Using `systrace` under OpenBSD

Alexander von Gernler

<grunk@steelix.kd85.com>

FOSDEM Brussels

February 21-22, 2004

About this talk

This talk will . . .

- discuss the idea of system call interposition/interception
- introduce the use of `systrace` under OpenBSD
- present you with links to the topic

The author . . .

- participates in the [de] OpenBSD Translation Project
- runs the OpenBSD mirror at the University of Erlangen, Germany (`anoncvs2.de.openbsd.org`)
- does not want to sell `systrace` for something brand new, but he thinks that still too few people know about it

Motivation

The problem

- important daemons running on your system need root, e. g. for binding to ports <1024, writing raw packets, . . .
- yes, of course you can `chroot` them. You actually should.
- whenever possible, you should also make use of *privilege separated* daemons
- upon exploitation, causing damage is still possible, at least in the chroot environment (if at all set up :)

So what exactly is „causing damage“?

- misbehavior of daemons, especially not doing what they are usually supposed to (spawning root shells and what not)
- under UN*X, the important operations are done using syscalls

System call interposition/interception

Looking at the symptoms

- `httpd` needs to read files from disk, bind to port 80 and does much more other stuff
- but it does *not* need to spawn shells
- then again, why should it be allowed to do so?

That's the idea!

- interpose between system calls and keep track of what is supposed to be a *normal behavior* of your daemon.
- do this on a non-compromised vanilla installation within a test scenario
- after knowing what your daemon needs, forbid everything else

This can be done using `systrace`!

Controlling syscalls using `systrace`

- has nothing to do with chroot, and can be used independently
- explicitly allows or forbids the execution of certain syscalls to specific programs
- *privilege elevation* also possible: users' processes can do single syscalls as root (comes in handy for programs with suid bits which would drop root after doing the privileged work)
- decisions base on policy files, either global in `/etc/systrace`, or local in `${HOME}/.systrace`
- matching can be done via textual equality, globbing, regexes, substrings. Possible targets to match on include sockaddr, filename, gid, and other parameters of the syscall
- profile creation by hand, by examples from the net or through profiling of vanilla programs

Practical examples using `systrace`

- Profile generation (interactively with GUI):

```
root@vario:~% systrace /usr/libexec/ftpd
```

- Profile generation (non-interactively):

```
root@vario:~% systrace -A /usr/libexec/ftpd
root@vario:~% ps ax | grep ftpd
24421 ?? Ixs    0:00.00 /usr/libexec/ftpd
12929 ?? Is     0:00.01 systrace -A /usr/libexec/ftpd
root@vario:~%
```

  The 'x' flag in the `ps` output indicates that `ftpd` is monitored by `systrace`

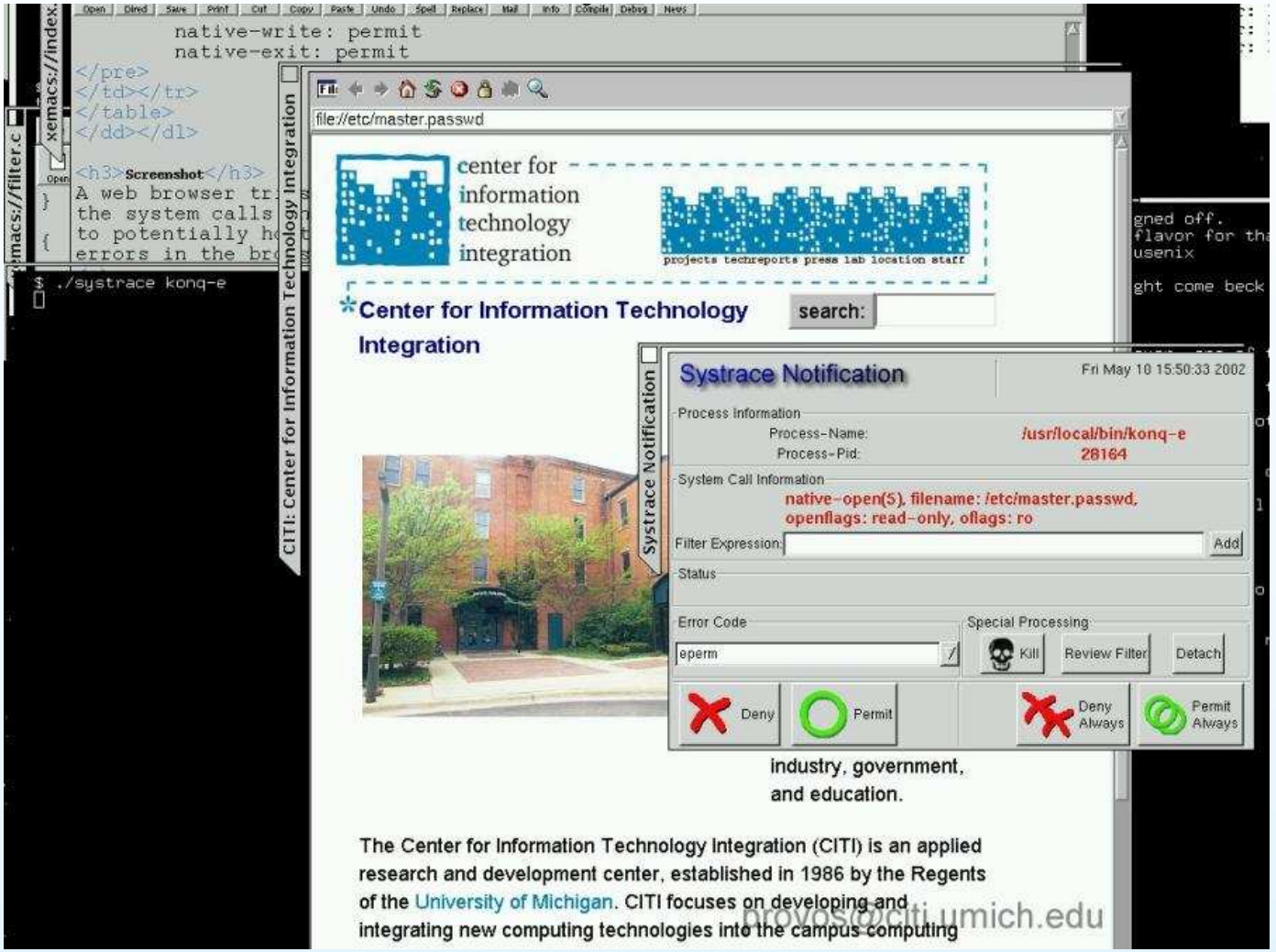
- Profile enforcement:

```
root@vario:~% systrace -a /usr/libexec/ftpd
```

- Interactively changing profiles by attaching to a running `systrace`:

```
root@vario:~% systrace -p 12929 /usr/libexec/ftpd
```

Nifty screenshot, taken from `www.systrace.org`



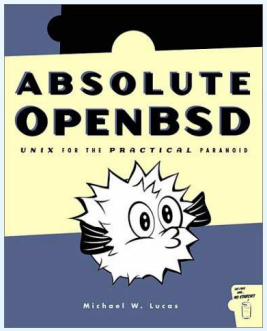
Policy example ${HOME}/.systrace/bin_ls

```
Policy: /bin/ls, Emulation: native
        native-munmap: permit
[...]
        native-stat: permit
        native-fsread: filename match "/usr/*" then permit
        native-fsread: filename eq "/tmp" then permit
        native-fsread: filename eq "/etc" then deny[enotdir]
        native-fchdir: permit
        native-fstat: permit
        native-fcntl: permit
[...]
        native-close: permit
        native-write: permit
        native-exit: permit
```

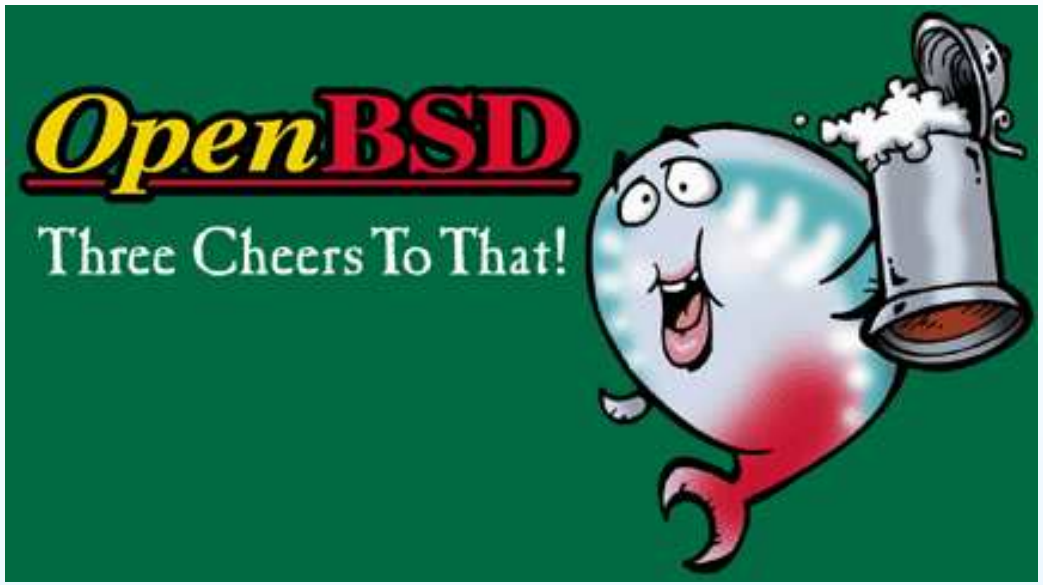
Links to the topic

`systrace` was created by NIELS PROVOS `<provos@citi.umich.edu>`

- The `systrace` Homepage:
  `http://www.systrace.org`
  You can also find still more related links there
- The Hairy Eyeball Project collects sample policies:
  `http://www.blafasel.org/~floh/he/`
- The Book *Absolute OpenBSD* by Michael Lucas contains a nice little section about `systrace`



More questions?



- Slides created using LaTeX, `prosper`, `make` and CVS under OpenBSD 3.4 / i386
- Request the slide source code from `<grunk@steelix.kd85.com>`
- Mails: 72 characters per Line, no HTML mails