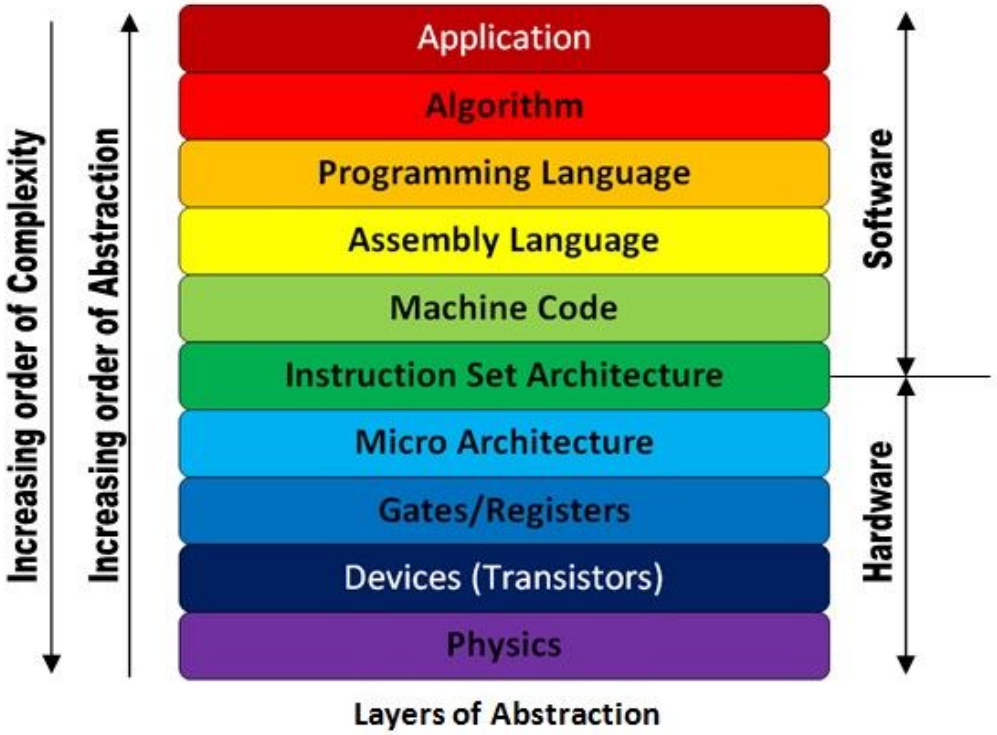
Low Level Security

Thushara Desapriya

Why Low Level



Web Application Issues

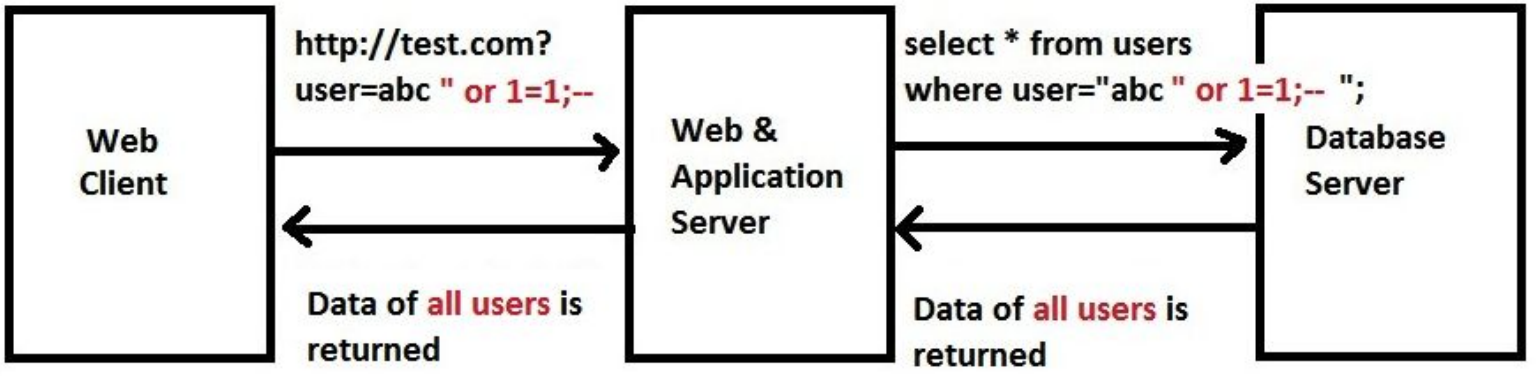
Insecure Deserialization

Treat it Right

Data should not become part of the Code

- Injections
- Cross Site Scripting

SQL Injection



SQL Injection Example

SQL Injection

```
String SQLQuery ="SELECT Username, Password
 FROM users WHERE Username='" + Username +
    "' AND Password='" + Password +"'";

Statement stmt = connection.oreateStaement();
ResultSet rs = stmt.executeQuery(SQLQuery);
while (rs.next()) { … }
```

SQL Injection

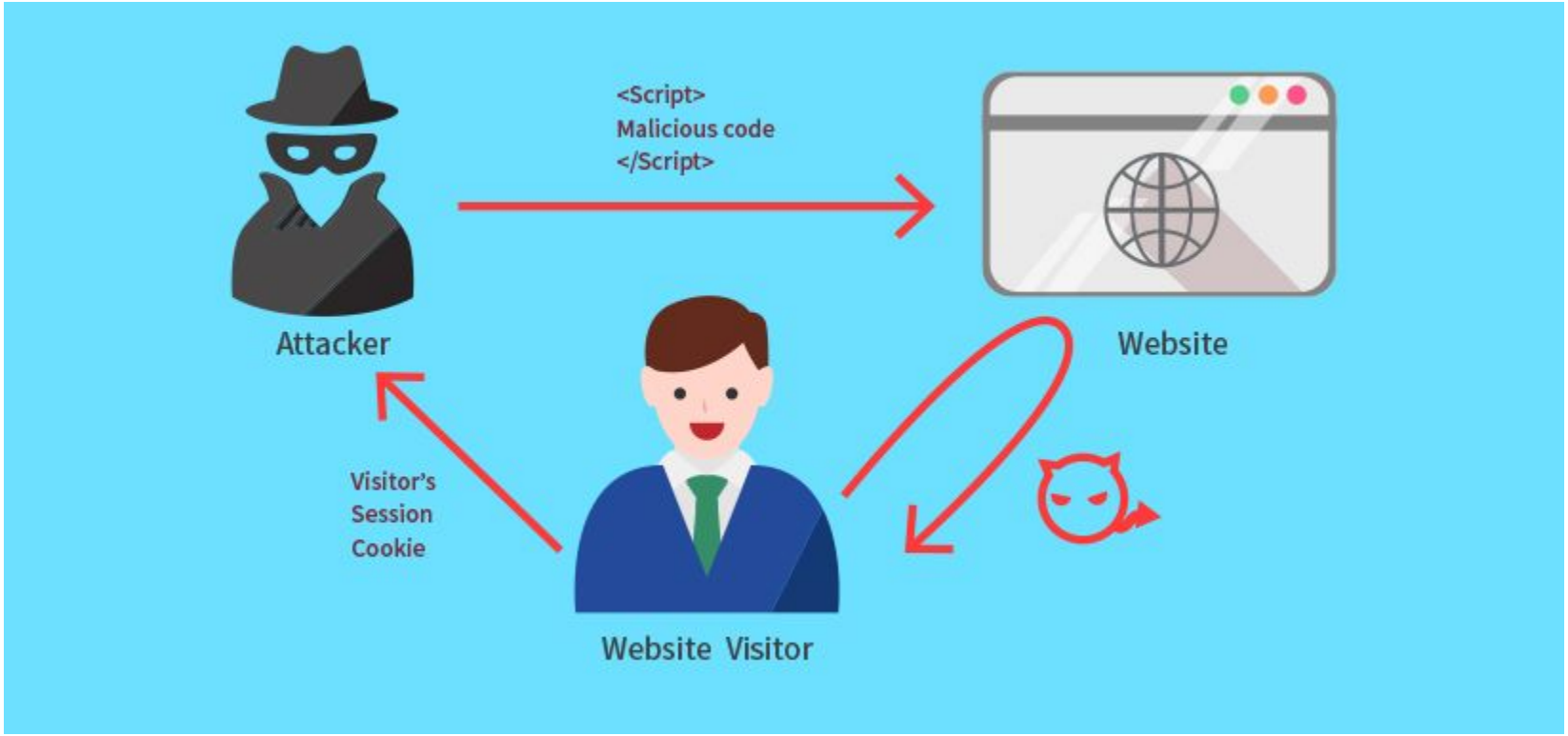
UserName = abc Password= xxx’ or ‘1’=’1

SELECT Username, Password FROM users WHERE

Username=’**abc**’ AND

Password=’**xxx’ or ‘1’=’**1’;

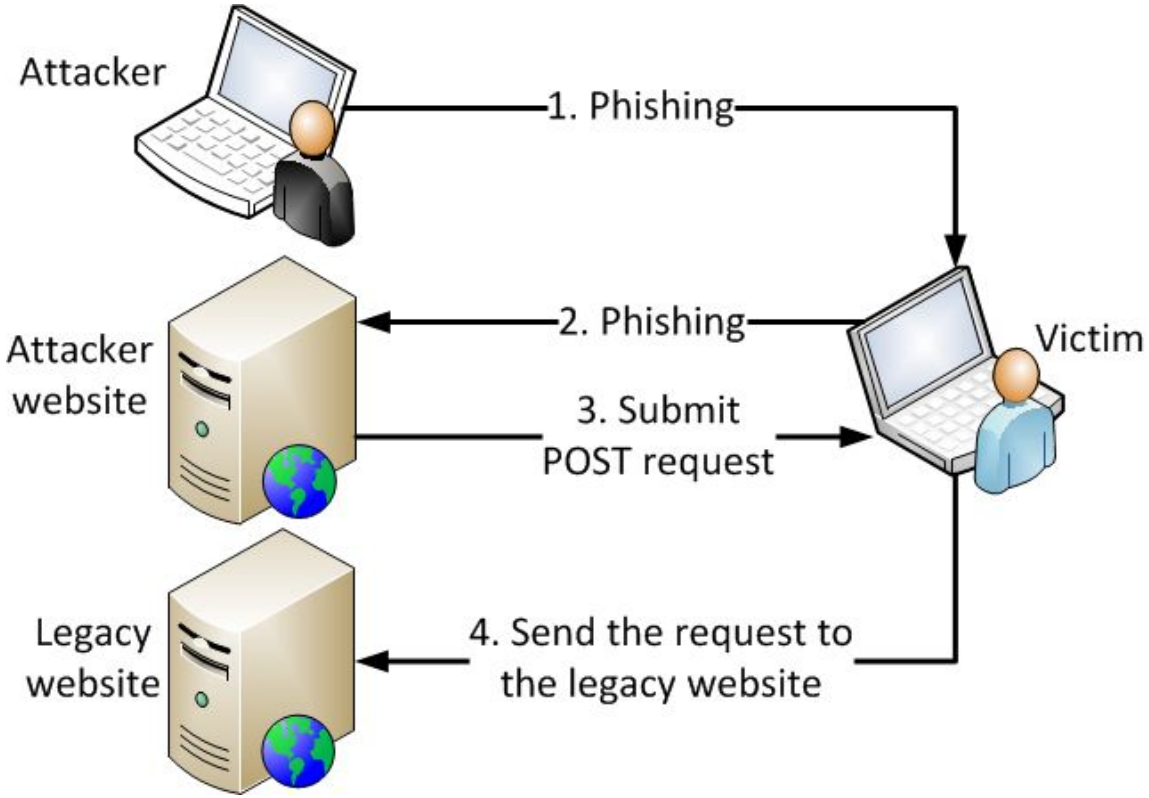
Cross Site Scripting - XSS



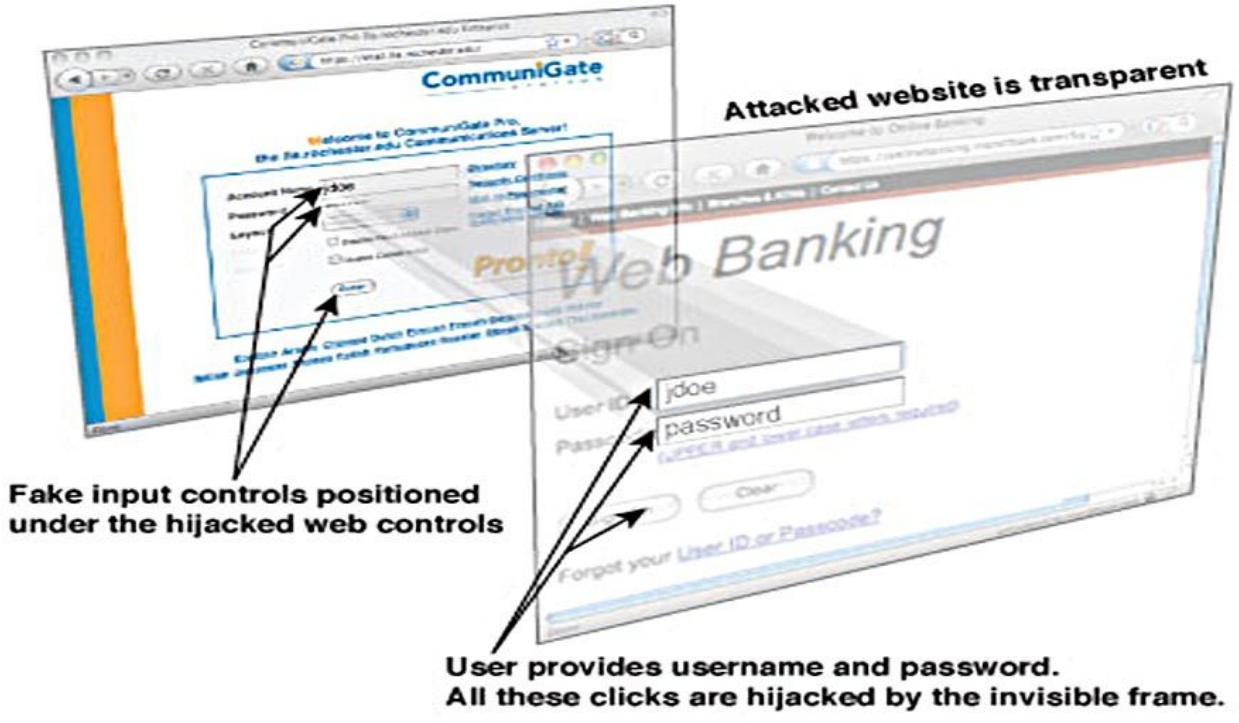
Trick the User

No Patch for Human Stupidity

CSRF - Cross-Site Request Forgery



Clickjacking



Unvalidated Redirects

Buffer Overflow

Buffer overflows were understood and partially publicly documented as early as 1972

1988 Morris Worm

- Shutdown over 6000 systems in just a few short hours
- Exploited fingered through exploiting a unchecked buffer initialized by gets()

Among SANS top 25 most dangerous software errors

- (Ref : http://www.sans.org/top25-software-errors/)

Buffer Overflow

2001 the Code Red worm exploited a buffer overflow in Microsoft's Internet Information Services (IIS) 5.0

2003 the SQL Slammer worm compromised machines running Microsoft SQL Server 2000.

2003, buffer overflows present in licensed Xbox games have been exploited to allow unlicensed software, including homebrew games, to run on the console without the need for hardware modifications, known as modchips.

The PS2 Independence Exploit also used a buffer overflow to achieve the same for the PlayStation 2.

The Twilight hack accomplished the same with the Wii, using a buffer overflow in The Legend of Zelda: Twilight Princess

What is Buffer Overflow

buffer overflow, or buffer overrun, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations

What is Buffer Overflow

A buffer overflow occurs when data written to a buffer also corrupts data values in memory addresses adjacent to the destination buffer due to insufficient bounds checking.

This can occur when copying data from one buffer to another without first checking that the data fits within the destination buffer.

What is Buffer Overflow

```
char           A[8] = "";
unsigned short B    = 1979;
```

Initially, A contains nothing but zero bytes, and B contains the number 1979.

variable name	A								B	
value	[null string]								1979	
hex value	00	00	00	00	00	00	00	00	07	BB

What is Buffer Overflow

```
strcpy(A, "excessive");
```

"excessive" is 9 characters long and encodes to 10 bytes including the null terminator, but A can take only 8 bytes. By failing to check the length of the string, it also overwrites the value of B:

variable name	A								B	
value	'e'	'x'	'c'	'e'	's'	's'	'i'	'v'	25856	
hex	65	78	63	65	73	73	69	76	65	00

```
strlcpy(A, "excessive", sizeof(A));
```

Buffer Overflow - Solution

```
strlcpy(A, "excessive", sizeof(A));
```

Can be prevented easily through using bounds checking

```
#include <stdio.h>
int main(int argc, char **argv)
{
        char buf[8];
        printf("Type your message : ");
        fgets(buf,8,stdin);
        printf("My Message : %s \n", buf);
        return 0;
}
```

What is Buffer Overflow

Writing data past the end of allocated memory can sometimes be detected by the operating system to generate a segmentation fault error that terminates the process.

Segmentation Fault : Fault raised by hardware with memory protection notifying an Operating System about a memory access violation

```
char password[16];

printf("Enter your Password :\n");
gets(password);
```

```
udesath@UDESATH-PC:~/c-classic/bypass$ ./bypass
Enter your Password :
ggggg
Invalid...!!!
udesath@UDESATH-PC:~/c-classic/bypass$ ./bypass
Enter your Password :
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Invalid...!!!
Segmentation fault (core dumped)
udesath@UDESATH-PC:~/c-classic/bypass$
```

Buffer Overflow - When

Data typed into text fields of a GUI

Data sent via a program

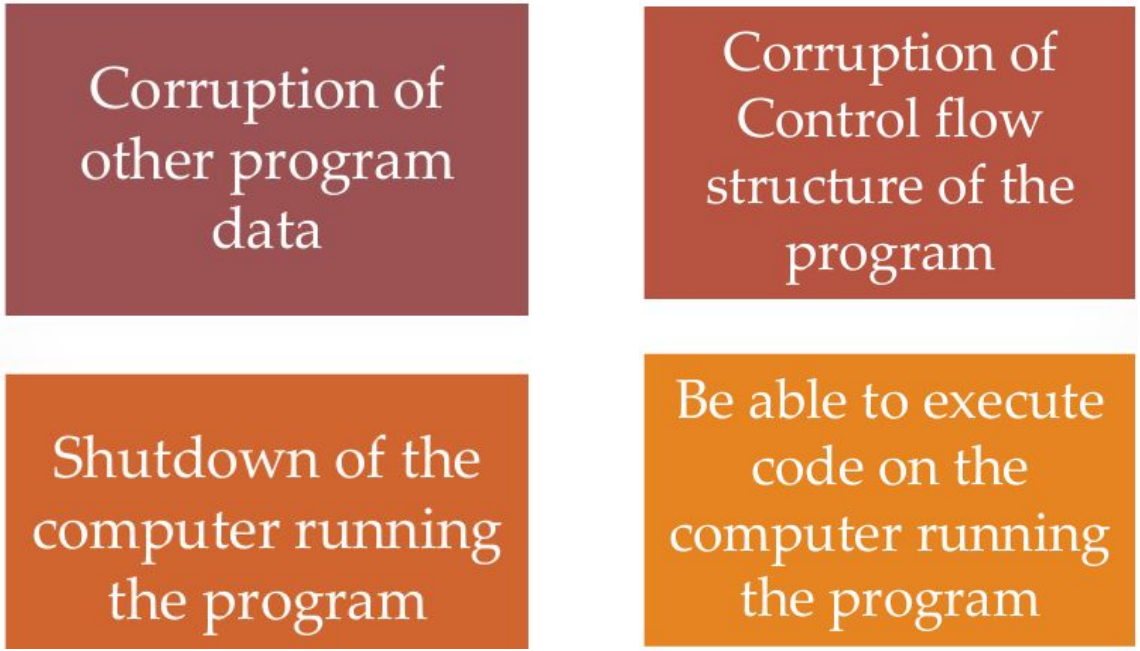
Data provided in a file

Data provided via command line

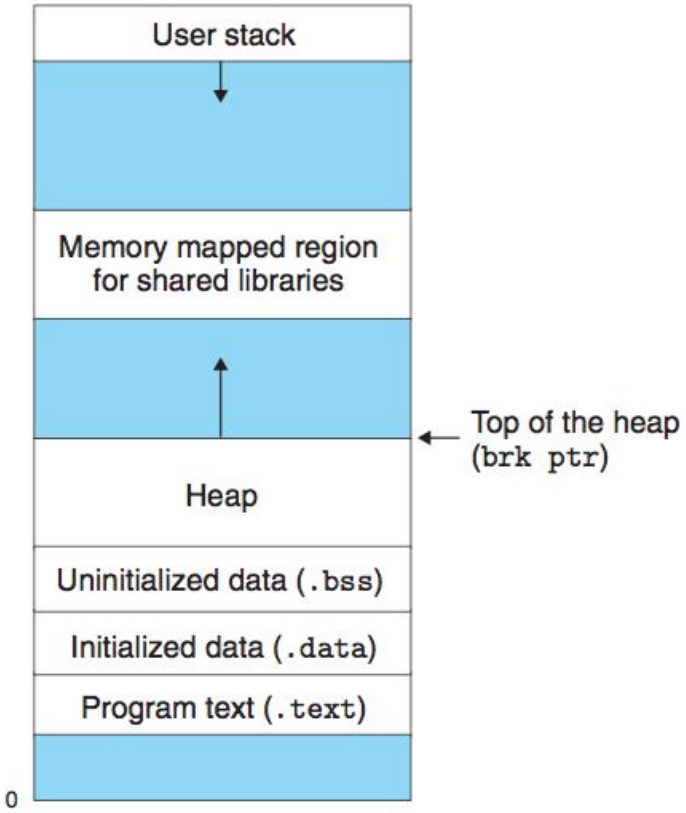
Data provided in Environment Variables

Buffer Overflow - Damage

So Buffer overflow can raise a memory access violation error and halt a program execution which may cause a denial of service.



Memory Structure



Heap

- Memory Allocated to variables during runtime using malloc()

Stack

- Used to store Local variables which are only used in one function. These variables are stored in a element called as frame

Data

- Readable and writable segments containing the static, global, initialized data segments and variables

BSS

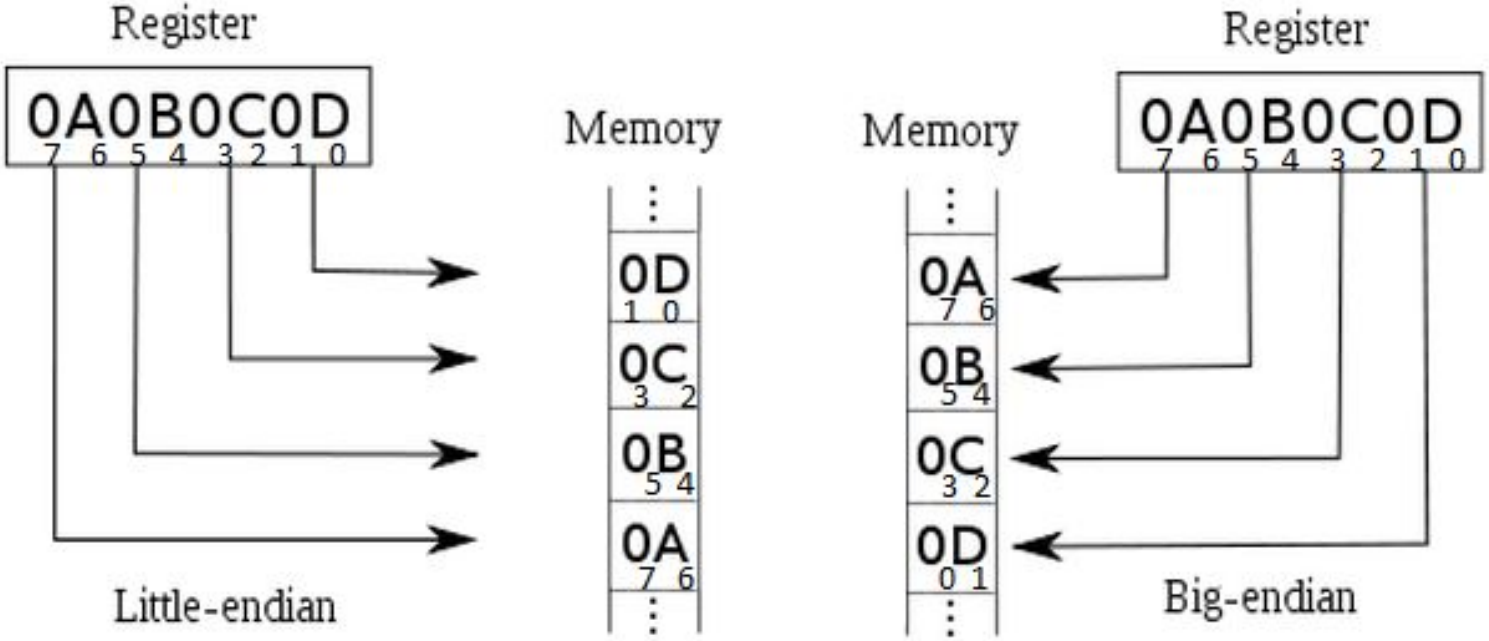
- Readable and writable segments containing the static, global, un-initialized data segments and variables

Text

- Read only segment that contains the compiled executable of the program

Big-endian and Little-endian

Big-endian and little-endian are terms that describe the order in which a sequence of bytes are stored in computer memory



Registers

Registers are either 32 or 64 bit high-speed storage locations directly inside the CPU, designed for high-speed access. For the purposes of discussion, registers can be grouped into the four categories of Data, Segment, Index, and Control

REGISTER SET INTEL 80386 ARCHITECTURE		
Category	**Register**	**Function**
Data	EAX (accumulator) EBX (base) ECX (counter) EDX (data)	Used for arithmetic and data movement. Each register can be addressed as either a 16 or 32 bit value. EBX can hold the address of a procedure or variable.
Segment	CS (code segment) DS (data segment) SS (stack segment) ES (extra segment) FS & GS	Used as base locations for program instructions, data, and the stack. All references to memory involve a segment register used as a base location.
Index	EBP (base pointer) ESP (stack pointer) ESI (source index) EDI (destination index)	Contain the offsets of data and instructions. The term offset refers to the distance of a variable or instruction from its base segment. The stack pointer contains the offset of the top of the stack
Control	EIP (instruction pointer) EFLAGS	The instruction pointer always contains the offset of the next instruction to be executed within the current code segment.

Stack Frame

Stack has frame which stores routines data

Return Address

- When a subroutine is called this stores the location of the instruction to be returned

Local Variable Storage Area

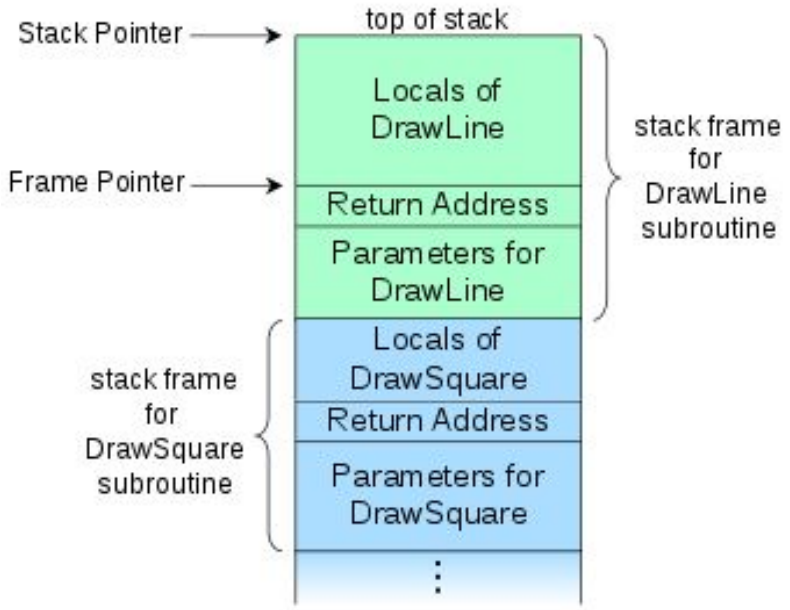
- Store values of local variables. Not retained after subroutine returns

Functions Parameter

- Store values which we pass to the subroutine

Frame Pointer / Base Pointer

- This stores previous value of the frame pointer register. The value the frame pointer register had before calling the subroutine.



Stack Buffer Overflow (Stack Smashing)

By overwriting a local variable that is located near the vulnerable buffer on the stack, in order to change the behavior of the program

By overwriting the return address in a stack frame. Once the function returns, execution will resume at the return address as specified by the attacker - usually a user-input filled buffer

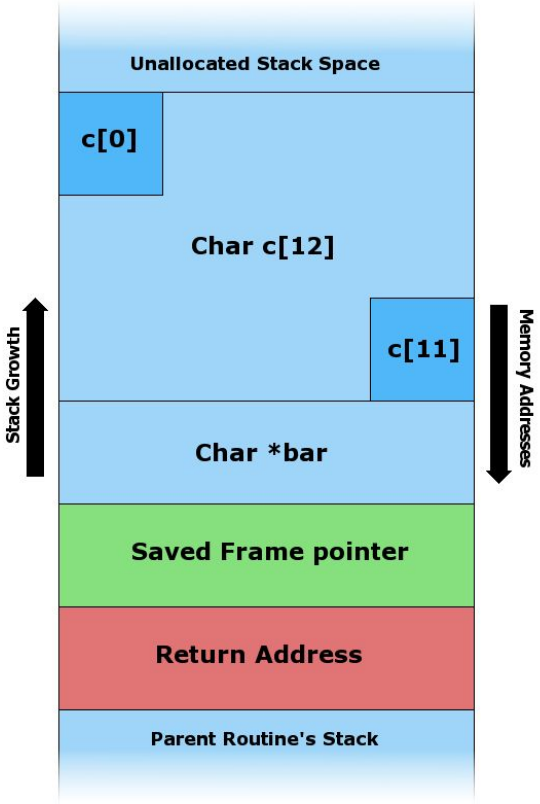
By overwriting a function pointer or exception handler, which is subsequently executed

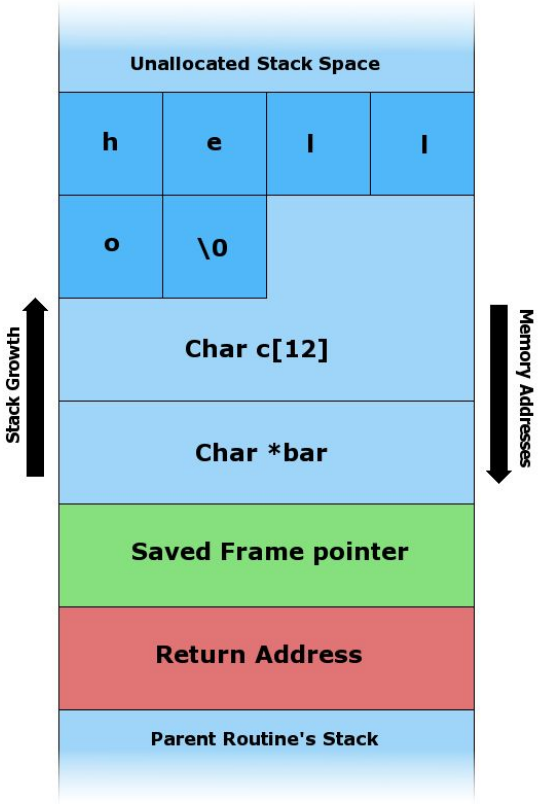
By overwriting a local variable (or pointer) of a different stack frame, which will be used by the function which owns that frame later

Stack Buffer Overflow

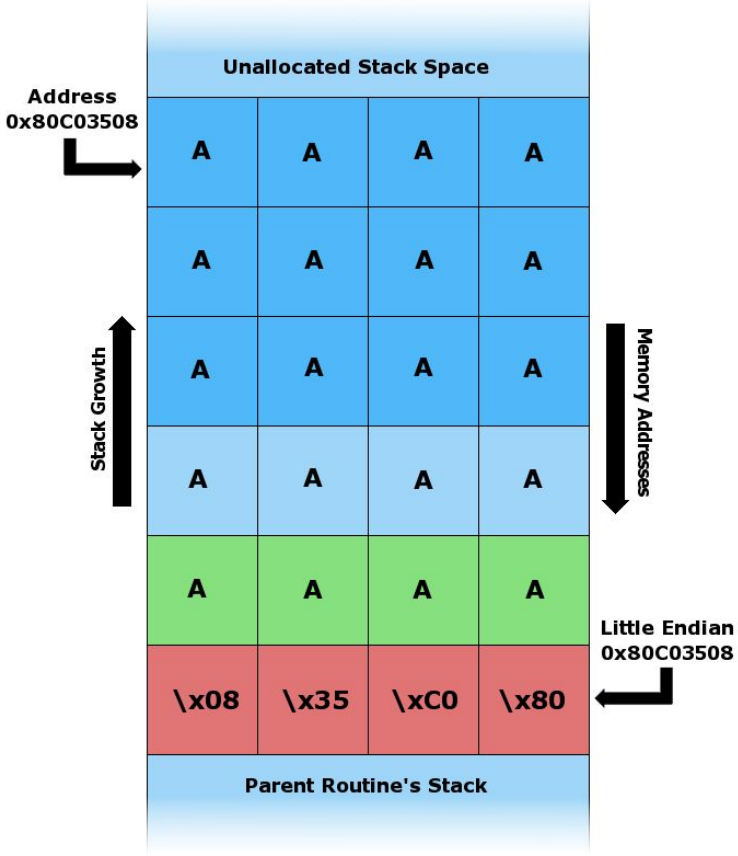
```c
#include <string.h>

void foo (char *bar)
{
   char  c[12];

   strcpy(c, bar);  // no bounds checking
}

int main (int argc, char **argv)
{
   foo(argv[1]);

   return 0;
}
```

Stack Buffer Overflow





Stack Buffer Overflow



Heap Overflow

Different from stack-based overflows.

Memory on the heap is dynamically allocated by the application at run-time and typically contains program data.

Exploitation is performed by corrupting this data in specific ways to cause the application to overwrite internal structures such as linked list pointers.

The canonical heap overflow technique overwrites dynamic memory allocation linkage (such as malloc metadata) and uses the resulting pointer exchange to overwrite a program function pointer

Read Overflow

Similar to writing pass the end of a buffer, read overflow could permit reading past the end of buffer leaking secret information

The heartbleed bug

- SSL Server should accept heartbeat message and echoes it back
- Heartbeat message specifies the length of its echo-back portion, but the buggy SSL software did not check whether the length was accurate
- Therefore an attacker was able to request longer length and read pass the contents of the buffer leaking passwords, crypto keys etc

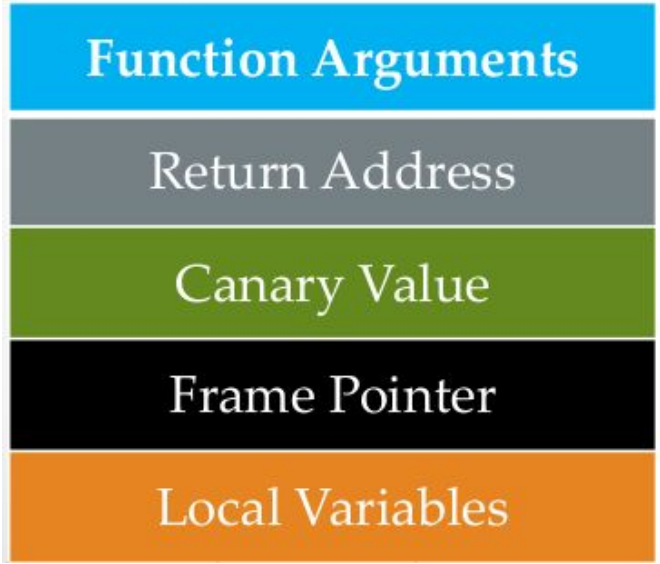
Counter Measures

Libsafe project

- Re-implemented functions like strcpy(), strcat() and gets()

Stack Guard

- Achieved by a compiler extension that adds so called canary values before the EIP saved at the function



Remote Code Execution (RCE)

```php
<?php
  if (isset($_GET['domain'])) {
    echo '<pre>';
    $domain = _GET['domain'];
    $lookup = system("nslookup {$domain}");
    echo($lookup);
    echo '</pre>';
  }
?>
```

Notice how the 'domain' parameter is taken in from the GET request, and immediately interpolated into a command string.

Questions

sdesapriya@gmail.com