USA Today, 2004.10.08:

“Tests uncover lax security at Newark

“Security screeners at Newark Liberty International Airport, one of the airports breached by the Sept. 11 hijackers, failed to detect 25% of fake bombs or weapons in inspection tests, a report said Thursday.

“The Newark Star-Ledger, citing confidential inspection reports, said the vast majority of failures resulted from the federal screeners’ inability to detect phony explosive devices hidden in carry-on bags sent through X-ray machines.

“They also missed some guns in carry-on luggage or concealed under the clothing of
inspectors as they walked through metal detectors at the airport near New York.

“A total of 327 tests were conducted at the airport’s nine checkpoint areas. Screeners succeeded 246 times and failed 81 times for a failure rate of 24.8%, according to the TSA documents cited by the newspaper.

‘‘We’re working diligently to increase our explosive detection capabilities at our passenger checkpoints,’ said Mark Hatfield, a TSA spokesman. ‘The key point here—testing is training.’ ”
The printing problem

A university system administrator creates accounts for thousands of students and faculty members.

(The system administrator is someone authorized to control the entire computer; e.g., the owner.)

Computer has a laser printer. Any picture written to /dev/ulpt0 is sent directly to the laser printer.

System administrator wants to allow people with accounts to use the printer. How does he do it?
First try at a solution

Each file has **owner** and **permissions**. Owner is allowed to change permissions.

/dev/ulpt0 is owned by root, the system administrator. Normal permissions: 600, meaning other users can’t open /dev/ulpt0.

System administrator runs command

```
chmod 622 /dev/ulpt0
```
changing permissions to 622. This allows all users to write data to /dev/ulpt0.

Now, to print, a user simply copies a picture to /dev/ulpt0.
Security holes

An unscrupulous student (or maybe a faculty member?) prints thirty copies of a book, consuming all the printing resources.

As revenge, another student opens `/dev/ulpt0` at the same time and writes random garbage, ruining twenty copies of the book.

System administrator decides to limit access to the printer:
only 500 pages per user;
only one print job at a time.
How does he do it?
Second try at a solution

System administrator writes a printing program, lpr, and makes it available to everyone:

```
vi lpr.c
gcc -o lpr lpr.c
cp lpr /usr/bin/lpr
```

System administrator tells users to print using this program:

```
lpr < mypicture
```
lpr looks up the user’s home directory, say /home/joe, and creates a new file
/home/joe/.pagesprinted containing the number 1.

If /home/joe/.pagesprinted already exists, lpr increments the number in it. If the number reaches 500, lpr exits.

lpr then opens /dev/ulpt0, applies flock to /dev/ulpt0, and copies its input to /dev/ulpt0.

What does flock do? It waits until any previous programs that used flock have closed /dev/ulpt0. (“Exclusive advisory lock.”)
Security holes

Users can skip the lpr program and write directly to /dev/ulp0t0. Setting permission back to 600 would make lpr fail.

Joe can also change /home/joe/.pagesprinted from 500 back to 1, or simply remove it.

Users can also run a separate program that flock /dev/ulp0t0 and waits forever, making lpr fail for everyone else.
Third try at a solution

System administrator changes \texttt{1pr} to make a TCP connection to port 515, send username, send picture to be printed.

System administrator runs
\begin{verbatim}
tcpserver 0 515 lpd &
\end{verbatim}
so that any TCP connection to port 515 runs \texttt{lpd} as root and talks to it.

\texttt{lpd} reads user’s account name, say \texttt{joe}, from the connection; handles \texttt{/etc/lpd/joe/pagesprinted}; and copies input to \texttt{/dev/ulpt0}, making sure not to wait forever.

System administrator sets permissions 600 for \texttt{/dev/ulpt0} and \texttt{/etc/lpd/\*/\*}. 
Security holes

Joe makes a TCP connection, sends name Bill, sends picture.

Spammer in China connects to port 515, sends name Bill, sends an ad.

lpd has no idea who it’s talking to. It blindly trusts username controlled by an attacker.

How can lpd figure out who it’s talking to?
Fourth try at a solution

System administrator turns off the network service and keeps permissions at 600.

System administrator changes `lpr` to directly handle `/dev/ulpt0` and `/etc/lpd/joe/pagesprinted`.

`lpr` doesn’t have permission to access those files—until system administrator turns `lpr` into a **setuid program**:

```
chmod 4755 /usr/bin/lpr
```
What happens when the owner of a program makes it setuid?

That program runs as the owner, rather than as whichever user started the program.

Lower level: When a process execve's a setuid program, the process owner (the “uid”) changes to the program's owner.

So lpr runs as root. It can write to /dev/ulpt0 and /etc/lpd/joe/pagesprinted, even though Joe can't.
Security holes

Setuid lpr can be secure, but only if it’s written very, very, very carefully.

Local attacker has many ways to control a setuid program: fds, args, environ, cwd, tty, rlimits, timers, signals, etc. Even worse, this list varies between Linux, BSD, Solaris, etc.

Writing a program that handles all of these channels safely is much more difficult than writing a program that handles a single input channel safely.