Ryan Naraine, Eweek, 2004.11.11:

“XP SP2 flaw warning sparks debate on disclosure

“The debate over responsible disclosure of security flaw warnings has erupted again, with Microsoft chiding a private research firm for releasing information on 10 new flaws found in the Windows XP SP2 (Service Pack 2) operating system.

“San Jose, Calif.-based Finjan Software released an alert warning that attackers could ‘silently and remotely’ hijack SP2 machines because of ‘major flaws’ that compromise end-user security.

“Finjan chief executive Shlomo Touboul told eWEEK.com that full technical details of the vulnerabilities including proof-of-concept code were given to Microsoft, but the software giant reacted sharply by suggesting that the Finjan warning is overblown.

“ ‘Our early analysis indicates that Finjan’s claims are potentially misleading and possibly erroneous regarding the breadth and severity of the alleged vulnerabilities in Windows XP SP2,’ a Microsoft spokesperson said.

“ ‘Once Microsoft concludes investigating Finjan’s claims and if Microsoft finds any valid vulnerability in Windows XP SP2, it will take immediate and appropriate action to help protect customers,' she added.

“According to Finjan, the flaws are so serious that XP SP2 users are at risk if they simply browse a Web page. The holes also could be exploited to allow malicious hackers to remotely access users' local files or to switch between Internet Explorer Security Zones to obtain rights of local zone.

“The research outfit also claims that it discovered a bug in the notification mechanism built into XP SP2 to warn users when executable files are being downloaded. Finjan claims it has already proven to Microsoft that hackers can bypass the mechanism to inject arbitrary code without any warning or notification.

"When told that Microsoft was discounting the severity of his company's claims, Finjan's Touboul lashed back: 'These are not theoretical assumptions. These findings are based on code implementing each and every one of those 10 vulnerabilities.'

"Microsoft said it would continue investigating Finjan's claims to confirm valid vulnerability claims before rolling out possible fixes.

" '[We encourage] Finjan to abide by the principles of responsible disclosure and to decline to provide further comment or details on the alleged vulnerabilities until Microsoft is able to complete its investigation and can respond properly to protect customers,’ the spokesperson said.”

Next week: Guest lectures and midterm.
Check web page for latest schedules.

Assignment due 2004.11.22: read textbook Chapter 4.

## Handling open() failures

```
#include <errno.h>
#include <fcntl.h>
fd = open(...);
if (fd == -1)
  if (errno == ENOENT)
    file does not exist;
  else
    not sure, try again later;
else
  file exists, use fd;
```

Should also check for a few variations of `ENOENT`: `ENOTDIR`, for slash after non-directory; `ENAMETOOLONG`.

The following code is wrong:

```
   fd = open(...);
   if (fd == -1)
     file does not exist;
   else
     file exists, use fd;
```

Many reasons that `open()` can fail even if the file exists:
`EACCES`, permission denied;
`ENFILE`, full system file table;
`EINTR`, interrupted by a signal;
`EIO`, disk failure; etc.

Sometimes programmer is sure that
permission won't be denied,
signals won't be received,
system file table is big enough
for all normal use, etc.

But attacks are not normal use!
Usually attacker can fill up
system file table.

Wrong code concludes, incorrectly,
that file does not exist.
Attacker has forced code to misbehave.

Right code tries again later.
Attacker has merely delayed service.

## Is delayed service a problem?

Often not. Charge users
for their resource use.
Extra use means extra charge.

Another user can avoid delay by
paying for independent resources.
UNIX has very few facilities
to allocate independent resources;
so run an independent machine.

Can still have security problems:
(1) attacker avoiding the charges
for the resources he used;
(2) attacker doing something worse
to another user than delaying service.

## Handling `getpwnam()` failures

```
#include <sys/types.h>
#include <pwd.h>
pw = getpwnam(...);
if (!pw)
  if (errno == ESRCH)
    account does not exist;
  else
    not sure, try again later;
else
  account exists, use pw;
```

Oops—`getpwnam()` manual
does not promise this use of `errno`!

Hard to write `getpwnam()` replacement:
details depend somewhat on system.

## Handling `fork()` failures

```
#include <sys/types.h>
#include <unistd.h>
pid = fork();
if (pid <= 0)
  if (pid == 0)
    running in new process;
  else
    failed, try again later;
else
  running in old process;
```

Attacker can consume all memory,
making `fork()` fail;
important to check.

## Handling `read()` failures

```
#include <sys/types.h>
#include <unistd.h>
r = read(...);
if (r <= 0)
  if (r == 0)
    end of file;
  else
    try reading again later;
else
  have read r bytes, use them;
```

Can attacker force `read()` from a disk file to fail?

With network filesystems, yes: attacker can flood the network.

Reading from another process:
if the other process dies,
`read()` returns 0.
(Should be `-1`. Too late to fix UNIX.)

On many systems, attacker can force
another user's programs to die:
attacker allocates more memory
than the computer actually has,
and then writes to that memory.

"Overcommitment":
unfortunate kernel feature
falsely reporting allocation success
and then killing other programs.

To avoid confusion, sender must
send explicit "I'm done" message.