Reuters, 2004.11.09:

“Worm breaks speed record from discovery to life

“A new computer worm emerged on Tuesday which broke the speed record from the announcement of a security vulnerability in Microsoft’s Internet Explorer to a full-blown virus that spreads in the wild.

“The vulnerability was discovered and made public by two hackers with aliases ‘ned’ and ‘SkyLined’ on Friday, and only four days later a worm exploiting the weakness was developed and set loose, several virus-trackers reported.

“Microsoft said the worm is a variant of MyDoom and that it was investigating the threat the worm poses.

“Some anti-virus companies said the new worm was different from MyDoom because it spreads via weblinks and not e-mail attachments.

“ ‘People will receive an e-mail saying that their PayPal account has been credited or that they are invited to watch a webcam. When they click on the link, just by viewing a site it executes code and infects the computer,’ said technical consultant Graham Cluley at Sophos Anti-Virus.

“Microsoft was expected to issue its monthly batch of security patches later on Tuesday, but the company could not immediately say if a patch for the new worm would be part of it.

“However, the U.S. software giant said that consumers who had installed Service Pack 2 for Windows XP were at a reduced risk.

“The weakness in Internet Explorer is known as the IFRAME buffer overflow vulnerability.”

[AUS-CERT admits that future exploits may work under SP2.]

2004.11.15: Guest lecture
by Jon Solworth, Director,
Kernel Security and Networking Lab,
CS.

2004.11.17: Midterm 2,
focusing on setuid and related topics.

Assignment due 2004.11.22: read
textbook Chapter 4.

Attacker blocking permission bits

Each process has, in system data, **umask** ("file mode mask").

Typical umask: `022`.
Another typical umask: `077`.

Any permission bit in umask is removed from new files.

e.g. `open("foo",O_CREAT,0666)` creates `foo` with permissions `0644` if umask is `022`; or `0600` if umask is `077`.

Umask is preserved by `execve`.

Joe can run a setuid program
with umask set to 0777.

Files created by program
then have permissions 000:
not readable, not writable,
even to the file owner.

`root` can read and write anyway,
but maybe program is setuid
to something other than `root`.

Fix: Program sets its own umask.

System-specific setuid problems

OS designer adds system data
and neglects to consider effect
of data after setuid exec.
(Even worse: considers effect,
and blames the setuid programs.)

e.g. FreeBSD allows two processes
to share their signal actions.

FreeBSD bug fixed 2001.07.09:
shared signal actions weren't
un-shared by `execve`.
Any user can take over
any setuid program.

Another Sendmail example

Bug sort-of-fixed 1996.09.17:

```
a->q_uid = daemon_uid;
a->q_gid = daemon_gid;
pw = getpwnam(user);
if (pw != NULL) {
  a->q_uid = pw->pw_uid;
  a->q_gid = pw->pw_gid;
}
```

`getpwnam()` looks for a uid and gid in `/etc/passwd`.

e.g. `getpwnam("djb")` returns uid and gid 1001 if `/etc/passwd` has `djb:*:1001:1001:...`

Context: Sendmail delivers messages to accounts such as `djb`. `/home/djb/.forward` can specify a program to run for each message; Sendmail runs that program under `djb`'s uid.

To figure out `djb`'s uid, Sendmail calls `getpwnam("djb")`, which reads `/etc/passwd` and returns 1001.

Sendmail calls `setuid(1001)`.

Sendmail also delivers messages to aliases such as `postmaster`. `/etc/aliases` can specify a program to run for each message; Sendmail runs that program under uid 1 (`daemon`).

Sendmail calls `getpwnam("postmaster")`, which doesn't find `postmaster` in `/etc/passwd`; returns 0.

Sendmail sees the 0 and calls `setuid(1)`.

Joe runs Sendmail, telling it
to deliver a message to `joe`.

Sendmail looks in `/home/joe/.forward`,
which says “Run `/home/joe/evil`.”

Oops, system is very busy.
Sendmail saves message in queue,
along with the following note:
“Deliver message to `joe`
by running `/home/joe/evil`.”

Joe starts Sendmail again,
telling it to run the queue:

```
    joe% sendmail -q
```

System is no longer busy.
Sendmail tries to deliver message
by running `/home/joe/evil`.
But what uid should it use?

Sendmail calls `getpwnam("joe")`
to find the uid and gid.

By setting resource limits,
Joe can make `getpwnam()` fail.
Easiest: file-descriptor limits.

`getpwnam()` returns 0,
even though `joe` is in `/etc/passwd`.

Sendmail runs `/home/joe/evil`
as uid 1.

Joe can now read and destroy
subsequent mailing-list deliveries.

Sendmail “fix”:
Remove file-descriptor limits.

But Joe can still force
`getpwnam()` to fail.

System has limit on
total number of open files
across all processes.
If Joe opens many files,
`getpwnam()` can’t open more.

Joe can attack any program,
not just setuid programs,
in this way.

Underlying source of problem:
`getpwnam()` returns 0
for “permanent” errors
(user not in `/etc/passwd`)
and for “temporary” errors
(unable to open `/etc/passwd`).

For temporary errors,
Sendmail needs to try again later;
but Sendmail can’t tell
whether the error was temporary.

For comparison: If `open()`
fails because file doesn’t exist,
it sets `errno` to `ENOENT`.
If it fails because of fd rlimit,
it sets `errno` to `EMFILE`.
`getpwnam()` should use `ESRCH`.