Application Security

Today: Florin Stancu

Asst. Prof. Mihai Chiroiu

SYSTEMS LABORATORY

Computer Science & Engineering Department

Contents

- Computer Vulnerabilities
  - Cause & classification
  - Memory safety bugs
- State of the Art
  - Eternal War in Memory presentation

Vulnerability

- *A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy*

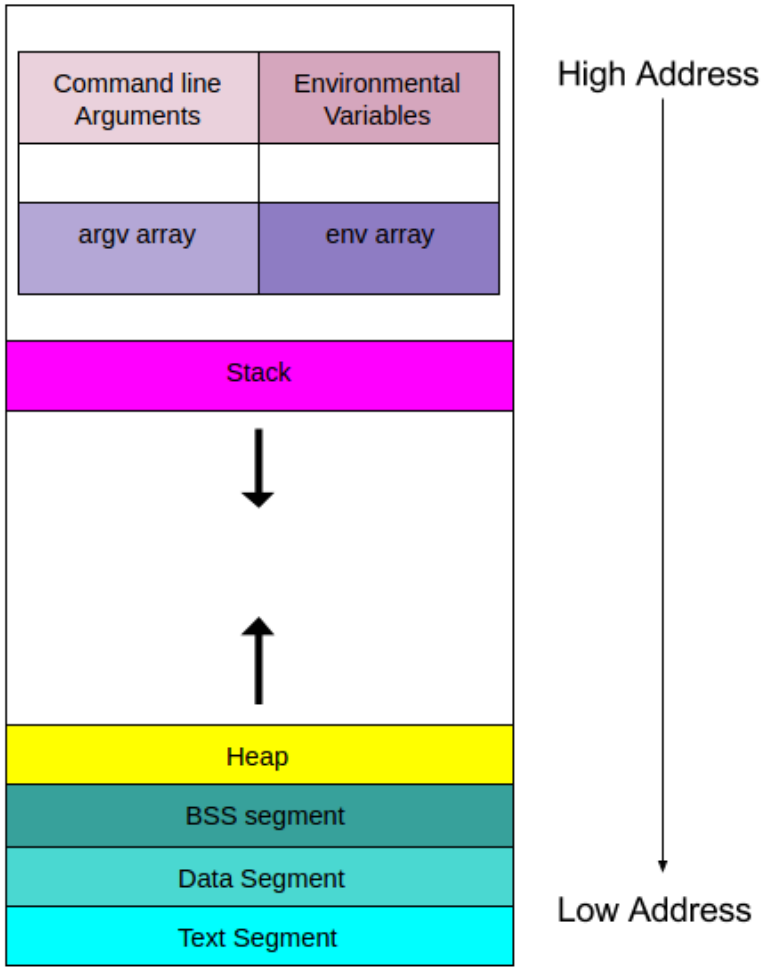
- **Scope:** hardware, **software**, network, personnel, physical, organizational

Software vulnerabilities

- **Memory safety**: buffer overflow, dangling pointer, race condition, memory leak etc.
- Input validation: code injection, format string attacks, path traversal…
- Side channel attacks
- UI confusion
- Privilege escalation
- And many more!

Intro: address space

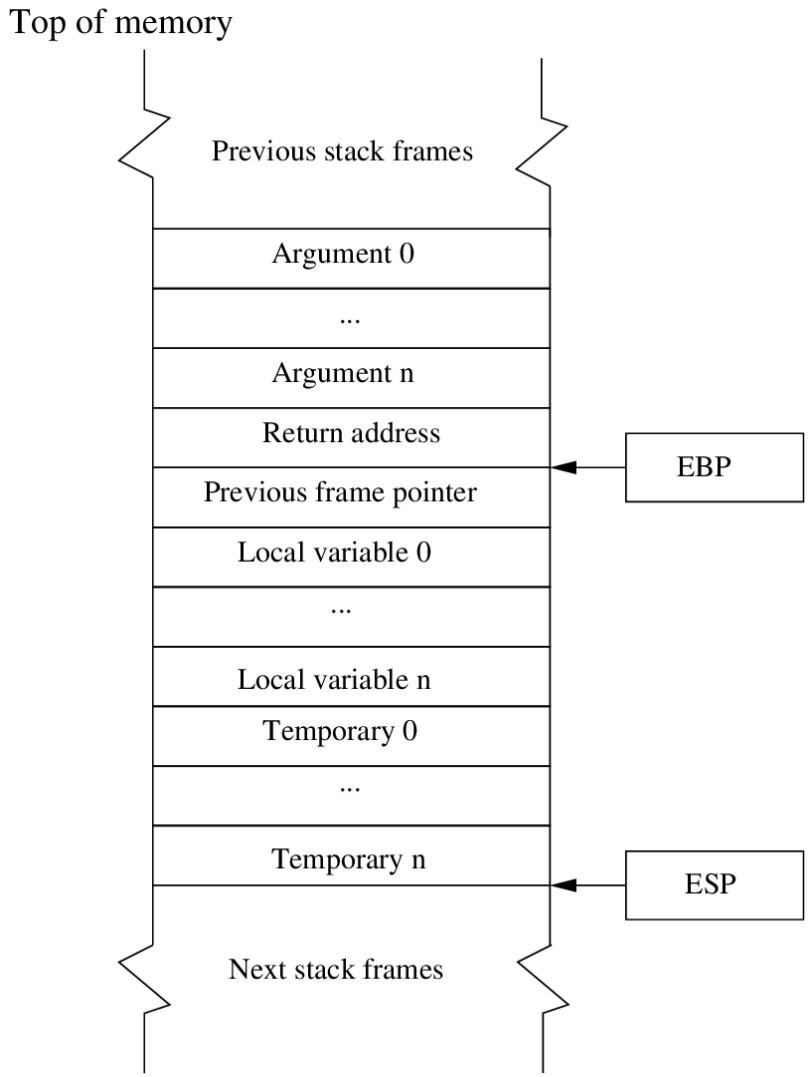
- Userspace processes have virtual memory
- Compilers (linker) decide where each segment goes.
- Address space layout is pretty standard



Intro: stack frame

- Stack grows downward (x86)

- Contains function arguments, saved CPU state (registers, instruction / frame pointers) and local variables.

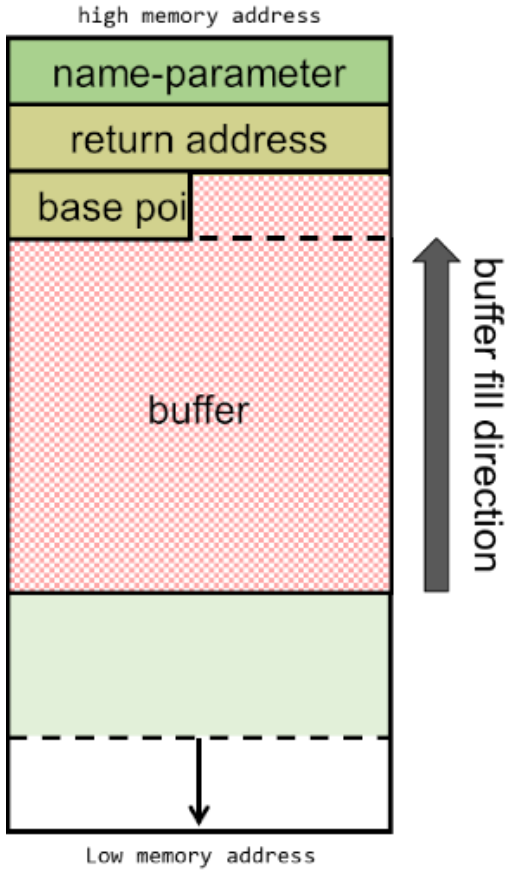


Buffer overflow (stack)

- Happens when a buffer's is written after its allocated size.

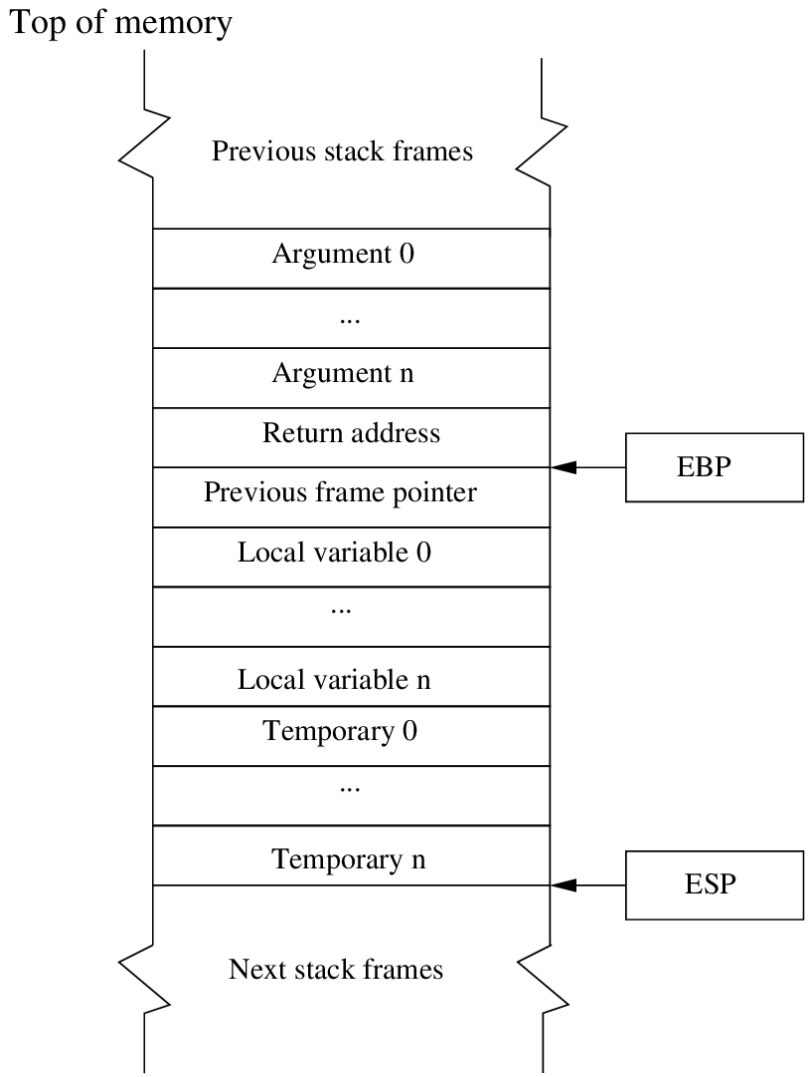
```
char buf[10];
char *input = "This text is larger than
expected";

strcpy(buf, input);
```



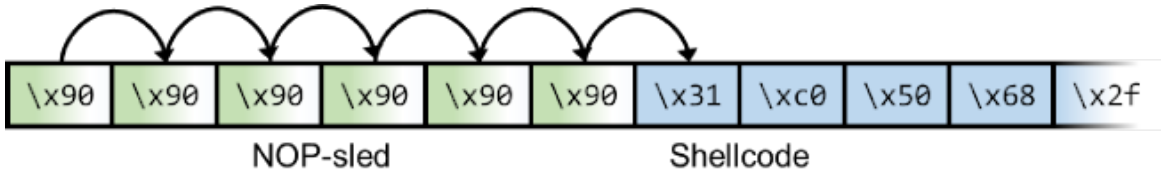
Buffer overflow (2)

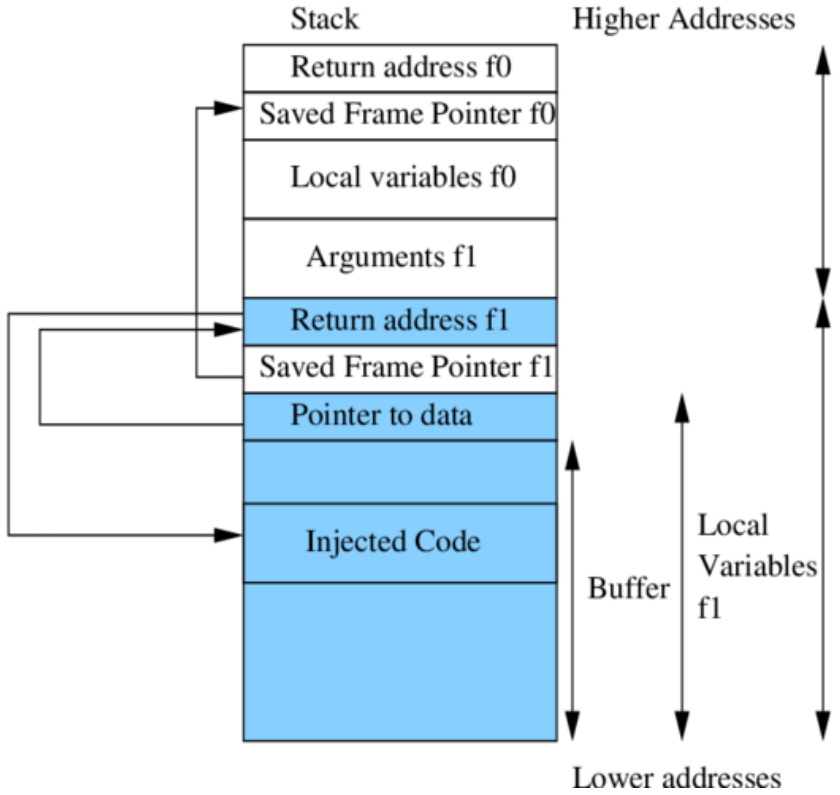
```
(gdb) disas func
Dump of assembler code for function func:
   0x0804841b <+0>:     push   %ebp
   0x0804841c <+1>:     mov    %esp,%ebp
   0x0804841e <+3>:     sub    $0x64,%esp
   0x08048421 <+6>:     pushl  0x8(%ebp)
   0x08048424 <+9>:     lea    -0x64(%ebp),%eax
   0x08048427 <+12>:    push   %eax
   0x08048428 <+13>:    call   0x80482f0 <strcpy@plt>
   0x0804842d <+18>:    add    $0x8,%esp
   0x08048430 <+21>:    lea    -0x64(%ebp),%eax
   0x08048433 <+24>:    push   %eax
   0x08048434 <+25>:    push   $0x80484e0
   0x08048439 <+30>:    call   0x80482e0 <printf@plt>
   0x0804843e <+35>:    add    $0x8,%esp
   0x08048441 <+38>:    nop
   0x08048442 <+39>:    leave
   0x08048443 <+40>:    ret
End of assembler dump.
```



Buffer overflow (3)

- ret instruction will pop the return address from the stack, then jump to it.
- CPU will execute the injected code (shellcode).
- NOP sled when the address is not fixed:





Buffer overflow mitigation

- DEP (data execution prevention) / No-execute (NX) bit
  - *defeated by ROP (return oriented programming)*

- Address space layout randomization
- Stack Canaries
  - *defeated by memory leakage, side channels etc.*

- Control flow integrity

Return oriented programming