ARL South CyberRIG Exercise: Remote Exploitation

Dr. Jaime C. Acosta
Security awareness is essential

“Know your enemy and know yourself and you can fight a hundred battles without disaster.”

-Sun Tzu (Author of Art of War)
WARNING!

Never tamper without written consent
This can lead to academic suspension, legal prosecution, prevent you from getting a job, ...
Buffer Overflow

- Earliest buffer overflow exploit documented in 1988
  - Exploited Finger Program
  - Started at MIT and was “used to determine the size of the Internet”
  - Infected 6000 Unix machines
- 1996, Phrack Magazine
  - Provides a step by step guide
- NIST Stats:
  - 1996: 8 vulnerabilities
  - 2008: 339 vulnerabilities
  - 2012: 418 vulnerabilities
  - 2015: 344 vulnerabilities
Buffer Overflow - ExerciseCode.c

```c
#include <stdio.h>

int i; //assume an integer is 4 bytes

void doCopy(char *name)
{
    char copy[8];
    for(i=0; name[i] != '\0'; i++)
        copy[i] = name[i];
    copy[i] = '\0';
}

void main()
{
    char input[16]; //assume each char is 1 byte
    printf("Enter input\n");
    fgets(input, 16, stdin); //parameters pushed in reverse order
doCopy(input);
    printf("done copying: ");
    printf(input);
}
```
#include <stdio.h>

int i; //assume an integer is 4 bytes

void doCopy(char *name)
{
    char copy[8];
    for(i=0; name[i] != '\0'; i++)
    {
        copy[i] = name[i];
        copy[i] = '\0';
    }
}

void main()
{
    char input[16]; //assume each char is 1 byte
    printf("Enter input\n");
    fgets(input, 16, stdin); //parameters pushed in reverse order
    doCopy(input);
    printf("done copying: ");
    printf(input);
}
# Buffer Overflow - ExerciseCode.c

```c
#include <stdio.h>

int i; //assume an integer is 4 bytes

void doCopy(char *name)
{
    char copy[8];
    for(i=0; name[i] != '\0'; i++)
        copy[i] = name[i];
    copy[i] = '\0';
}

void main()
{
    char input[16]; //assume each char is 1 byte
    printf("Enter input\n");
    fgets(input, 16, stdin); //parameters pushed in reverse order
    doCopy(input);
    printf("done copying: ");
    printf(input);
}
```
C Program

```c
void main()
{
    char input[16];  //assume each char is 1 byte
```
C Program

```c
void main()
{
    char input[16];  // assume each char is 1 byte
}
```

Allocates 16 bytes of memory on the **stack**
C Program

```
void main()
{
    char input[16]; //assume each char is 1 byte
```

Allocates 16 bytes of memory on the stack

Heap calls the allocator explicitly (new, malloc, calloc....)
C Program

```
printf("Enter input\n");
fgets(input, 16, stdin);
```
C Program

Print data to screen

Using an external library (we did not implement printf)

printf("Enter input\n");

fgets(input, 16, stdin);

External library function

destination

Maximum size of input (anymore will be ignored)

Source (terminal input)
C Program

```c
    doCopy(input);
    printf("done copying: ");
    printf(input);
```
C Program

```c
17
18
19
20
}
docopy(input);
printf("done copying: ");
printf(input);
```

- Call the `user` function
- Print to screen with external library
C Program

```c
void doCopy(char *name)
{
    char copy[8];
}
```
C Program

```
void doCopy(char *name)
{
    char copy[8];
    //allocate
    Push 0x800 000c
    Mem
    "????"
    Input parameter
    Pointer to character
    "address of location where this array starts"
```

void doCopy(char *name)
C Program

```c
for (i=0; name[i] != '\0'; i++)
    copy[i] = name[i];

copy[i] = '\0';
```
C Program

Copy all characters from name[ ] to copy[ ]

Then add null "terminator"
Is there a problem?

```c
#include <stdio.h>

int i; // assume an integer is 4 bytes

void doCopy(char *name)
{
    char copy[8];
    for(i=0; name[i] != '\0'; i++)
        copy[i] = name[i];
    copy[i] = '\0';
}

void main()
{
    char input[16]; // assume each char is 1 byte
    printf("Enter input\n");
    fgets(input, 16, stdin); // parameters pushed in reverse order
doCopy(input);
    printf("done copying: ");
    printf(input);
```
#include <stdio.h>

int i; //assume an integer is 4 bytes

void doCopy(char *name)
{
    char copy[8];
    for(i=0; name[i] != '\0'; i++)
        copy[i] = name[i];
    copy[i] = '\0';
}

void main()
{
    char input[16]; //assume each char is 1 byte
    printf("Enter input\n");
    fgets(input, 16, stdin); //parameters pushed in reverse order
    doCopy(input);
    printf("done copying: ");
    printf(input);
}
Compilation...

$ i686-w64-mingw32-gcc exerciseCode.c -o exerciseCode.exe
Compilation...

Source code

A windows c compiler

Output file

(Results are the same w/ cygwin’s gcc)
Exercise