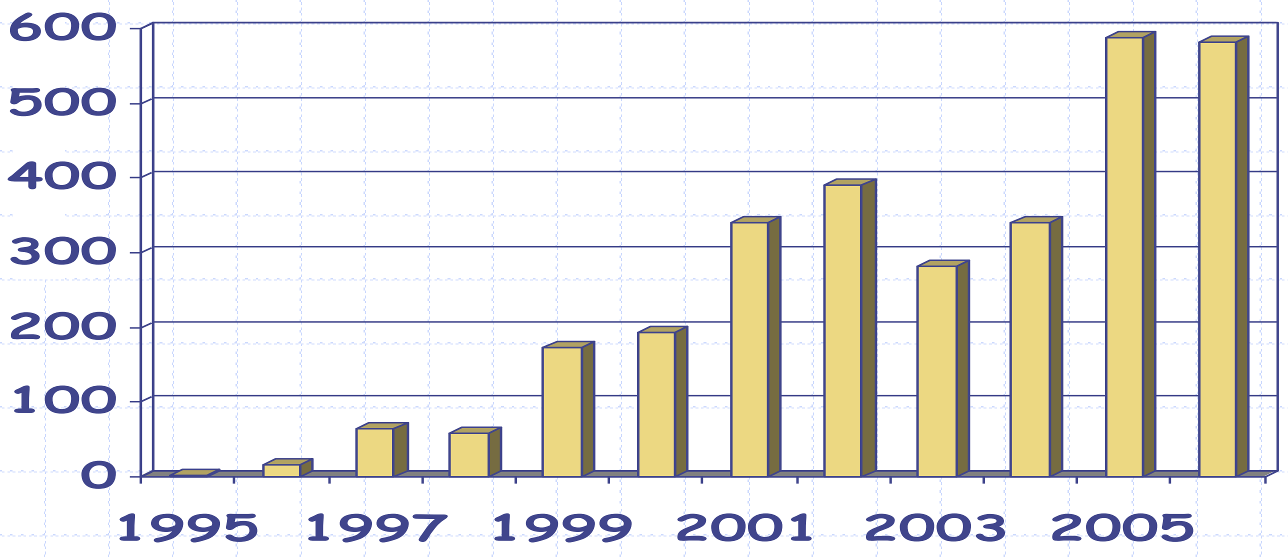
Buffer overflows (slides courtesy: http://cs.uccs.edu/~cs591/bo2.ppt)

- Extremely common bug.
  - First major exploit: 1988 Internet Worm. fingerd.



≈20% of all vuln.

2005-2007: ≈ 10%

Source: NVD/CVE

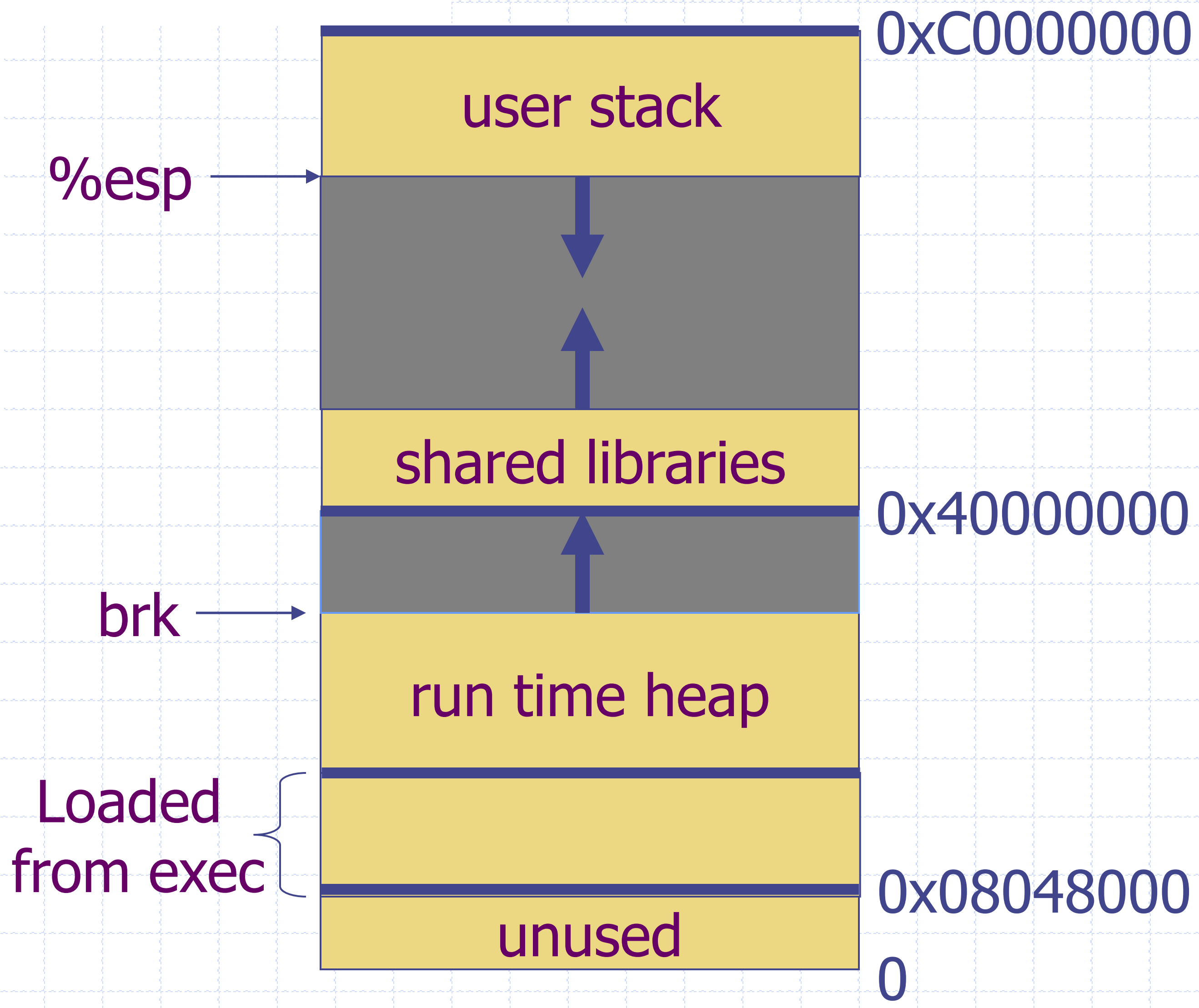
- Developing buffer overflow attacks:
  - Locate buffer overflow within an application.
  - Design an exploit.

What is needed

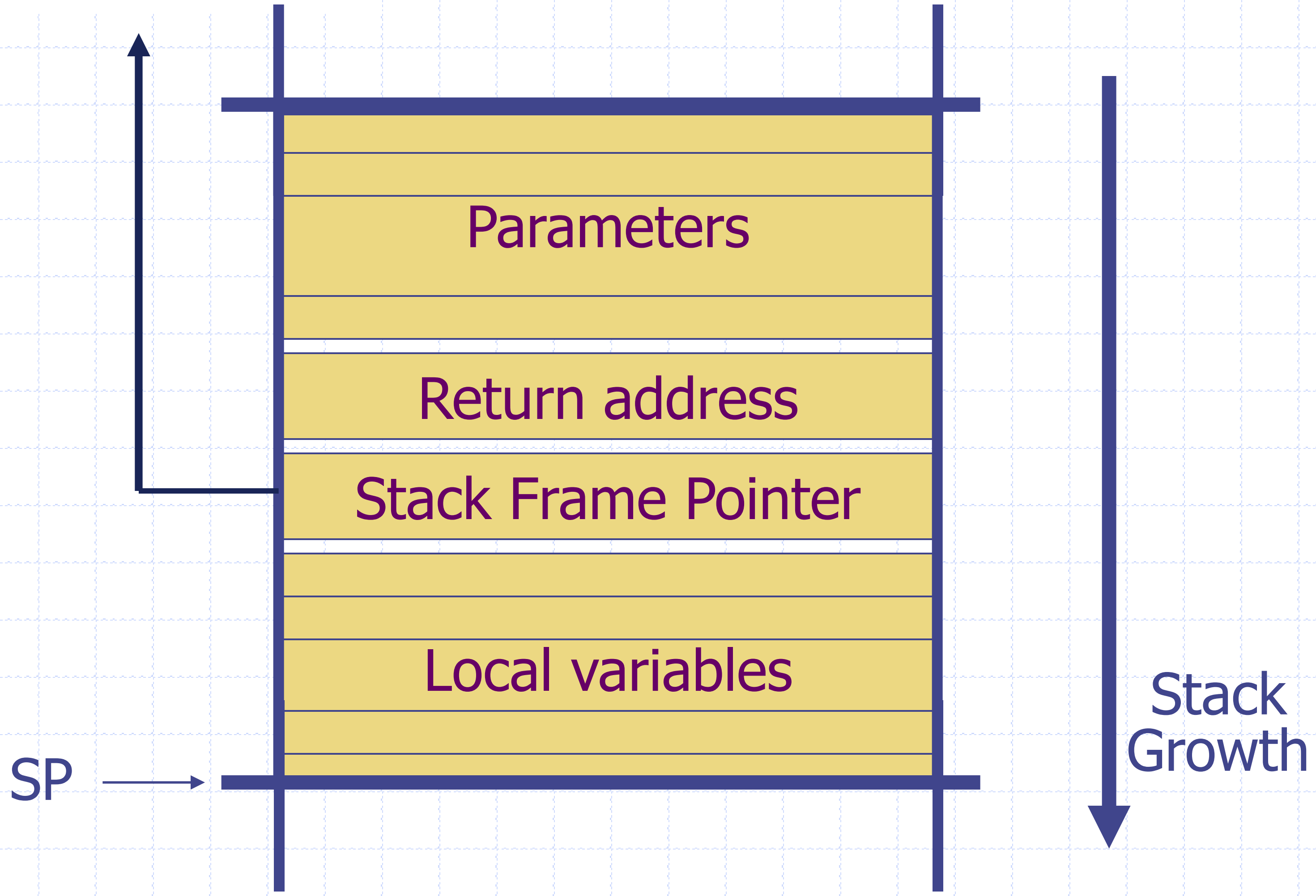
- Understanding C functions and the stack
- Some familiarity with machine code
- Know how systems calls are made
- The exec() system call

- Attacker needs to know which CPU and OS are running on the target machine:
  - Our examples are for x86 running Linux
  - Details vary slightly between CPUs and OSs:
    - Little endian vs. big endian (**x86 vs. Motorola**)
    - Stack Frame structure (Unix vs. Windows)
    - Stack growth direction

Linux process memory layout



Stack Frame


Parameters
Return address
Stack Frame Pointer
Local variables
SP
Stack Growth

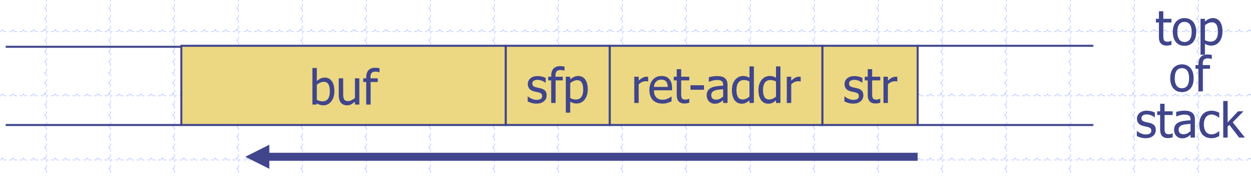
What are buffer overflows?

- Suppose a web server contains a function:

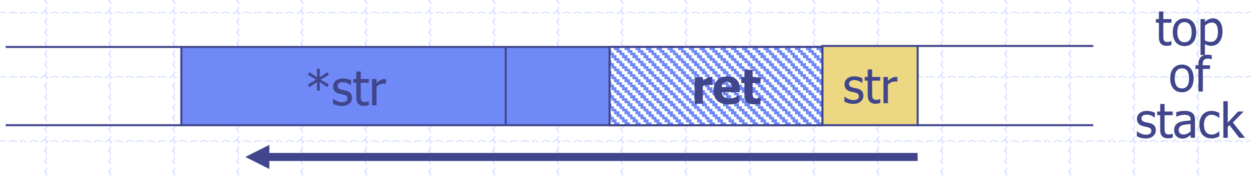
```
void func(char *str) {
   char buf[128];

   strcpy(buf, str);
   do-something(buf);
}
```

- When the function is invoked the stack looks like:

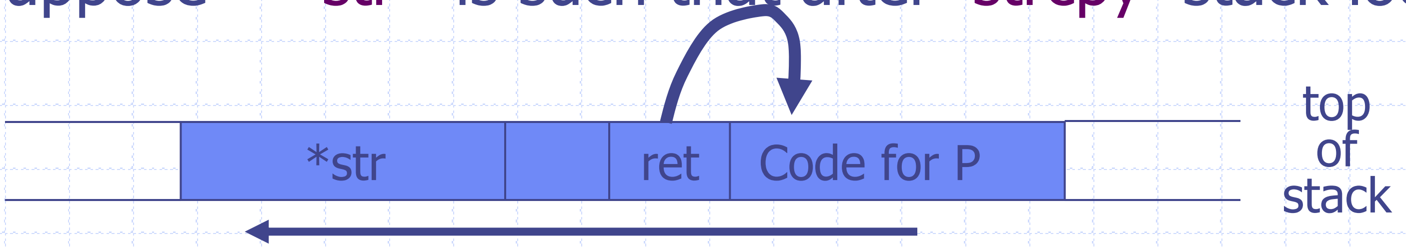


- What if `*str` is 136 bytes long? After `strcpy`:



Basic stack exploit

- Problem: no range checking in strcpy().
- Suppose *str is such that after strcpy stack looks like:



Program P: `exec( "/bin/sh" )`

(exact shell code by Aleph One)

- When func() exits, the user will be given a shell !
- Note: attack code runs *in stack*.

- To determine ret guess position of stack when func() is called

Many unsafe C lib functions

```
strcpy (char *dest,  const char *src)
strcat (char *dest, const char *src)
gets (char *s)
scanf ( const char *format, … )
   ⋮
```

- “Safe” versions strncpy(), strncat() are misleading
  - strncpy() may leave buffer unterminated.
  - strncpy(), strncat() encourage off by 1 bugs.

Exploiting buffer overflows

- Suppose web server calls func() with given URL.
  - Attacker sends a 200 byte URL. Gets shell on web server
- Some complications:
  - Program P should not contain the '\0' character.
  - Overflow should not crash program before func() exists.
- Sample remote buffer overflows of this type:
  - (2005) Overflow in MIME type field in MS Outlook.
  - (2005) Overflow in Symantec Virus Detection

```
Set test = CreateObject("Symantec.SymVAFileQuery.1")
test.GetPrivateProfileString "file", [long string]
```