Doreen Hemlock, Fort Lauderdale Sun Sentinel, 2004.10.21:

"FedEx chief stresses need for real, virtual security in business

"Calling security a top priority for business, FedEx Corp. Chief Executive Fred Smith appealed Wednesday in Miami Beach for tough legislation against e-mail tampering, identity theft and other offenses on the Internet. . . .

"Governments need to penalize breaches online, from hackers who plant viruses in computers to those who pose as banks to seek private information by e-mail from bank clients, he said.

“While it’s a federal offense to tamper with U.S. Postal Service mail, there are no tough penalties for tampering with e-mail, Smith told the Cargo Facts 2004 conference.”

Course grade:

60% homework.

10% midterm 1.

10% midterm 2, probably 17 November.

20% final.

Need 85% for A, 75% for B, etc.

Another setuid security hole

Sendmail bug fixed 1996.11.17:

```
    execv(argv[0],argv);
```

What is this? Why is it a bug?

When Sendmail starts,
it reads several configuration files.
Sendmail can run for days
handling thousands of messages.
What if configuration changes?

User can tell Sendmail
to re-read configuration.
How does Sendmail do this?
By restarting itself.

On some UNIX systems,
Sendmail is `/usr/lib/sendmail`.
On others, `/usr/sbin/sendmail`.

Normally the name is in `argv[0]`.
Sendmail calls

```
    execv(argv[0],argv)
```

which eventually does
`execve("/usr/lib/sendmail",...)` or
`execve("/usr/sbin/sendmail",...)`.

Unfortunately for Sendmail,
`argv[0]` can be changed
by whoever started Sendmail—
any user on the system.

Joe calls

```
    execve("/usr/lib/sendmail"
       ,{"/home/joe/evil",...}
       ,{...})
```

to run `/usr/lib/sendmail`
with arguments `/home/joe/evil` etc.

Because `/usr/lib/sendmail`
is setuid (4755) 0 (owned by `root`),
this process now has uid 0.

Sendmail now runs `argv[0]`,
i.e., `/home/joe/evil`.
Process still has uid 0.

Joe's `/home/joe/evil` program
now controls the entire computer:
it can read and write any user's file.

Another setuid security hole

Bug announced 2004.08 by Max Vozeler.

`/dev/cdrom` reads CD-ROMs,
reads and writes CD-RWs.

`cdrecord` is a setuid program
so that it can write to `/dev/cdrom`.

It can also log into another computer
to record a CD on that computer:

```
   cdrecord \
   dev=REMOTE:djb@x:1,0,0 -
```

`RSH` environment variable
specifies remote-login program.
“Use e.g. `RSH=/usr/bin/ssh`
to create a secure shell connection.”

Joe runs

```
    env RSH=/home/joe/evil \
    cdrecord \
    dev=REMOTE:x:1,0,0 -
```

`cdrecord` is setuid 0,
and runs `/home/joe/evil`.
Joe's `/home/joe/evil` program
now controls the entire computer.

Fix: Before calling `execve`,
`cdrecord` calls

```
    setuid(getuid());
```

to set uid to real uid,
i.e., switch back to Joe's uid.

Note: setuid program; `setuid` syscall.

Does `setuid(getuid())`
really give up all extra powers
obtained by a setuid program?
Not necessarily!

1. For programs setuid to non-`root`,
Linux and Solaris allow process to undo
`setuid(getuid())`. (BSD doesn't.)

Say `cd` user owns `/dev/cdrom`
and `cdrecord` is setuid `cd`.

`cdrecord` calls `setuid(getuid())`
and then `execve`'s `/home/joe/evil`.
`evil` undoes `setuid(getuid())`
and now can write to `/dev/cdrom`,
destroying or modifying next user's CD.

Linux kernel bug, fixed 2000:
Joe could disable `setuid()`
even for setuid-root programs,
easily taking over through (e.g.) Sendmail.
How?

As a “security” mechanism,
Linux invented new system data:
process can disable its ability
to perform various syscalls.

In particular, process can disable
the `setuid()` syscall. Oops!
Joe does this, runs Sendmail.

(Actually disabled the ability
for `setuid()` to set “saved uid.”
Setting saved uid prevents undo.)