Capsicum: Practical capabilities for UNIX

Anna Kornfeld Simpson &
Venkatesh Srinivas

Capsicum: Practical capabilities for UNIX

And a little bit about Confused Deputies too...

Anna Kornfeld Simpson & Venkatesh Srinivas

A Tale of Woe

Evil Solitaire

A Tale of Woe, pt.2

“I was just trying to share a photo!”

Problem, pt. 1

UNIX (& friends) treat the user as a principal

Any process of a user can do anything that user can do.

Confused Deputy

Tricking a compiler

The Problem

Abelson: “Name gives you Power over object”

In UNIX & friends, a process can pronounce names that it might not have power over.

The names convey no authority or reference

The Problem

The amount of Ambient Authority is ….

(Orthogonal problem)

- We don’t trust other users on our machine
- Existing access control systems specify which other users can do what to each file (using techniques such as chmod() in Linux or ACLs in Windows, or sharing online)
- If the user has permission to access the file, any program run by that user can access

What are capabilities?

An unforgeable reference.

The reference itself conveys authority.

What are capabilities?

“Protection”, Lampson 1971

Access Control Matrix

	Object 1	Object 2
P1	R,W,X	-
P2	R	R

What are capabilities?

An elegant idea from a more civilized time

What are capabilities?

An elegant idea from a more civilized time

But actually…

“Programming Semantics for Multiprogrammed Computation”, Dennis, van Horn 1966.

GNOSIS ‘Great New Operating System In the Sky’ (1979)

KeyKOS (1985)

EROS (1999)

E language (2006)

Cap’n Proto / Sandstorm (coming to an Internet near you any day now!)

… many, many others.

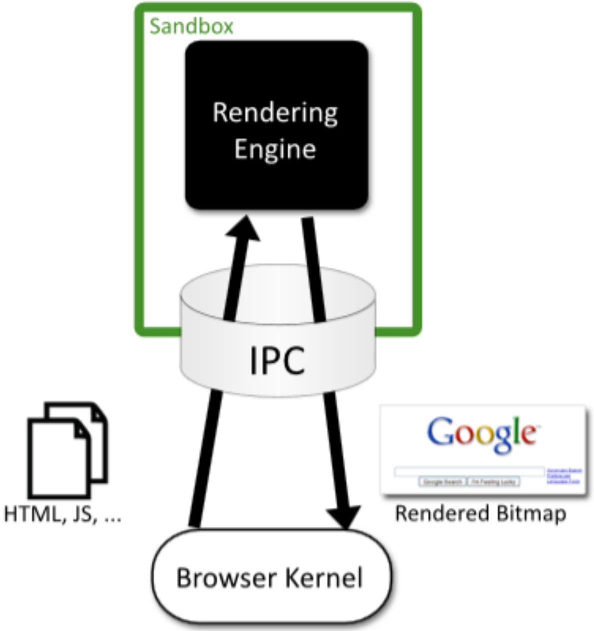
A Pure Capability System

No ambient authority

All object access for a process via explicit caps a process holds.

And now we change gears...

Sandboxing Programs



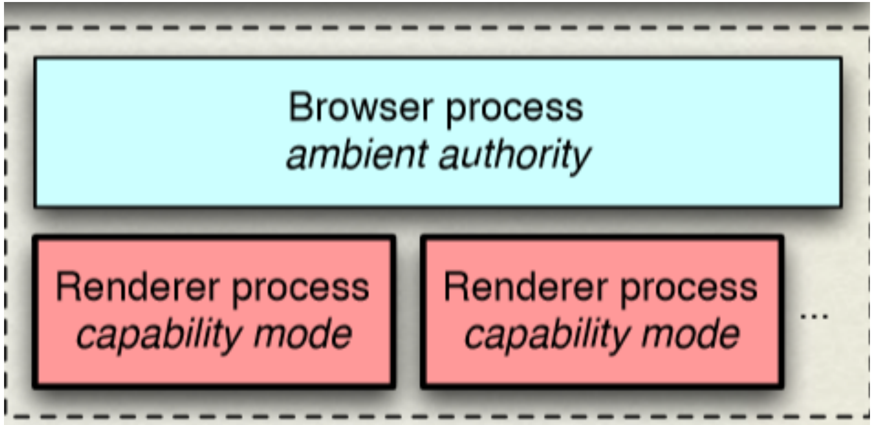
Modern applications are large and should have their pieces sandboxed.

Ex. the rendering engine in the browser is the location of most vulnerabilities.

From: Barth et. al. Security Architecture of Chromium Browser, 2008

Sandboxing Programs

- Sandboxed processes shouldn't have ability to see filesystem etc.
- Capabilities allow us to do sandboxing very neatly!



From: Cambridge slide deck

Delegation

- Want to be able to pass capabilities to other programs (example: photo gallery shares image with Dropbox)
- Share only the specific capabilities
- Programs delegate specific capabilities by sharing the file descriptors

Imagine

What if …?

Capsicum

New UNIX interface (developed for FreeBSD), ~2010.

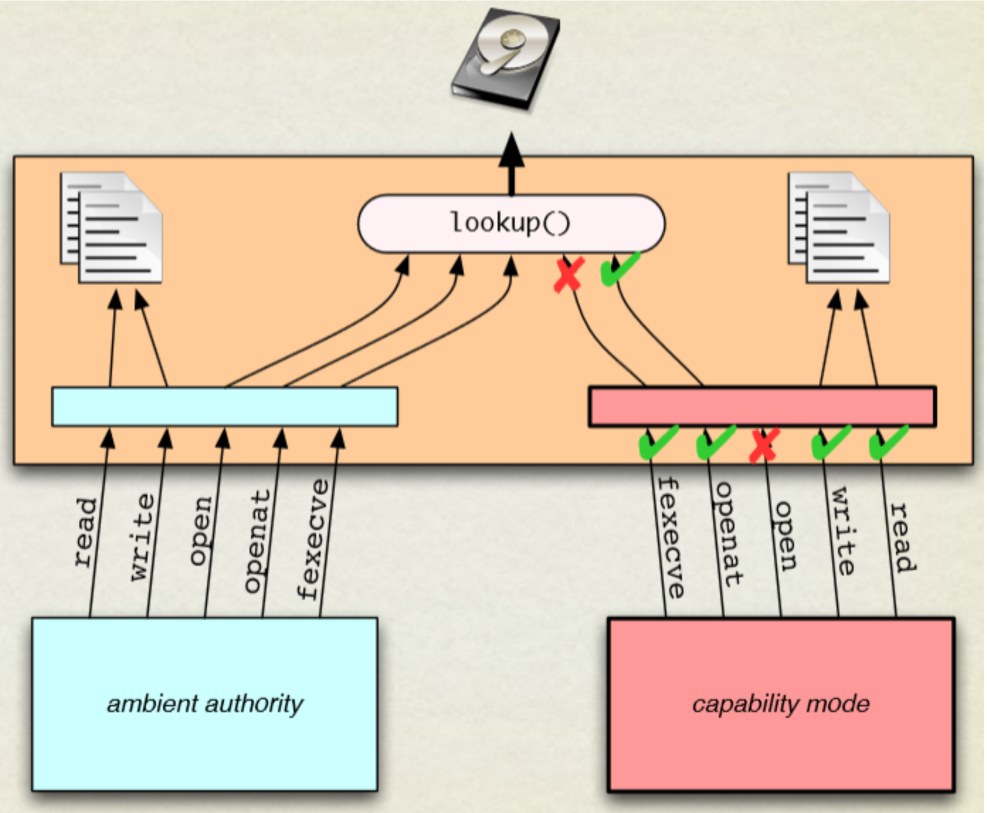
Merged for FreeBSD-9.

Capsicum

Modal interface -- a process may enter ‘cap’ mode

Gives up old interfaces when it does so.

Capsicum


lookup()
read
write
open
openat
fexecve
ambient authority
fexecve
openat
open
write
read
capability mode

cap_enter

Enter capability mode.

Irrevocable for a process *and its descendants*

Capability mode

No ambient authority.

No global filesystem namespace. (sys_open no more)

All file descriptors preserved on entry.

cap_new

Generates a ‘weakened’ copy of a file descriptor

Ex: you can hold a read/write fd to a file.

And generate a weakened one to send to some other process

PIDs as file descriptors?!

fork() / wait() / kill() rely on a global namespace (PIDs)

fork() is disallowed in capability mode.

pdfork() replaces it -- same sematics, but returns a file descriptor.

fexecve

execve() doesn’t work in capability mode (relies on a string path)

fexecve() should be easy, right?

Surprise global namespace! (RTLD)

Sandboxing is easy now!

- “immediate benefits” in tcpdump: 10 lines
    - Acquire resources up front, packet-process in capability mode
    - On-demand initialization
- gzip is surprisingly complicated
    - Need usermode library for pipeline mode
    - 400 lines, mostly in RPCs

Performance is fine

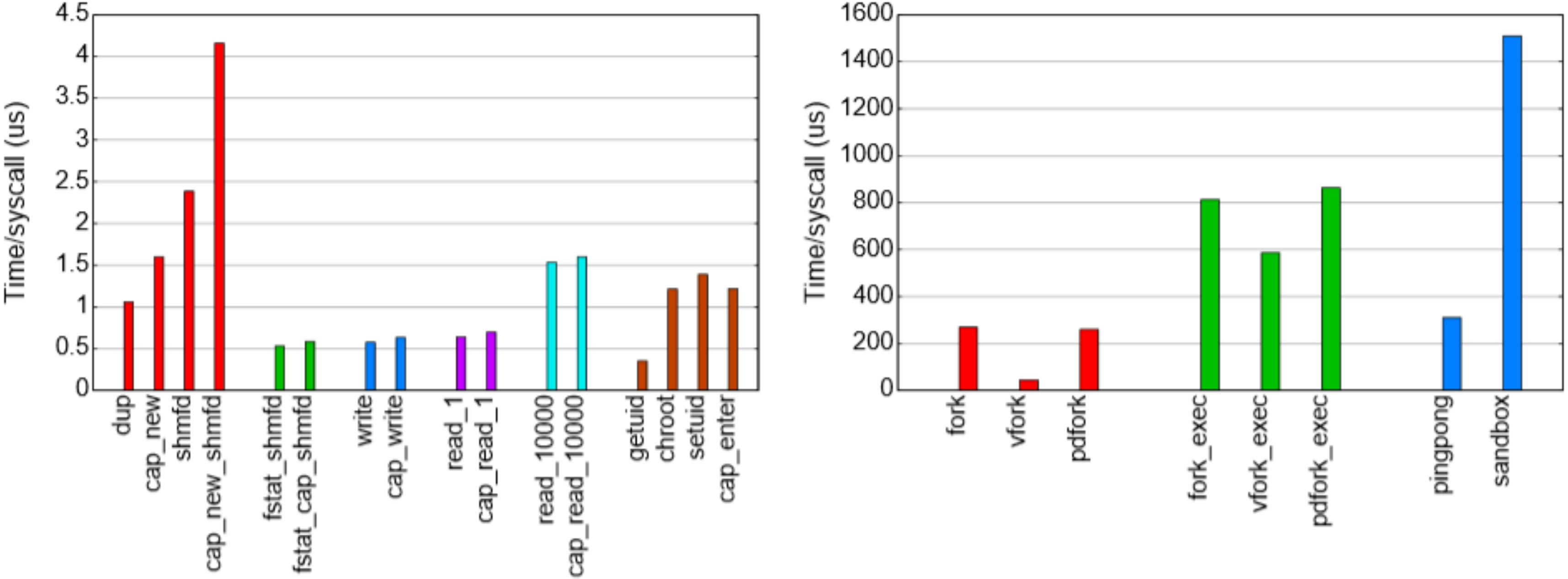
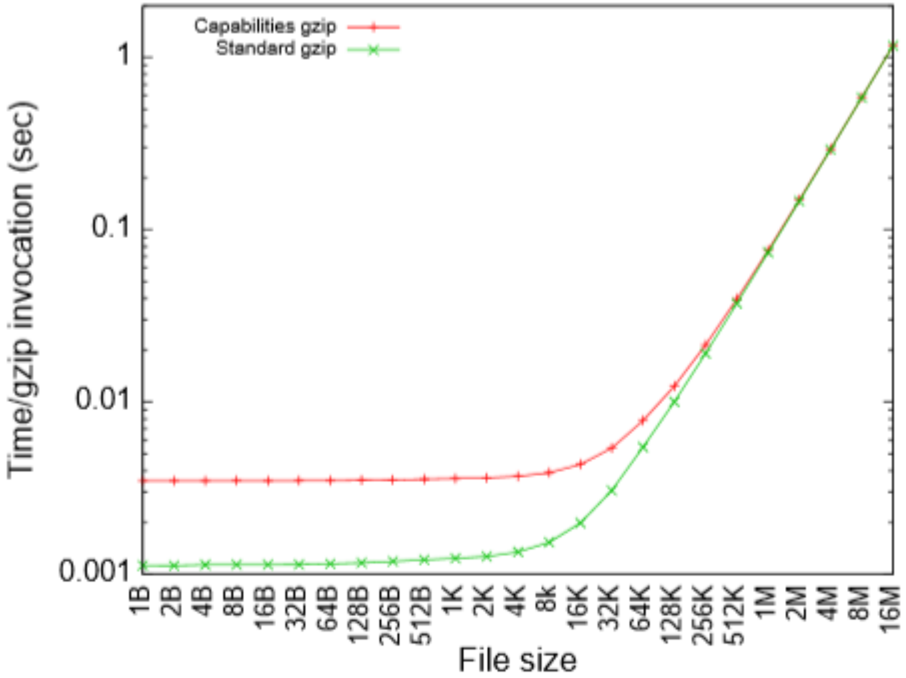


Figure 13: Capsicum system call performance compared to standard UNIX calls.

Performance is fine



Now available in a store near you!

Shipping since FreeBSD 9

Ported to DragonFly BSD

Discussion

- Does this go far enough in sandboxing?
    - kernel/user implementation question?
- Is this worth all the re-writing that is necessary?
- How would you sandbox Evil Solitaire or Dropbox without capabilities?