University of BRISTOL

SysSoft Security

A very quick Introduction to Stack Buffer Overflow

Sanjay.rawat@Bristol.ac.uk

bristol.ac.uk

Lecture Agenda

- Buffer Overflow, in general
  - Stack Overflow
  - Heap Overflow (more details in other lecture)
- Code patterns leading to such bugs.
- Book Ref.
  - *The Craft of System Security*, by Sean Smith; John Marchesini. Chapter 6, section 6.1
  - *24 DEADLY SINS OF SOFTWARE SECURITY- Programming Flaws and How to Fix Them*. Section 5.
  - SANS report: https://www.sans.org/reading-room/whitepapers/threats/buffer-overflows-dummies-481

Buffer Overflow

- Several decades old problem (still appears in SANS TOP 25 Software errors!!)
- Main cause: putting more data than *intended*!!
- Consequences: memory corruption (can be very dangerous!)

Buffer Overflow

- Several decades old problem (still appears in SANS TOP 25 Software errors!!)
- Main cause: putting more
- Consequences: memory c y dangerous!)

Stack based BoF

- Cause
  - Stack grows downward
  - Local buffers are allocated onto the stack
  - With no memory protection, these variables can overflow!
- Effect- security vulnerability
  - At CALL, return address is saved on the stack
  - Return address is POPed into the RIP
  - RIP can point to anywhere in the memory!

Side effects

Side effects

- Over/underflow

Side effects

- Over/underflow
- Sensitive data corruption

Side effects

- Over/underflow
- Sensitive data corruption
-

Side effects

- Over/underflow
- Sensitive data corruption
-

If done properly-exploit

Side effects

- Over/underflow
- Sensitive data corruption
-

If done properly-exploit

Otherwise crash!

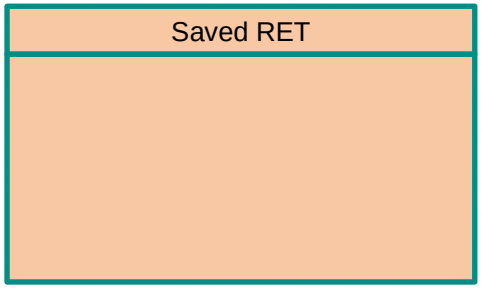
Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```

main

Example

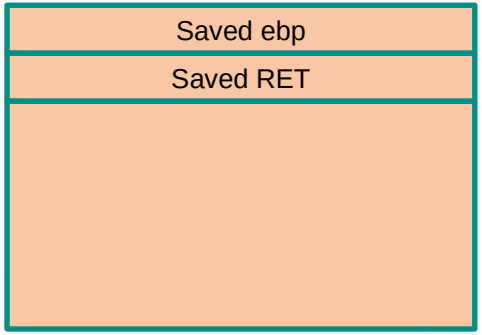
```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```



main

Example

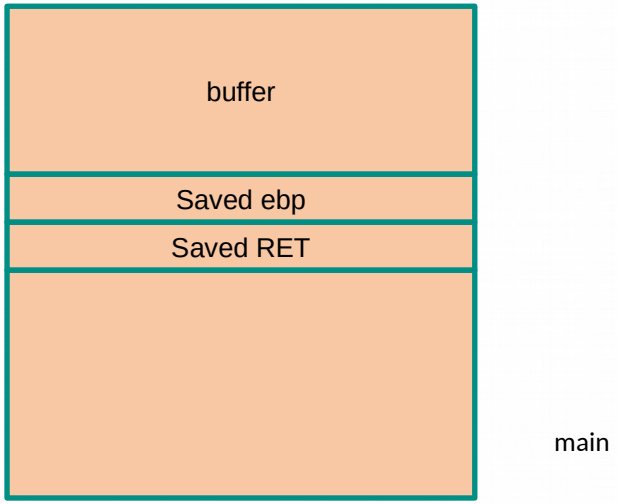
```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```



main

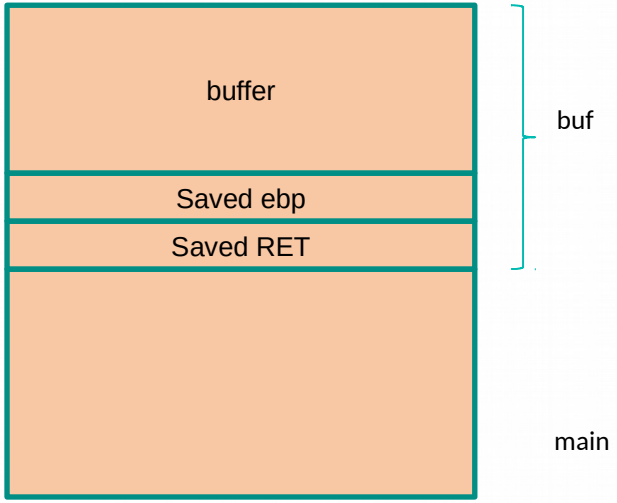
Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```



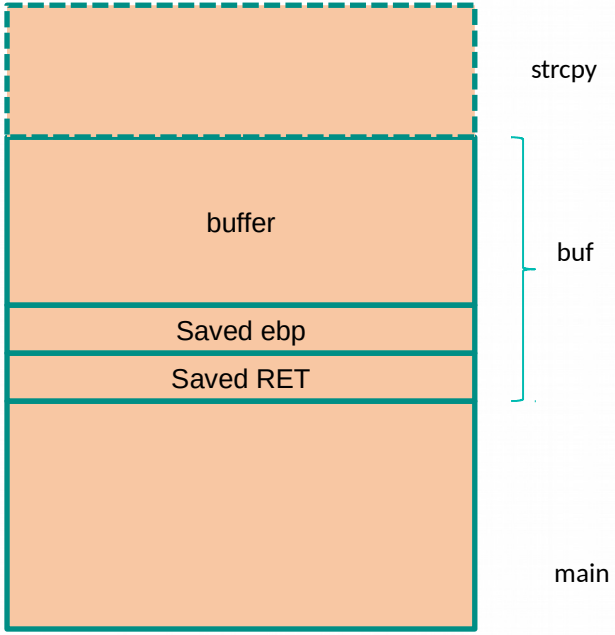
Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```



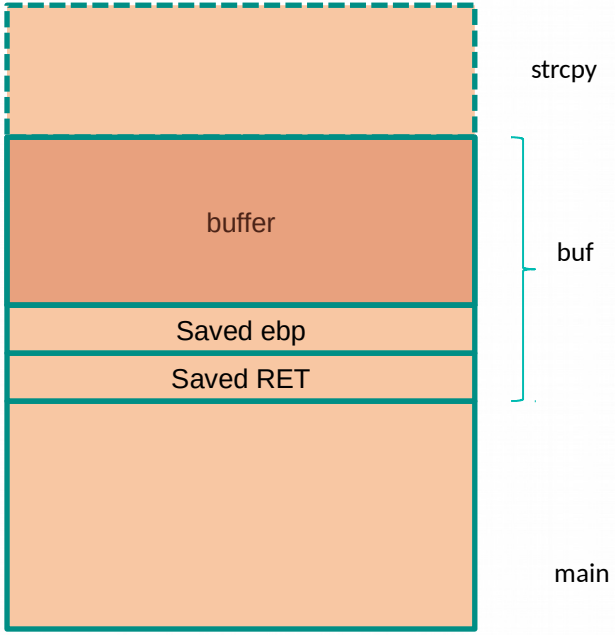
Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```



Example

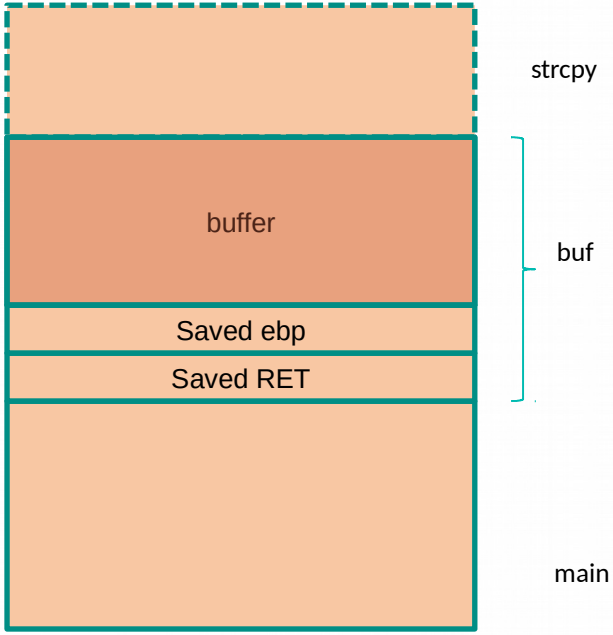
```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```



Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```

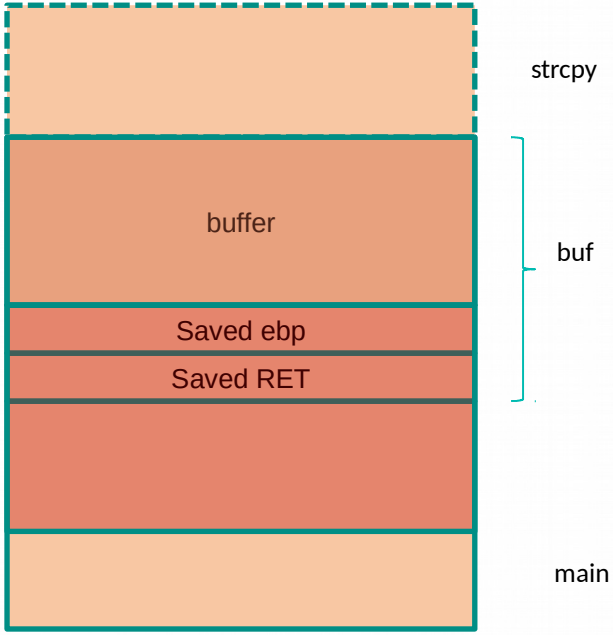
String > buffer??



Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```

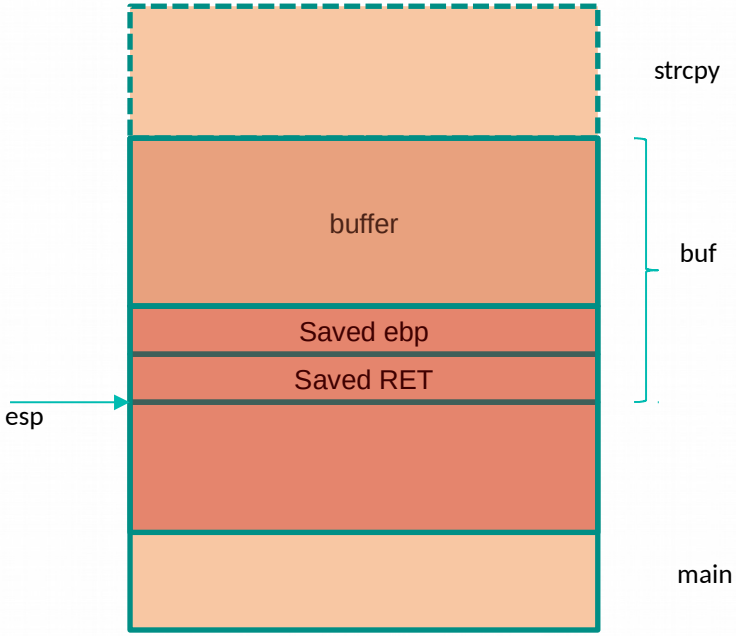
String > buffer??



Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```

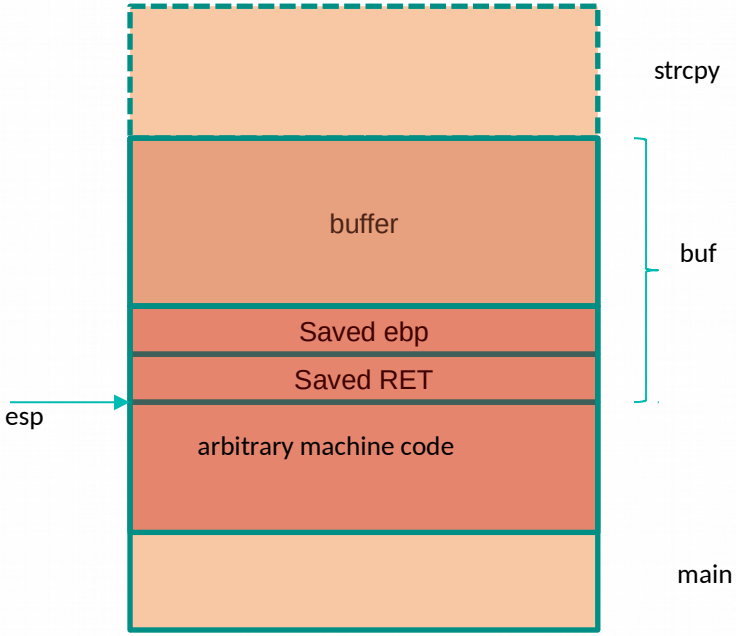
String > buffer??



Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```

String > buffer??

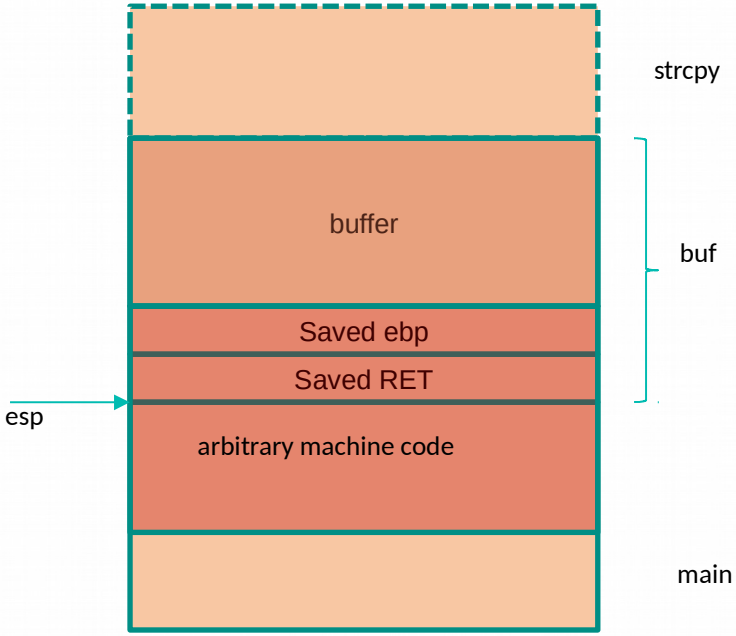


Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```

String > buffer??

Find address to jmp esp (XYZ)

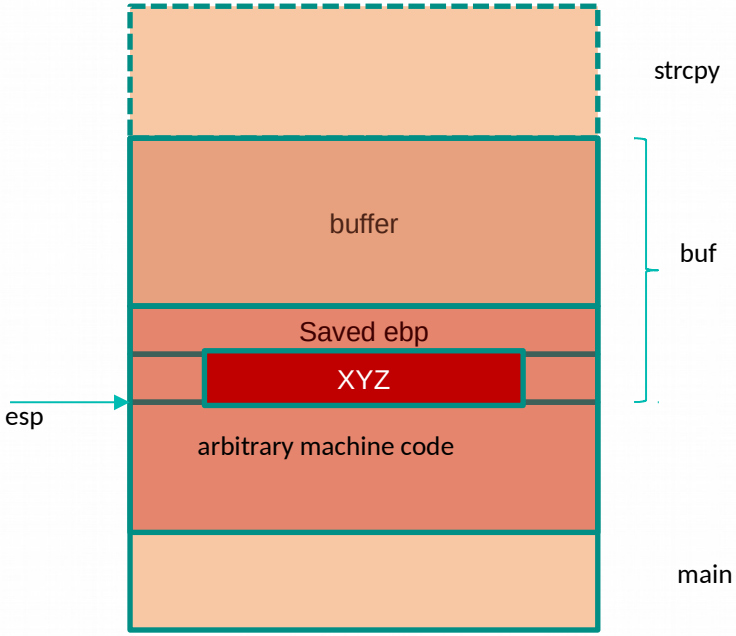


Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```

String > buffer??

Find address to jmp esp (XYZ)

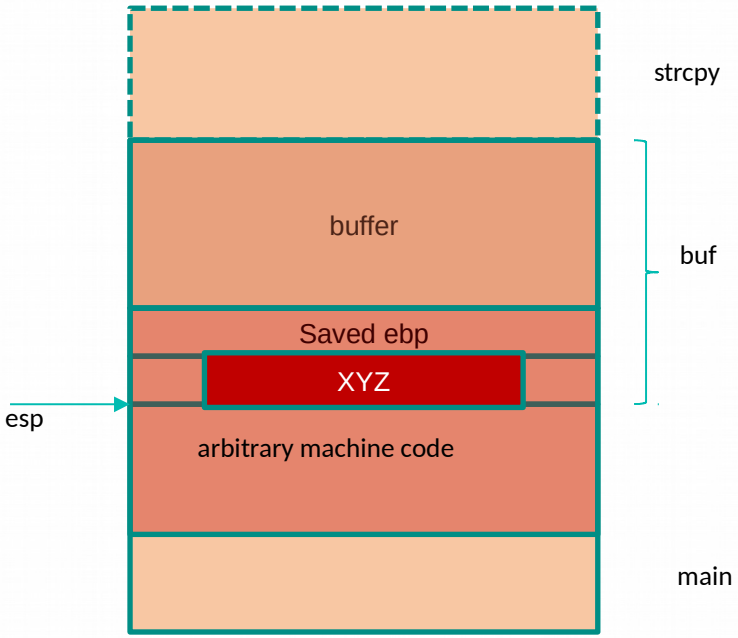


Example

```
buf(char *string) {
    char buffer[20];
     strcpy(buffer,
     string);
       return 1;
        }
```

String > buffer??

Find address to jmp esp (XYZ)



Payload = junk data + return addr overflow + shellcode

What can be done? Exploit conditions

What can be done? Exploit conditions

1. We should be able to overflow the buffer to inject our shellcode (Machine code that we want to execute) onto the stack.

What can be done? Exploit conditions

1. We should be able to overflow the buffer to inject our shellcode (Machine code that we want to execute) onto the stack.
2. The shellcode has overflowed in such a way that out of all general purpose registers, at least one register is pointing to shellcode's beginning.

What can be done? Exploit conditions

1. We should be able to overflow the buffer to inject our shellcode (Machine code that we want to execute) onto the stack.
2. The shellcode has overflowed in such a way that out of all general purpose registers, at least one register is pointing to shellcode's beginning.
3. We should be able to overwrite the EIP to some memory location whose address is predictable and it does not throw error (e.g. access violation).

What can be done? Exploit conditions

1. We should be able to overflow the buffer to inject our shellcode (Machine code that we want to execute) onto the stack.
2. The shellcode has overflowed in such a way that out of all general purpose registers, at least one register is pointing to shellcode's beginning.
3. We should be able to overwrite the EIP to some memory location whose address is predictable and it does not throw error (e.g. access violation).
4. There exits an instruction at that location that will allow us to access/jump to the register pointing towards our shellcode

Regarding Exploits