The Capsicum Security Framework: Sandboxing Done the Right Way

Ilya Bakulin
kibab@FreeBSD.org

University of Applied Sciences,
Vienna, Austria
May 5, 2012

Outline

MS-DOS, Windows 9x

- Single-user OSes
- 1 computer == 1 user
- Internet? No, I don't have this

No protection of user data at all

Modern UNIX-like systems, Windows NT

- Multi-user systems, protect user's data from other users

Modern UNIX-like systems, Windows NT

- Multi-user systems, protect user's data from other users
- But don't protect user's data from applications that this user runs

You're paranoid. Why should I care?

- Modern applications are very complex, and sometimes just poor written

You're paranoid. Why should I care?

- Modern applications are very complex, and sometimes just poor written
  ... Firefox... :-)

You're paranoid. Why should I care?

- Modern applications are very complex, and sometimes just poor written
  ... Firefox... :-)
- Even well written applications may depend on libraries that aren't

You're paranoid. Why should I care?

- Modern applications are very complex, and sometimes just poor written
  ... Firefox... :-)
- Even well written applications may depend on libraries that aren't
  ...libpng...

You're paranoid. Why should I care?

- Modern applications are very complex, and sometimes just poor written
  ... Firefox... :-)
- Even well written applications may depend on libraries that aren't
  ...libpng...
- Once application is pwn'ed, the attacker has access to ALL user data and possibly sensitive information. Sad, but true.

Okay, let's chroot our process, and that's it!

- chrooting doesn't prevent process from opening new network sockets
- It's still possible to send signals to processes outside chroot
- `chroot(2)` requires root rights

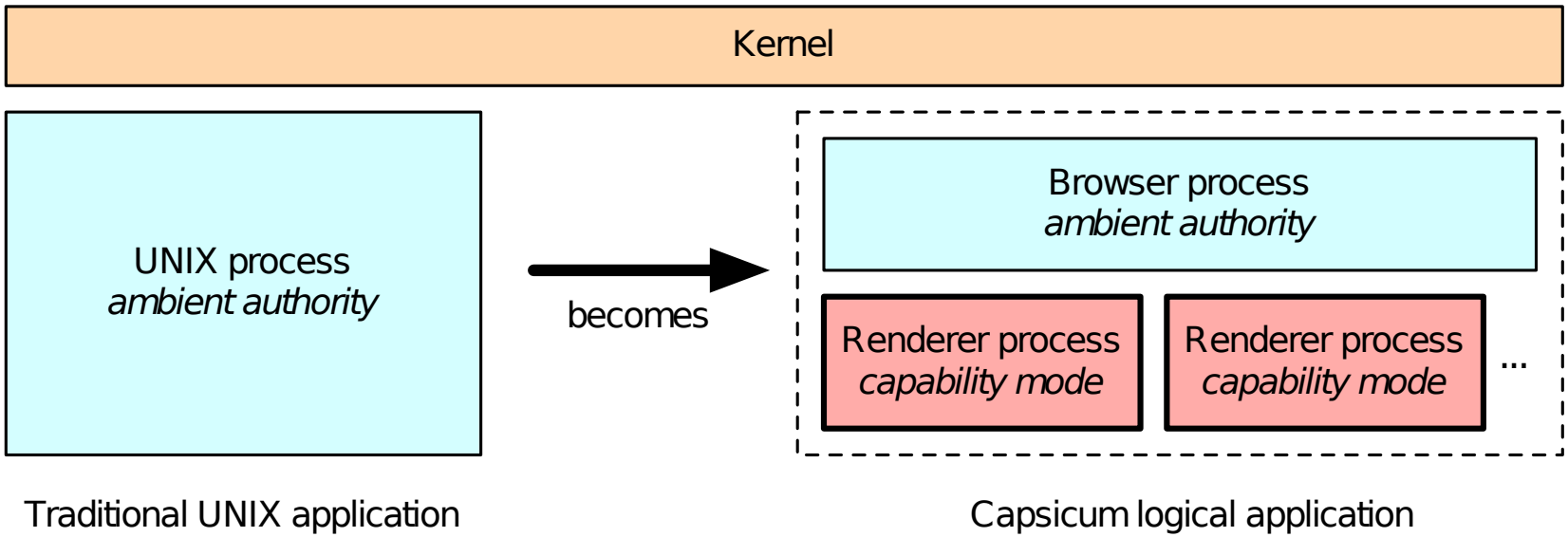
So what is Capsicum?

- Lightweight operating system capability and sandbox framework
- Included in FreeBSD 9.0, being ported to OpenBSD and Linux
- New kernel primitives (sandboxed capability mode and capabilities) and a userspace sandbox API
- Access restrictions are requested by application and enforced by OS kernel
- Requires modifications of application source code
- Extends, rather than replaces, traditional POSIX objects like file descriptors and network sockets

The Idea

- One big application may be split in several smaller ones
- Different parts are restricted differently and communicate with each other via IPC
- Principle of least privilege – applications should be able to access only those resources that are nessesary for their normal operation.

Guess what browser is it? :-)



Some examples

- Easy modifications (a few lines of code): `bspatch(1)`, `bsdiff(1)`, `tcpdump(1)`
- Somewhat more complicated: `fetch(1)`, `bzip2(1)`
- Complex: `syslogd(8)`

bspatch(1)

Possibly insecure code that performs bzip2 decompression and patching

- Only moved opening all files to the beginning of main(), then called `cap_enter()` and limited access rights on already opened FD's
- Prepare: `https://github.com/kibab/capsicum/commit/a73971bdbeeb2af3c90d98411e5ecdee2acedc57`
- Implement: `https://github.com/kibab/capsicum/commit/b03dad076d53900be85a9bc780612e35d4da3e5a`

fetch(1)

Possibly insecure code handling compressed SSL streams

- Fork after opening network socket and/or local file(s). Parent controls terminal interaction, child downloads a file and writes it to disk
- Had to add simple message-parsing, because `fetch(1)` wants to display download progress
- Slightly modified `libfetch` to always supply information about file descriptors

Chromium

Possibly insecure code that performs HTML parsing, renders images and multimedia. Multiple libraries that can also be buggy.

- Excellent target to add Capsicum sandboxing because it already employs several sandboxing technologies available on Mac OS X, Linux and Windows.

Chromium sandboxing

OS	Model	LoC	Description
Windows	ACLs	22,350	Windows ACLs and SIDs
Linux	chroot	605	setuid root helper sandboxes renderer
MacOSX	Seatbelt	560	Path-based MAC sandbox
Linux	SELinux	200	Restricted type enforcement domain
Linux	seccomp	11,301	seccomp + userspace syscall wrapper
FreeBSD	Capsicum	100	Capsicum sandbox using `cap_enter`

Effectiveness of different sandboxes

- DAC/MAC-based systems, as well as SELinux, separate enforcement policy from code
- chroot requires setuid + doesn't protect network and other processes
- seccomp is just very hard to use properly

The cost of security

- Applications designed in secure way already accept certain performance drop
- Those that not – depends on nature of the application
- In general, performance cost of Capsicum is almost 0. Some nice features like `openat(2)` may help to reduce amount of nessesary IPC.

Current status

- FreeBSD 9.0: not turned on in GENERIC kernel
- FreeBSD 9.1 (upcoming): features present in GENERIC, support in some applications
- OpenBSD: development suspended, but may be continued in the meantime
- NetBSD: unknown...
- Linux/ChromeOS port in progress

Hey, I want to try it!

No problems!

- Start by reading articles on Capsicum Project page
- Subscribe to the Capsicum mailing list
- Follow developers on GitHub
- Install FreeBSD 9 with Capsicum-enabled kernel

Hey, I want to try it!

No problems!

- Start by reading articles on Capsicum Project page
- Subscribe to the Capsicum mailing list
- Follow developers on GitHub
- Install FreeBSD 9 with Capsicum-enabled kernel
- ... And may the Force be with you.

Useful links

- Capsicum project website: `http://www.cl.cam.ac.uk/research/security/capsicum`
- Capsicum mailing list: cl-capsicum-discuss@lists.cam.ac.uk
- GitHub: projects of the following users: trombonehero, benlaurie, kibab

Thank you for your attention!

flood me questions :-)

Thank you for your attention!

flood me questions :-) Happy hacking!