`nbc4.tv`, 2004.11.19:

"Two people who worked at LAX blew the whistle on what appeared to be serious security problems. They went to NBC4, they say, only after first trying to warn authorities. Now one of the people who helped expose these problems says the airport is trying to drive him out of business.

" 'I've been targeted. I've been challenged by them. I've been discriminated,' said Aamir Chishty, who runs Aeroex Corporation, a company that contracts with airlines to handle baggage at LAX.

“In NBC4’s first report Chishty said there were private companies handling baggage at LAX without the proper permits and security badges and accessing doors marked restricted even though signs say ‘ID badge required.’

“So NBC4 sent a producer undercover to the Tom Bradley International terminal to investigate. He was able to walk in and out of doors marked restricted and gain access behind ticket counters to conveyor belts with screened luggage. He also got into the back areas where he found most bags sitting unguarded—leaving it possible for someone to plant something in a bag.

“Most of the time nobody questioned him to see if he had a security badge. . . .

“Since NBC4’s story aired it appears neither the airport nor the airlines have taken any action regarding the doors—many of them remain wide open. But they have taken action against Aeroex by confiscating their security badges.”

Continuing homework:
Find security holes!

Your targets:
9 holes per person
by the end of this week,
10 holes per person
by the end of next week.

Homework procrastinators
are likely to fail the course.

Memory allocation

Several syscalls tell kernel
to set aside extra memory.

e.g. `brk()` and `sbrk()` syscalls,
used by `malloc()` and `realloc()`,
need memory for process RAM.

e.g. `fork()` syscall
needs memory for new process,
as large as old process.

e.g. `execve()` syscall,
if running a larger program,
needs memory for that program.

Wasted allocation

Sometimes memory is allocated
but never actually used.

e.g. Sloppy program
allocates 1-megabyte buffer
but uses only 37 bytes.
(Fix: allocate only
the memory you need.)

e.g. Process calls `fork()`;
child process calls `execve()`
of a much smaller program.
Temporarily uses much more memory.
(Fix: use better syscalls,
`vfork()` or `posix_spawn()`.)

Copy-on-write

Memory is divided into
4096-byte **pages**.

When one page of memory is
created as a copy of another,
kernel stores the pages
in the same physical location.

Writes to the pages are intercepted.
Kernel creates two copies,
then allows the write
to change one copy.

Advantage: For wasted allocations,
copy-on-write saves time,
because the copy never happens.

Overcommitment

UNIX kernel keeps track of
physical locations used,
but many UNIX kernels
fail to keep track of
number of pages allocated.

e.g. When process calls
`malloc(1048576)`,
creating 256 empty pages,
kernel doesn't just skip making
256 copies of an empty page;
it fails to set aside
256 pages of memory.

What if memory isn't available?

`malloc()` succeeds anyway.
Kernel doesn't notice
the lack of memory.

Process then writes to page.
Kernel tries to copy page.
Oops, there's no memory!

Process can't continue.
Kernel kills it.
(Some kernels look around for
big new processes to kill.)

Bottom line: Process may be
killed at any moment.
No way for program to react.

For comparison:
If kernel keeps track of
number of pages allocated,
then `malloc()` returns 0.
Program checks for that
and takes appropriate action.

But this takes extra effort
for kernel implementors.

Security impact

Attacker uses up all memory.
Is this a security problem?

Charge users for memory.

Make sure that all programs
are prepared for sudden death.
Always use `rename()`
for rewriting files.
Always have explicit
"I'm done" message
when one process is
sending data to another.