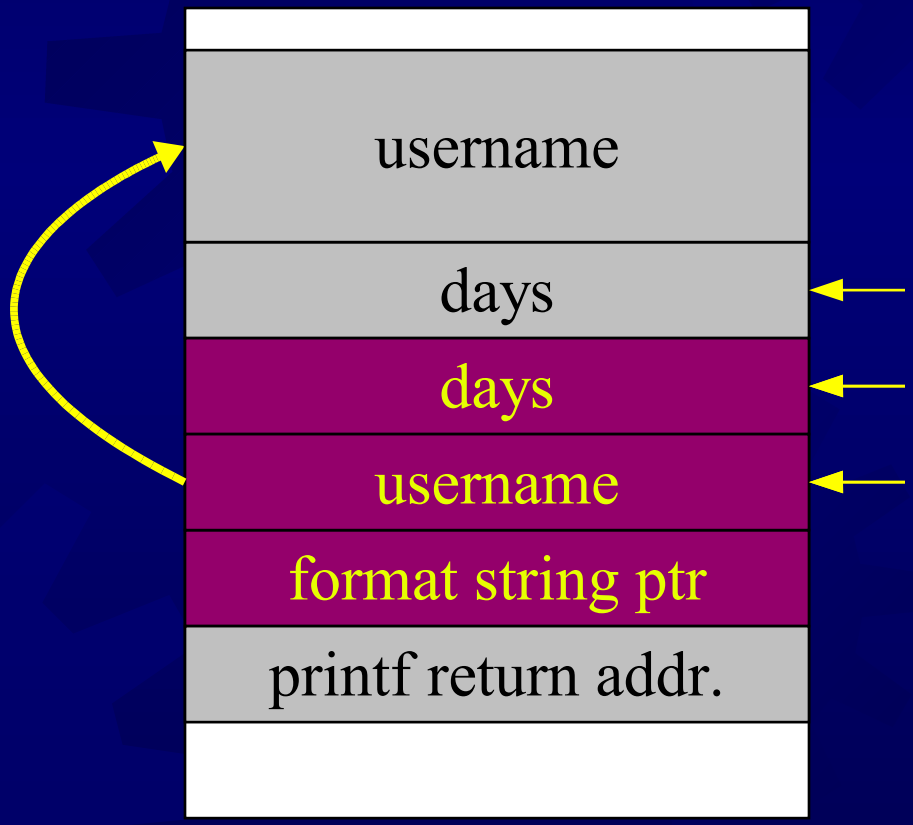
Caveat Emptor

- sysinfo hasn’t got the telltale “printf” word in it.
  - It is all too easy to mistake it for accepting a plain string.
- Accepting a format string is all too common with error functions.
  - An error is usually a situation triggerable by an attacker.
- Projects usually have similar, private, functions.

How Format Strings Work

```
printf("Hello %s, it's been %d days since your last login\n",
    username, days);
```

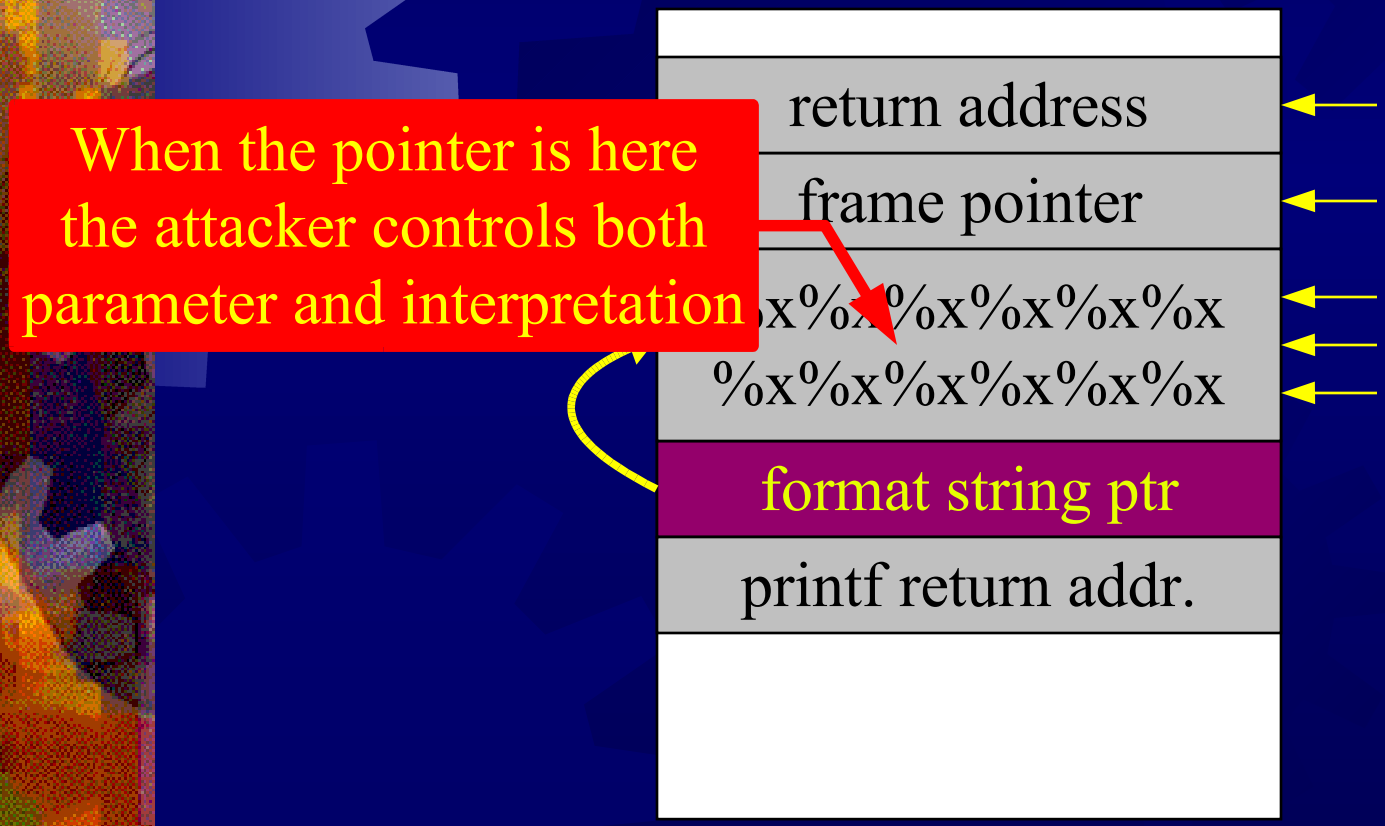
Hello sun, it's been 25 since your last login



When Format String and Parameters Mismatch

printf(buffer);

What if “buffer” is “%x%x%x%x%x%x%x%x”?

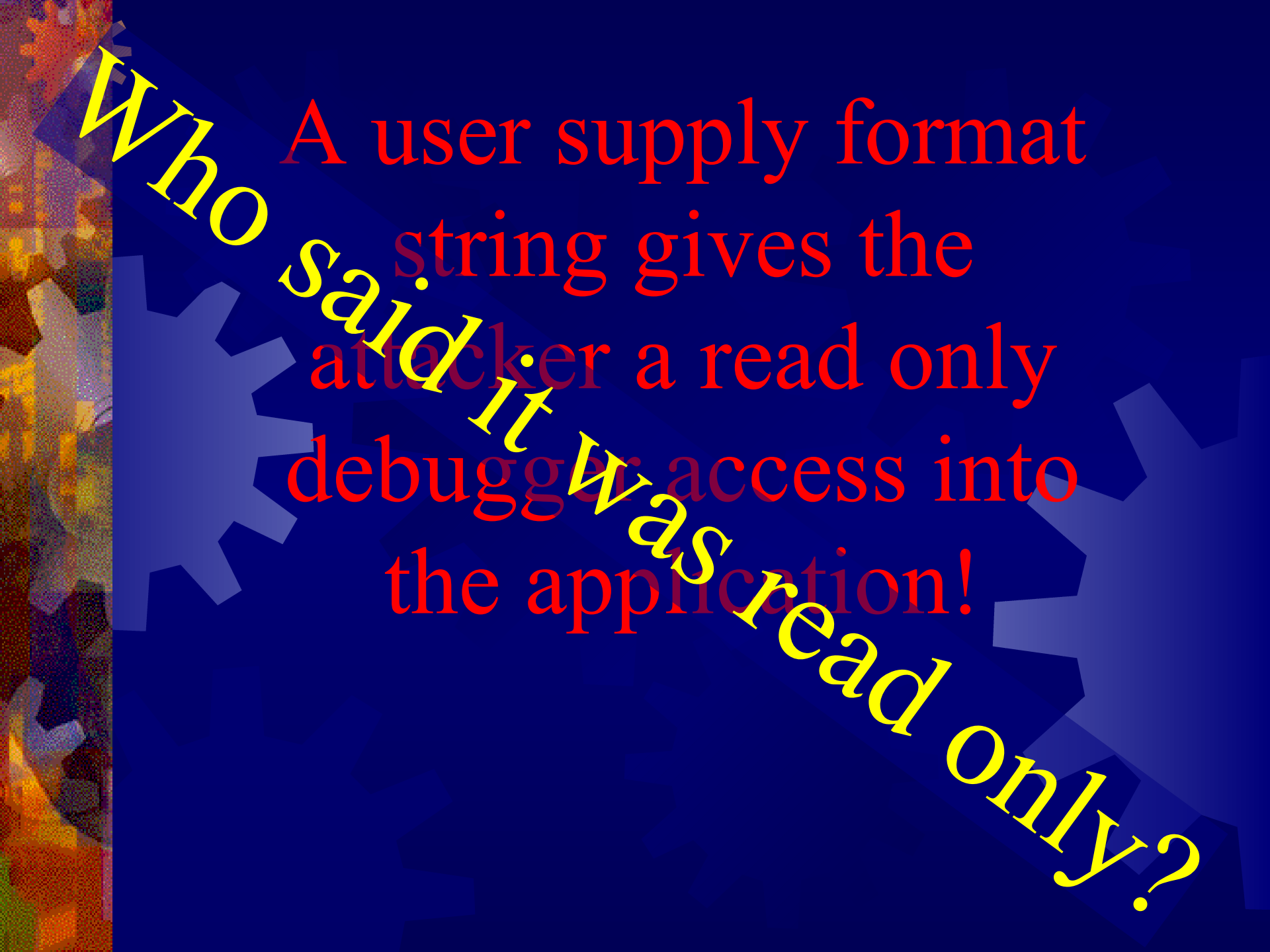


What Can be Done?

- Query parameters from the stack
  - %x, %d etc.
- Query data from anywhere in the program
  - %s when the pointer is inside the buffer to get info from anywhere.
  - Passwords, private keys.....
- Crash the program
  - %s from non-readable memory.
  - %f wich requires to devide by zero.

“Walk Ratio”

- The ratio between the pointer advance and the characters it take.
- Determines how far up from our buffer we can peek.
  - %x gives 1:2
  - %f gives 1:4, but risks devide by zero.
  - what does “`printf("%3$d %2$d %1$d",5,6,7);`” print?


Who said it was read only?
A user supply format string gives the attacker a read only debugger access into the application!

%n – The Little Option Noobody Knows

- %n writes into the int pointed to by the respective argument the number of characters printed so far.
- An attacker can choose to write (%n), where to write (supply a pointer).
- By playing with field length, can also control what to write.
- Bare shortcuts, that may require printing an average of 2GB of data.

Some of the Shortcuts

- Write four times to addresses increasing by 1 each time.
  - Will only work on platform that don’t enforce integral boundries (e.g. - Intel).
- Use %hn to write to short.
  - Now only requires printing 64K.
- Use %hhn to write to byte.
  - Only prints 256 bytes.

A Few Bad Habits

or – you won't believe what people do!

- The following code samples represent errors found (not necessarily by me) in shipping code (some commercial, some free).
- The exact code was modified to protect ~~paying customers~~ the innocent.

A Few Bad Habits

```
#define ASSERT(cond, err) \
   if(!(cond)) { \
   printf(err); exit(100); }
```

What will the following imagenary code do?

```
   ASSERT(progress>10,
      "Couldn't pass 10% mark" );
```

Pointless use

```
/* Initialize title */
sprintf( title, “About to copy files” );
```

- ‘*sprintf*’ scans the format string for fields.
  - Unofficial benchmark shows 50% performance of ‘*sprintf(buff, data)*’ over ‘*sprintf(buff, “%s”, data)*’.
- Using ‘*strcpy*’ or ‘*strncpy*’ would be much better in this case.

Dangerous Pitfall

```
int logerr( char *fmt, ... )
{
   va_list args;
   char buff[1024];

   va_start(args, fmt);
   vsnprintf(buff, sizeof(buff), fmt,
      args );
   va_end(args);

   return fprintf(errlog, buff);
}
```

Famous Example

```
#include <stdio.h>

main( )
{
    printf("%s",
        "Hello, world\n");
}
```

"The C Programming Language", Second Edition
Brian W. Kernighan
Dennis M. Ritchie
Prentice Hall Software Series, 1988