COMPUTER SYSTEMS AND ORGANIZATION

Buffer Overflows

Daniel G. Graham Ph.D

1. Data as Code.
2. Why the gets function is insecure
3. Socket and Buffer Overflows

PLAYING WITH BYTES

```c
#include <stdio.h>
#include <stdint.h>

// Define a struct with three members shorts, char, char
typedef struct {
    short a;
    char b;
    char c;
} StructA;


int main() {
    // An array of hexadecimal values
    uint8_t hexArray[] = {0x12, 0x34, 0x56, 0x78};

    // Cast the array to StructA and access its members
    StructA *structA = (StructA*)hexArray;
    printf("StructA: a = 0x%x, b = 0x%x, c = 0x%x\n", structA->a, structA->b, structA->c);

    return 0;
}
```

WILL IT COMPILE?
WHAT WILL IT PRINT?

A. a = 0x3412, b = 0x56, c = 0x78
B. a = 0x1234, b = 0x56, c = 0x78
C. Doesn't Compile
D. Segmentation can't read from stack

```c
#include <stdio.h>
#include <stdint.h>

// Define a struct with three members shorts, char, char
typedef struct {
    short a;
    char b;
    char c;
} StructA;


int main() {
    // An array of hexadecimal values
    uint8_t hexArray[] = {0x12, 0x34, 0x56, 0x78};

    // Cast the array to StructA and access its members
    StructA *structA = (StructA*)hexArray;
    printf("StructA: a = 0x%x, b = 0x%x, c = 0x%x\n",
                    structA->a, structA->b, structA->c);

    return 0;
}
```

```
Home directory usage for /u/dgg6b: 1%
You have used 1.52G of your 100G quota

dgg6b@portal11:~$
```

```c
#include <stdio.h>
#include <stdint.h>

// Define a struct with three members shorts, char, char
typedef struct {
    short a;
    char b;
    char c;
} StructA;


int main() {
    // An array of hexadecimal values
    uint8_t hexArray[] = {0x12, 0x34, 0x56, 0x78};

    // Cast the array to StructA and access its members
    StructA *structA = (StructA*)hexArray;
    printf("StructA: a = 0x%x, b = 0x%x, c = 0x%x\n", structA->a, structA->b, structA->c);

    return 0;
}
```

WILL IT COMPILE?
WHAT WILL IT PRINT?

A. a = 0x3412, b = 0x56, c = 0x78
B. a = 0x1234, b = 0x56, c = 0x78
C. Doesn't Compile
D. Segmentation can't read from stack

DOES THIS WORK?

TWO STRUCTS ONLY ONE ARRAY

```c
#include <stdio.h>
#include <stdint.h>

// Define two different structs with short and char
typedef struct {
    short a;
    char b;
    char c;
} StructA;

typedef struct {
    char x;
    short y;
} StructB;

int main() {
    // An array of hexadecimal values
    uint8_t hexArray[] = {0x12, 0x34, 0x56, 0x78};

    // Cast the array to StructA and access its members
    StructA *structA = (StructA*)hexArray;
    printf("StructA: a = 0x%x, b = 0x%x, c = 0x%x\n", structA->a, structA->b, structA->c);

    // Cast the array to StructB and access its members
    StructB *structB = (StructB*)hexArray;
    printf("StructB: x = 0x%x, y = 0x%x\n", structB->x, structB->y);

    return 0;
}
```

```
 GNU nano 6.3                    dataAsCode.c
#include <stdio.h>
#include <stdint.h>

// Define two different structs with short and char
typedef struct {
    short a;
    char b;
    char c;
} StructA;

typedef struct {
    char x;
    short y;
} StructB;

int main() {
    // An array of hexadecimal values
    uint8_t hexArray[] = {0x12, 0x34, 0x56, 0x78};

    // Cast the array to StructA and access its members
    StructA *structA = (StructA*)hexArray;
    printf("StructA: a = 0x%x, b = 0x%x, c = 0x%x\n",
                    structA->a, structA->b, structA->c);

    // Cast the array to StructB and access its members
    StructB *structB = (StructB*)hexArray;
    printf("StructB: x = 0x%x, y = 0x%x\n",
                       structB->x, structB->y);

    return 0;
}
```

```
dgg6b@portal11:~$ clang dataAsCode.c
dgg6b@portal11:~$ ./a.out
StructA: a = 0x3412, b = 0x56, c = 0x78
StructB: x = 0x12, y = 0x7856
dgg6b@portal11:~$
```

DATA A CODE

ASSEMBLY AND THE ENCODING

```
int main(){
     return 7;
}
```

```
0000000000401108 <main>:
401108:   b8 07 00 00 00      mov  $0x7,%eax
40110d:   c3                  ret
```

```
#include <stdio.h>
#include <stdint.h>

// Define a function pointer type
typedef int (*func_ptr)();

int main() {

uint8_t code[] = {0xb8, 0x07, 0x00, 0x00, 0x00, 0xc3 };

    // Cast the array to a function pointer
    func_ptr func = (func_ptr)code;

    // Call the function through the pointer
    int result = func();

    printf("Function returned: %d\n", result);

    return 0;
}
```

DATA AS CODE

A. Doesn't compile can't cast array to function pointer
B. Runs but crashes the stack is not executable
C. Prints 7 if we compile in a special way
D. Prints Nothing

GNU nano 6.3 simple.c Modified

```c
#include <stdio.h>
#include <stdint.h>

// Define a function pointer type
typedef int (*func_ptr)();

int main() {

    uint8_t code[] = {0xb8, 0x07, 0x00, 0x00, 0x00, 0xc3 };

    // Cast the array to a function pointer
    func_ptr func = (func_ptr)code;

    // Call the function through the pointer
    int result = func();

    printf("Function returned: %d\n", result);

    return 0;
}
```

dgg6b@portal11:~$

```c
#include <stdio.h>
#include <stdint.h>

// Define a function pointer type
typedef int (*func_ptr)();

int main() {

uint8_t code[] = {0xb8, 0x07, 0x00, 0x00, 0x00, 0xc3 };

    // Cast the array to a function pointer
    func_ptr func = (func_ptr)code;

    // Call the function through the pointer
    int result = func();

    printf("Function returned: %d\n", result);

    return 0;
}
```

DATA AS CODE

A. Doesn't compile can't cast array to function pointer
B. Runs but Segfaults the stack is not executable
C. Prints 7 if we compile in a special way
D. Prints Nothing

BUFFER OVER HIGH LEVEL

nano text.c Normal use

nano*□□Àå□Ójo^□§ù¼□XM6¦õèÔÈ÷□í@□□.Å□□q+ýL□p>Ô□É2ìYÐ□□¿ exploit

nano 2a 7f 9b c0 e5 84 d3 6a 6f 5e 82 a7 f9 bc 0e 58 4d 36 a6 f5 e8 d4 c8 f7 03 ed 40 1c 93 2e c5 9d 01 71 2b fd 4c 8f 70 3e d4 01 c9 32 ec 59 d0 17 12 bf

EXAMPLE PROGRAM

```
#include <stdio.h>

void vulnerable_function() {
    char buffer[7];

    printf("Enter some text:\n");
    gets(buffer); // Unsafe use of gets()

    printf("You entered: %s\n", buffer);
}

int main() {
    vulnerable_function();
    return 0;
}
```

MAN PAGE OF GETS

```
NAME
       gets - get a string from standard input (DEPRECATED)

SYNOPSIS
       #include <stdio.h>

       char *gets(char *s);

DESCRIPTION
       Never use this function.

       gets() reads a line from stdin into the buffer pointed to by s
       until either a terminating newline or EOF, which it replaces
       with a null byte ('\0'). No check for buffer overrun is per-
       formed (see BUGS below).
```

Why: Because it doesn't Restrict the characters you Can enter.

HERE IS WHY LET'S THING ABOUT THE STACK

```
#include <stdio.h>

void vulnerable_function() {
    char buffer[7];
    gets(buffer);
}

int main() {
    vulnerable_function();
    return 0;
}
```

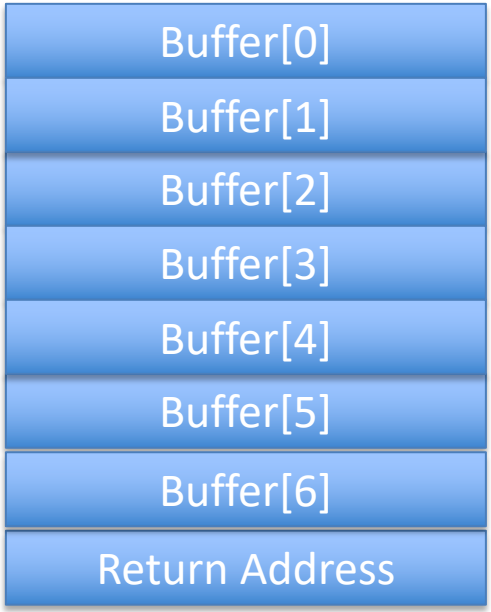
Return Address

HERE IS WHY LET'S THING ABOUT THE STACK

```
#include <stdio.h>

void vulnerable_function() {
    char buffer[7];
    gets(buffer);
}

int main() {
    vulnerable_function();
    return 0;
}
```

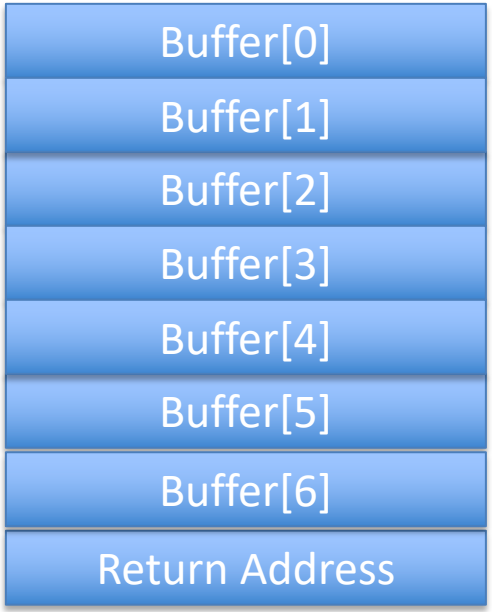


HERE IS WHY LET'S THING ABOUT THE STACK

```
#include <stdio.h>

void vulnerable_function() {
    char buffer[7];
    gets(buffer);
}

int main() {
    vulnerable_function();
    return 0;
}
```



What is we entered CS-2130

HERE IS WHY LET'S THING ABOUT THE STACK

```
#include <stdio.h>

void vulnerable_function() {
    char buffer[7];
    gets(buffer);
}

int main() {
    vulnerable_function();
    return 0;
}
```

What is we entered CS-2130

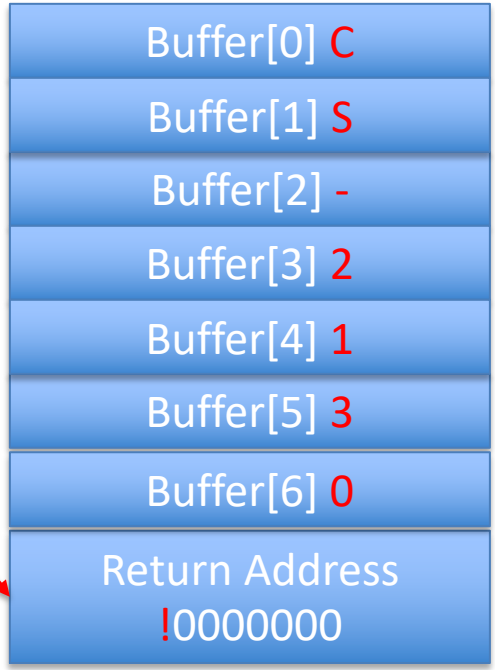
WHAT IF WE ENTERED 8 CHARS INSTEAD?

```
#include <stdio.h>

void vulnerable_function() {
    char buffer[7];
    gets(buffer);
}

int main() {
    vulnerable_function();
    return 0;
}
```

Override the first byte of return address



What is we entered CS-2130!

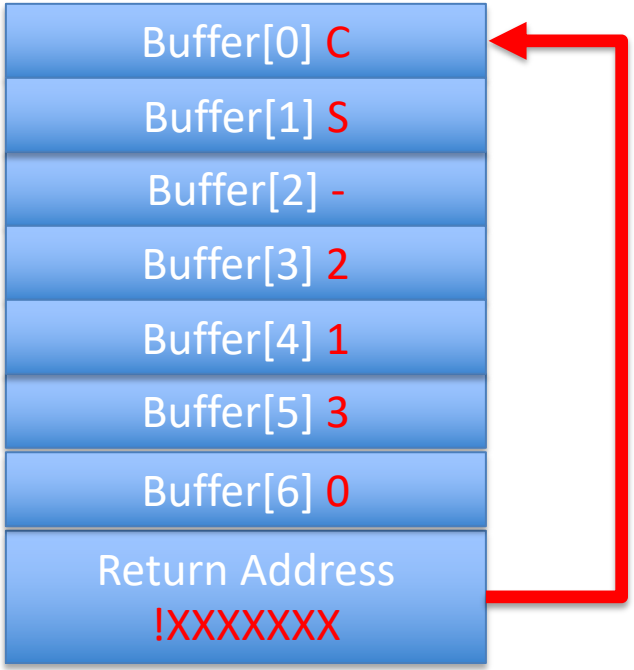
WHAT IF WE ENTERED 8 CHARS INSTEAD?

We have get program to execute any code that we place in the buffer.

Limitations:

- Buffer needs to be long enough
- Stack as to be executable (DEP turned off)
- Address Space randomization turned off. So that the attacker can determine the address of the buffer

What is we entered CS-2130!XXXXXXX



REMOTE BUFFER OVERFLOWS

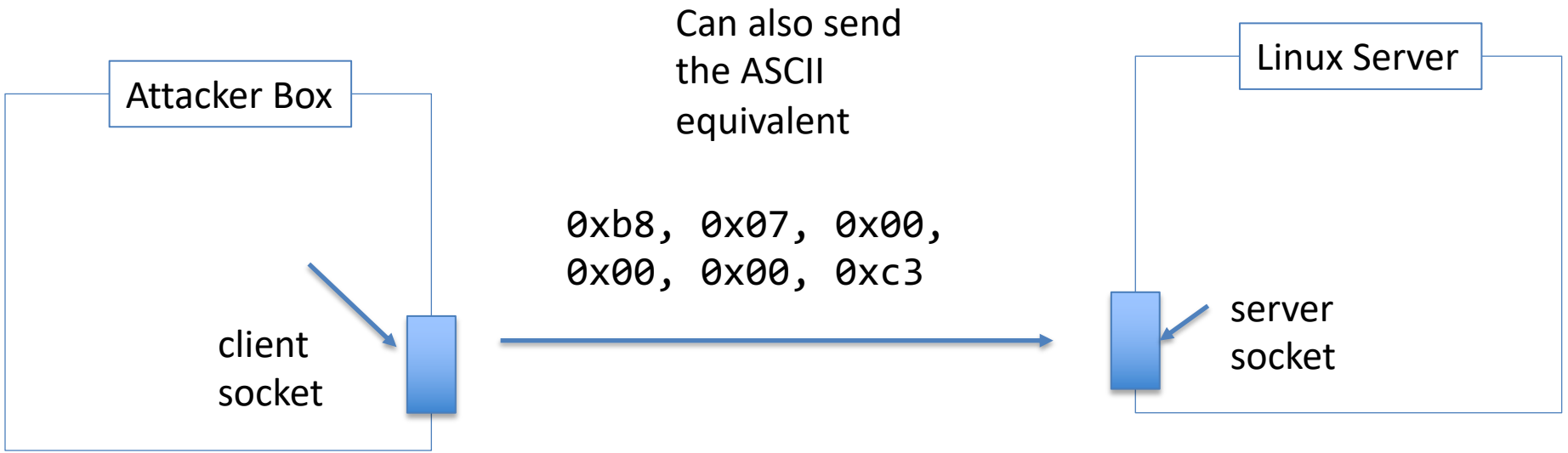
```
#include <stdio.h>

void vulnerable_function(int sockfd) {
    char buffer[7];
    fgets(s);
}

int main() {
    vulnerable_function();
    return 0;
}
```

Instead reading for standard in. If we read from the socket then an attacker can execute arbitrary code.

ATTACKER BOX



CAN YOU SPOT THE ERROR?

```
int copy_something(char *buf, int len) {
        char kbuf[800];
        if(len > sizeof(kbuf)) {
            return -1;
        }
        memcpy(kbuf, buf, len);
        return len;
    }
```

Chat with your neighbor?

```
extern void * memcpy(void *dst, const void *src, size_t n);
```

CAN YOU SPOT THE ERROR?

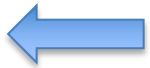
```
int copy_something(char *buf, int len) {
        char kbuf[800];
        if(len > sizeof(kbuf)) {
            return -1;
        }
        memcpy(kbuf, buf, len);
        return len;
    }
```

What if we pass in -1?

```
extern void * memcpy(void *dst, const void *src, size_t n);
```

CAN YOU SPOT THE ERROR?

```
int copy_something(char *buf, int len) {
        char kbuf[800];
        if(len > sizeof(kbuf)) {
            return -1;
        }
        memcpy(kbuf, buf, len);
        return len;
    }
```



What if we pass in -1? Passes this check so, it doesn't return -1

```
extern void * memcpy(void *dst, const void *src, size_t n);
```

CAN YOU SPOT THE ERROR?

```
int copy_something(char *buf, int len) {
        char kbuf[800];
        if(len > sizeof(kbuf)) {
            return -1;
        }
        memcpy(kbuf, buf, len);
        return len;
    }
```

What about this line?

```
extern void * memcpy(void *dst, const void *src, size_t n);
```

CAN YOU SPOT THE ERROR?

```
int copy_something(char *buf, int len) {
        char kbuf[800];
        if(len > sizeof(kbuf)) {
            return -1;
        }
        memcpy(kbuf, buf, len);
        return len;
    }
```

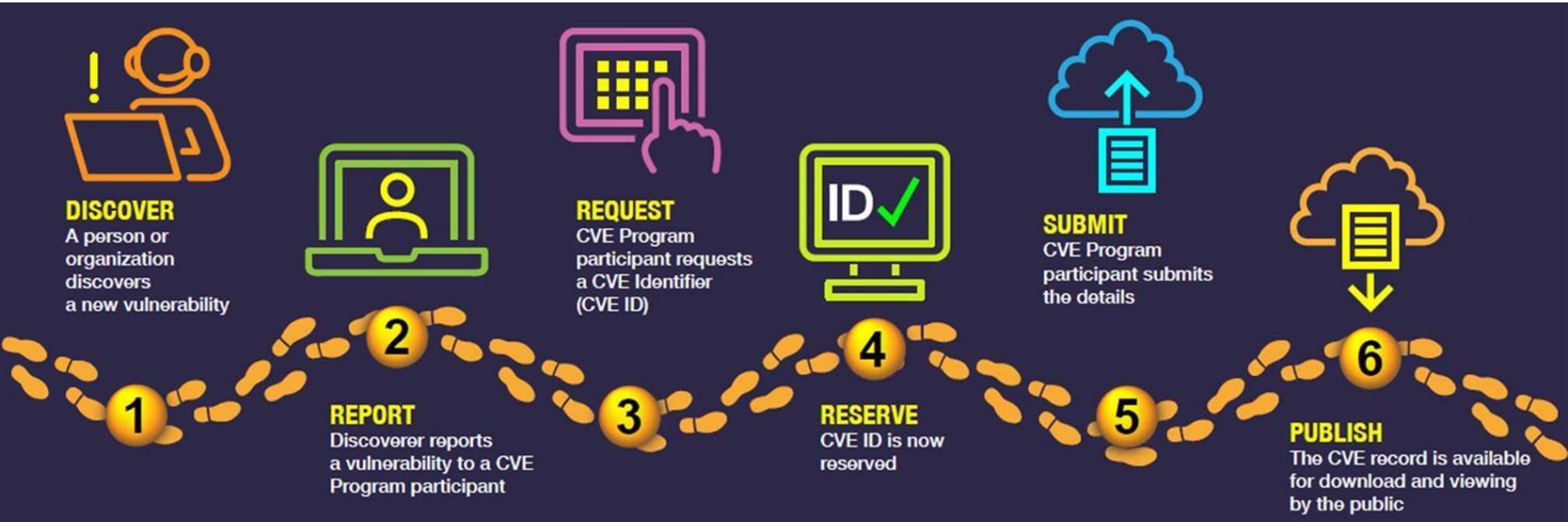
Type size_t is unsigned
So –1 is 0xffffff a large number so we end up copying more and overflowing the buffer

```
extern void * memcpy(void *dst, const void *src, size_t n);
```

NOTES ON ETHICS.

If you find something report it. https://www.cve.org/About/Process

Unless you are working for the Gov ☺ . Rules are little different there.



NOTES ON ETHICS.

If you find something report it. https://www.cve.org/About/Process

Unless you are working for the Gov ☺ . Rules are little different there.

P.S the it was 100k for finding and reporting the bug
And another 100k for proposing a fix.

LET'S LOOK AT AN BUFFER OFFERFLOW IN THE "WILD"

https://www.syncbreeze.com/about.html

https://www.exploit-db.com/exploits/49104