Associated Press, 2004.09.16:

“Security breach clears Oakland airport

“Oakland International Airport was evacuated and all flights were grounded for about an hour Thursday night after a suspicious item passed through a security checkpoint, authorities said.

“About 8:50 p.m., an airport screener saw a ‘threatening image’ on his video monitor from an item that had passed through an X-ray machine. When airport officials could not match the image to a bag or passenger, they evacuated both terminals and ground all departing flights around 9 p.m.”

Assignment due 2004.09.03: read Gaim.
`http://cr.yp.to/2004-494/gaim.html`

Assignment due 2004.09.08: read textbook Chapter 7 pages 277–308.

Assignment due 2004.09.15: read textbook Chapter 7 pages 309–336.

Assignment due 2004.09.17: read textbook Chapter 7 pages 360–366.

Assignment due 2004.09.20: read libpng.
`cr.yp.to/2004-494/libpng.html`

```
#define NOCHAR -1
register int c;
for (;;) {
  c = *p++;
  if (...)
    *q++ = ’\\’;
  ...
  if (c != NOCHAR)
    if (q > ...)
      break;
}
```

How do we know that `*q` is inside array?
The `q > ...` tries to check—but only
if c != −1. Can c be set to −1?
Byte `*p` is 0 through 255, right?
Not exactly! Actually −128 through 127.

```
m(...,char **x,...,int xlen)
{
  int nchar = 0;
  while (...) {
    ...
    if (++nchar > xlen) break;
    *(*x)++ = ...;
  }
}
char obuf[MAXLINE + 1];
char *obp = obuf;
while (...)
  m(...,&obp,...,MAXLINE);
```

`m` can write to `(*x)[0]`,
`(*x)[1]`, ..., `(*x)[xlen-1]`;
i.e., `obp[0]`, ..., `obp[MAXLINE-1]`.
How do we know these are inside `obuf`?

`obuf[0]`, ..., `obuf[MAXLINE]`
are all okay. Isn't `obp` equal to `obuf`?

Not necessarily!
`obp` starts out equal to `obuf`,
but `m` changes `*x`, i.e., changes `obp`.

The second call to `m` can overflow `obuf`.

Which writes are buffer overflows?

`*p = x` may be an overflow.

Typically `p` started out
pointing to the beginning of an array,
but was then increased or decreased.
How far was it moved?
How long is the array?

If `*p = x` is protected by
adjacent tests that `p >= thearray`
and `p < thearray + itslength`,
and if we're sure about `itslength`,
then there's clearly no buffer overflow.

Similarly: a[n] = x, same as
*(a + n) = x, may be an overflow.

How big is n? How long is a?

If a[n] = x is protected by
adjacent tests that n >= 0
and n < a + itslength,
and if we're sure about itslength,
then there's clearly no buffer overflow:

```
   int a[30];
   int n;
   ...
   if (n >= 0)
     if (n < 30)
       a[n] = j;
```

```
while (*tz != ’\0’)
  *q++ = *tz++;
```

Question you should be asking:

Is `q` buffer longer than `tz`?

```
if (first >= tTsize)
  first = tTsize - 1;
tTvect[first] = i;
```

Questions you should be asking:

What if `first` is negative?

Is `tTsize` the size of `tTvect`?

```
readdata(buf);
```

Question you should be asking:

Does `readdata` know how long `buf` is?

How serious is a buffer overflow?

You've found a write that can overflow a buffer in a program.

Does this bug allow an input source to take control of the program?
Is that source controlled by an attacker?

Example: `p = buf; ... p = 0; *p = 3` always crashes. No worse effects.

Example:
`myreadfile("/usr/src/README",buf)`
might overflow `buf` with data from the `/usr/src/README` file, but that file can't be affected except by the system administrator.

Finding new buffer overflows

`www.sourceforge.net`
has many free programs.

I decided to download `latex2rtf`.
You're not allowed to use `latex2rtf`
for your homework.

Let's look at `www.sourceforge.net`
and then look at `latex2rtf`.