Buffe Assignment Project Exam Help

https://eduassistpro.github.io/

A simple function

```
void f() {
  Assignment Project Exam Help int buf[9
  int i;
              https://eduassistpro.github.io/
  for (i=0; Aidd Wochit edu_assist_pro
    buf[4+i] = buf[4-i] = 0;
```

A simple function

```
void f() {
  Assignment Project Exam Help int buf[9
  int i;
              https://eduassistpro.github.io/
  for (i=0; Aidd WbChat edu_assist_pro
    buf[4+i] = buf[4-i] = 0;
```

The call stack

• A data structure that stores information about function calls in a program ssignment Project Exam Help

Bottom of stack

- In X86 the stackhttps://eduassistpro.github.io/
 - The stack bottom is at a high address Add WeChat edu_assist_pro

Top of stack

- The stack top is at a low address
- The stack grows towards lower addresses

Implementation

- Register %esp points to the top of the sessignment Project Exam Help
- The push instr https://eduassistpro.github.io/value onto the

• pop pops a value popl %eax

Calling a function

- Calling a function pushes a stack fram Signment Broject Exam Help
 - The stack base (%ebp) points https://eduassistpro.github.io/the current funcaidd WeChat edu_assistachro-
- Return pops the stack frame

%ebp

%esp

Calling conventions

Caller's stack Caller does: %ebp frame • Save registersignment Project %esp Saved Registers %esp uments Push argument %esp https://eduassistp critiladdress Call function %esp %ebp %espp, %ebp Add WeChat edu_assist_pro Callee does ariables • Save %ebp %esp • Set new %ebp Create space for local variables

Example

```
int q(int a, int b) {
 int x = a + 1;
                                      g:
 int y = b + 2;
                                          pushl
                                                  %ebp
             Assignment Project Examb Help *esp, *ebp
                                              $16, %esp
 return x*y;
                   https://eduassistprol.github.log.
                                                   %eax, -8(%ebp)
                   Add WeChat edu_assist_pfo(%ebp), %eax
                                                  $2, %eax
                           %esp
       Return address
                                                   %eax, -4(%ebp)
                                          movl
                           %esp
        Saved %ebp
                                          \mathtt{movl}
                                                   -8(%ebp), %eax
                           %esp, %ebp
                                          imull
                                                   -4(%ebp), %eax
                                          leave
                                          ret
```

%esp

Back to a simple function

```
Return address
void f() {
                                          Saved %ebp
           Assignment Project Exam Help
  char buf[9];
                https://eduassistpro.github.io/
                                                0
  for (i=0; i Add Wethat edu_assist_pro
    buf[4+i] = buf[4-i] =
                                                0
```

With a minor change

```
5
void f() {
  int i;
             Assignment Project Exam Help
  char buf[9];
                   https://eduassistpro.github.io/
                                                         5
  for (i=0; i < 10; i++)
buf[4+i] = buf[4] WeChat edu_assist_pro
                                                         5
5
```

Stack smashing

```
void f()
  char buf[512];
  gets (buf Assignment Project Exam Help
                                              Return address
  doSomething
                 https://eduassistpro.github.io/
• The attacker divadd WeChat edu_assist_pro
                                                  buf
 data it injected

    How does the attacker know

                                             gets stack frame
 where to jump to?
```

NOP Sled

• A sequence of NOP Assignment Project Exam Help instructions

to the attack

https://eduassistpro.github.io/

NOP



Problem patterns

Any use of gets

```
• strcpy, sprintf, strcat, etc.

sprintf(buf, "https://ss/index.html"pargv[1])

Assignment Project Exam Help

buf=new char[

strcpy(buf, ahttps://eduassistpro.github.io/

wchar_t buf[MAXLEN];
swprintf(buf, AddeWfeChat edu_assist_vpro);
```

Any low-level implementation of similar code

```
while (*src != ';')
  *dst++ = *src++;
*dst = '\0';
```

Avoiding buffer overflows

- Do not use gets.
- Replace unsafe C string functions with safe version
 Redefine unsafe functions to catch use, for example:

```
er *src) {char *str https://eduassistpro.github.io/
            abort () ;Add WeChat edu_assist_pro
```

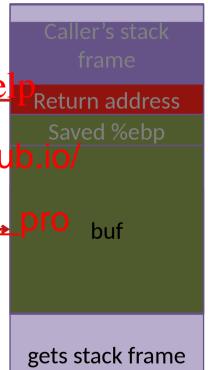
- May fail if library functions use strcpy
- Replace C strings with safe(r) C++ strings

Avoiding buffer overflows - 2

- Abstract over array access to include bounds checking
 - For example, use the C++ vector .at() method
 - What about seignment Project Exam Help
- Code reviews a https://eduassistpro.github.io/
- Use static code and We Chat edu_assist_pro
- Switch to Java, C#, etc.

Non-executable stacks

- The stack is only used for data. There's no need to run code from the stack
- The memory management unit can Exam Help revent code exe address https://eduassistpro.github.io/
- Only protects against arwelthat edu_assist, prothe stack
- Does not prevent:
 - Heap overflow
 - Return Oriented Programming



ROP Illustrated

Assignment Project Exam Help

https://eduassistpro.github.io/

StackGuard

- - On function entry, callee
 - Saves %ebp
 - Sets new %ebp Assignment Project Exam Help
 - Pushes the canary
 - Creates space for local
 - Verify the canary on

https://eduassistpro.github.io/turn address

Add WeChat edu_assist_pro

- The attacker has to overwrite the canar changing the return address
- There are ways around the canary
- Does not protect from heap overflows, changing function pointers, etc.

Caller's stack frame

Saved Registers

Arguments

Saved %ebp

Canary

Local Variables

STack Overwrite Protection

- Push a large buffer to the stack at process initialisation
- The attacker Advise moternt de discret le la return address https://eduassistpro.github.io/

• A large enough Nath Welchat edu_assist_pro

- negligible probability of a success
- Only protects the stack
 - ASLR (Address Space Layout Randomization) extends protection to the heap and to libraries

Summary

- Buffer overflow is a common vulnerability

- There's no silv https://eduassistpro.github.io/