sandboxing

last time (1)

Rust general syntax:

- `let name : type`
- references declared with `& Type`, created with `& value`, deref'd with *ref
- `let mut name` for mutuable variable
- `& mut X` for mutable reference to X
- `fn name(arg: argtype) -> rettype { code }`

ownership/borrowing rule Rust enforces

- version 1: one owner and only they can access, can move to new owner
- version 2: one owner, can move to new owner or be temporarily borrowed by one ref
- version 3: can be borrowed read-only by multiple refs

last time (2)

escape hatches in rust:

- code explicitly marked `unsafe`
- implement other rules: example: reference counting

concurrency

- problem: use-after-free bugs caused by concurrency issues
- Rust solution: explicitly mark types as multicore/thread safe

logistics note

data races

Rusts rules around modification built assuming concurrency

OSes and other "systems programming" applications use multiple cores/threads

particular problem: value being used from multiple threads at same time

data races from use-after-free

given x: Rc<Foo> variable calling x.clone() on two cores

- some variable shared between two cores
- reference counting will prevent use-after-free, right?

```
x.clone on core A             x.clone on core B
-------------------------------------------
x.inc_strong():
  temp <- self.count
                              x.inc_strong():
                                temp <- self.count
                                self.count <- temp + 1
  self.count <- temp + 1
```

problem: reference count one too low!

Rust solution?

one option: require Rc implementation to handle mutiple cores

- problem: not zero overhead

Rust solution: different types for multithreaded/multicore code

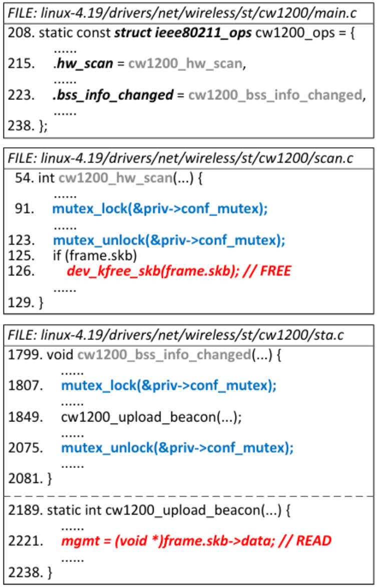
two “traits” to mark custom types:

- Sync: can be used from multiple cores/threads at once
- Send: can be moves from one thread to another

two implementations of referenc counting

- Rc: not suitable for multicore, not marked Sync/Send
- Arc: is suitable for multicore, slower than Rc probably

example: concurreny UAF bug



```
FILE: linux-4.19/drivers/net/wireless/st/cw1200/main.c
208. static const struct ieee80211_ops cw1200_ops = {
         ......
215.     .hw_scan = cw1200_hw_scan,
         ......
223.     .bss_info_changed = cw1200_bss_info_changed,
         ......
238. };
```

```
FILE: linux-4.19/drivers/net/wireless/st/cw1200/scan.c
 54. int cw1200_hw_scan(...) {
         ......
 91.     mutex_lock(&priv->conf_mutex);
         ......
123.     mutex_unlock(&priv->conf_mutex);
125.     if (frame.skb)
126.         dev_kfree_skb(frame.skb); // FREE
         ......
129. }
```

```
FILE: linux-4.19/drivers/net/wireless/st/cw1200/sta.c
1799. void cw1200_bss_info_changed(...) {
          ......
1807.     mutex_lock(&priv->conf_mutex);
          ......
1849.     cw1200_upload_beacon(...);
          ......
2075.     mutex_unlock(&priv->conf_mutex);
          ......
2081. }
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
2189. static int cw1200_upload_beacon(...) {
          ......
2221.     mgmt = (void *)frame.skb->data; // READ
          ......
2238. }
```

Figure 2: A reported bug in the *cw1200* driver in Linux 4.19

Figure from Bai, Lawall, Chen and Mu (Usenix ATC'19)

"Effective Static Analysis of Concurrency Use-After-Free Bugs in Linux drivers"

bug in a wireless networking driver

other policies Rust supports

RefCell — borrowing, but check at runtime, not compile-time

- detect at runtime if used while already used
- internally: destructor call when returned object goes out of scope

Weak — reference-counting, but don't contribute to count

- detect at runtime if used with count $= 0$

Mutex — with multicore, enforce one user at a time by waiting

…

other policies Rust supports

RefCell — borrowing, but check at runtime, not compile-time

- detect at runtime if used while already used
- internally: destructor call when returned object goes out of scope

Weak — reference-counting, but don't contribute to count

- detect at runtime if used with count $= 0$

Mutex — with multicore, enforce one user at a time by waiting

...

exercise: which smart pointer?

Rc, Arc (reference counting, w/ or w/o threading support

RefCell (borrowing, check at runtime)

Weak (reference counting, but don't contribute to count — works with Rc)

Mutex (with multicore, one-at-a-time by waiting)

say I have flight reservation system with Flight objects that have references to Ticket objects and vice-versa,
and Customer objects that have references to Ticket objects and vice-versa?

zero-overhead

normal case — lifetimes — have no overhead

compiler proves safety, generates code with no bookkeeping

other policies (e.g. reference counting) do

...but can implement new ones if not good enough

other things languages can enforce?

saw: enforcing no use-after-free

lots of coding conventions we might try to enforce:

code's runtime does not depend on secret data

- secret data has different type
- variable time operations prohibited with secret data

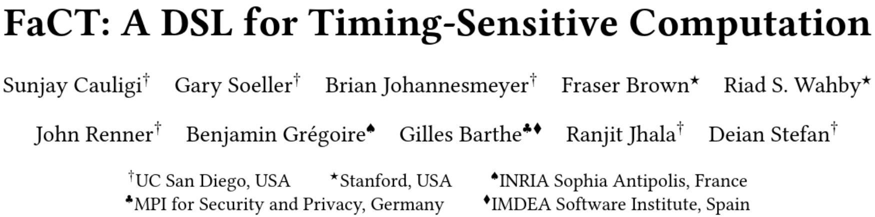
sensitive data not passed to wrong place

- sensitive data has different type
- assignment to wrong places is a type error

code has bounded runtime

- langauge prohibits not unbounded loops, recursion, etc.

some constant time ideas

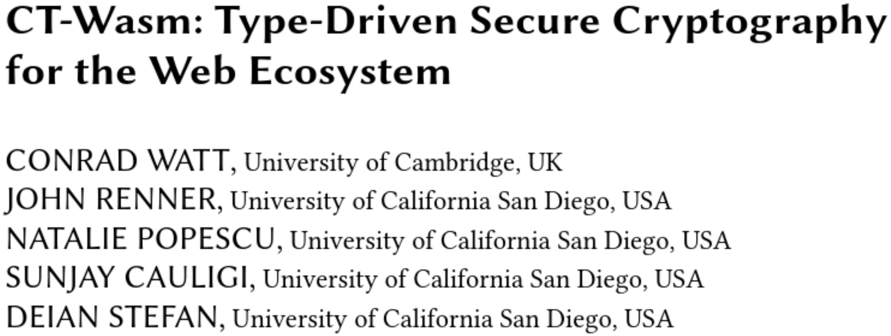


FaCT: A DSL for Timing-Sensitive Computation

Sunjay Cauligi[†] Gary Soeller[†] Brian Johannesmeyer[†] Fraser Brown[★] Riad S. Wahby[★]

John Renner[†] Benjamin Grégoire[♠] Gilles Barthe[♣♦] Ranjit Jhala[†] Deian Stefan[†]

[†]UC San Diego, USA [★]Stanford, USA [♠]INRIA Sophia Antipolis, France
[♣]MPI for Security and Privacy, Germany [♦]IMDEA Software Institute, Spain



CT-Wasm: Type-Driven Secure Cryptography for the Web Ecosystem

CONRAD WATT, University of Cambridge, UK
JOHN RENNER, University of California San Diego, USA
NATALIE POPESCU, University of California San Diego, USA
SUNJAY CAULIGI, University of California San Diego, USA
DEIAN STEFAN, University of California San Diego, USA

FaCT, PLDI 2019; CT-Wasm: POPL 2019

constant-time programming languages

active research area, no consensus on what works best

common approach: separate type for **secret** data

compiler or language virtual machine disallows variable-time operations using secret data

no secret-based array lookup (cache timing varies)

  e.g. array[secret_value] $\rightarrow$ compile error (type mismatch)

no secret-based integer division (usually variable speed instruction)

…

explicit operations for any secret-to-non-secret conversions

least privilege

a typical program I run on my desktop is allowed to...

make network connections to anywhere

upload all my files to the Internet

delete all my files

record all my keystrokes

...

but it probably doesn't need to...

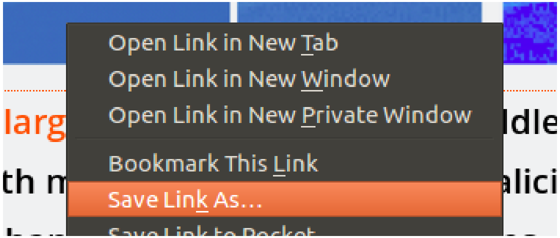
ideally: if typical program was compromised/malicious,
it still wouldn't be able to do most of these things

things applications need?

what things should browser be able to do?

what things should word processor be able to do?

things broswers need



save files

have your webmail password

...

multi-user OSs

```
cr4bd@labunix01:~$ cp myprogram.exe /bin/ls
cp: cannot create regular file ‘/bin/ls’: Permission denied
```

programs have limited privileges

permission enforcement

```
struct Process {
    int user_id;
    ...
};
int handle_open_system_call(char *filename, ...) {
    Process* currentProcess = GetCurrentProcess();
    File* file = GetFileByFilename(filename);
    if (!file->UserCanAccess(currentProcess->user_id)) {
        return ERROR_PERMISSION_DENIED;
    }
    ...
}
```

multi-user OSs

```
cr4bd@labunix01:~$ cp myprogram.exe /bin/ls
cp: cannot create regular file ‘/bin/ls’: Permission denied
```

programs have limited privileges

OS tracks “user” of running every program

result: malware I installed shouldn’t be able to effect other users

idea 1: reuse this support for web browsers

- webpage should run as “different user”
- malware should only affect web browser?

the privilege separation idea

can't make whole browser run as "different user"

- still need to save files, read password, etc.

how about just the parts that are "dangerous"?

- part that runs scripts, parses HTML

simple privilege separation

simple example: want to show videos

video decoding library is tens of thousands of lines of code

- often buggy, includes hard-to-check hand-written assembly

what does video decoding library do?

- read video file as input
- output images as output

simple privilege seperation

setup: create new user

start video decoder as new user

communicate via “pipes”

- like terminal to be used by program

simple privilege seperation

```
/* dangerous video decoder to isolate */
int main() {
    /* switch to right user */
    SetUserTo("user-without-privileges"));
    while (fread(videoData, sizeof(videoData), 1, stdin) > 0) {
        doDangerousVideoDecoding(videoData, imageData);
        fwrite(imageData, sizeof(imageData), 1, stdout);
    }
}

/* code that uses it */
    FILE *fh = RunProgramAndGetFileHandle("./video-decoder");
    for (;;) {
        fwrite(getNextVideoData(), SIZE, 1, fh);
        fread(image, sizeof(image), 1, fh);
        displayImage(image);
    }
```

issues with privilege separation (1)

“other user” can still do too much

read unprotected files

- most of them?

write temporary files?

open network connections

use all your memory

…

issues with privilege separation (2)

awkward to do

switching users requires special permissions

seperate user for each video decoder, audio decoder, web page renderer?

- users can debug processes from same user

slowdown — extra copying

program to OS interface

primary way application talks to OS: system calls

function calls that request OS do something

typically: how program can interact with rest of system

- files
- other programs
- the network
- devices
- …

controlling program behavior: control what system calls

Linux system call filtering API

privilege seperation support: system call filtering

simple API: `seccomp(SECCOMP_SET_MODE_STRICT, 0, 0)`

“The only system calls the calling thread is permitted to make are `read`, `write`, `_exit`, and `sigreturn`. Other system calls [kill the program].”

read/write only work on already open files

later: what if we want to be finer-grained?

“sandboxing”

result of filtering operations called a “sandbox”

idea: attacker can play in sandbox as much as they want

can’t do anything “harmful”

other possible implementations:

- e.g. virtual machine

Chrome architecture

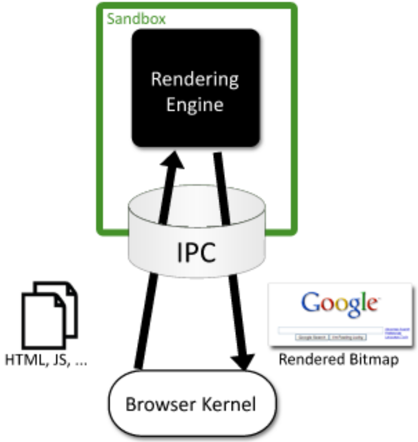


Figure 1: The browser kernel treats the rendering engine as a black box that parses web content and emits bitmaps of the rendered document.

talking to the sandbox

browser kernel sends commands to sandbox

sandbox sends commands to browser kernel

idea: commands only allow necessary things

original Chrome sandbox interface

sandbox to browser “kernel”

- show this image on screen
  - (using shared memory for speed)
- make request for this URL
- download files to local FS
- upload user requested files

browser “kernel” to sandbox

- send user input

original Chrome sandbox interface

sandbox to browser “kernel”

- show this image on screen
  - (using shared memory for speed)
- make request for this URL
- download files to local FS
- upload user requested files

browser “kernel” to sandbox

- send

needs filtering — at least no `file:` (local file) URLs

original Chrome sandbox interface

sandbox to browser "kernel"

- show this image on screen
    - (using shared memory for speed)
- make request for this URL
- download files to local FS
- upload user requested files

browser "kernel" to sandbox

- send user input

can still read any website!
still sends normal cookies!

original Chrome sandbox interface

sandbox to browser "kernel"

- show this image on screen
  - (using shared memory for speed)
- make request for this URL
- download files to local FS
- upload user requested files

browser "kernel" to sandbox

- send user input

files go to download directory only
can't choose arbitrary filenames

original Chrome sandbox interface

sandbox to browser "kernel"

- show this image on screen
  - (using shared memory for speed)
- make request for this URL
- download files to local FS
- upload user requested files

browser "kernel" to sandbox

- send user input

browser kernel displays file choser
only permits files selected by user

Site Isolation

Chrome since version 67 (desktop)/77 (Mobile) has process per site

site ≈ registered domain name (example.com, example.co.uk, etc.)

- slightly different than same origin policy

complicated to implement:

- single web page can embed content from multiple other sites
  - and those other sites can embed content from yet more sites
- web page can call services on other websites with "permission" of other website
- clicking link may or may not requiring switching to new process

same separation being prototyped in recent Firefox builds

OpenSSH privilege seperation

OpenSSH uses privilege seperation for its SSH server

what runs on the lab machines when you log into them

separate network processing code from authentication code

seperate process per connection — users don't share

OpenSSH privsep protocol

sandboxed process tells "monitor" to:

perform cryptographic operations

- long-term keys never in sandboxed process
- commands to ask for cryptographic messages they need

ask to switch to user — if given user password, etc.

- monitor process verifies login information

after authentication: new process running as logged-in user

- (normally) no issues with special privileges

privilege seperation overall

large application changes

- OpenSSH: 3k lines of code for communication/etc. added
- OpenSSH: 2% of existing code (950 of 44k lines) changed
- (but most changes simple)

lots of application knowledge

- what is a meaningful separation of 'privileged' and 'unprivileged'?

better application design anyways?

privilege separation for

let's say we wanted to add sandboxing/privilege separation to an (standalone) mail program

exercise 1: where would be concerned about security problems?

exercise 2: propose a way of dividing up the program

application confinement

confining whole browsers was hard
- we trust them to do a lot of things — e.g. write arbitrary files

but maybe we can do this for simpler applications?

idea 1: applications send system calls to OS
- limit syscalls like we limited browser kernel commands
- constructing command language "in reverse"

Linux system call filtering: detailed

Linux supports more fine-grained system call filtering

using BPF (Berkeley Packet Filter) programming language

- compiled in the kernel to assembly to check system calls

can check system call argument values, but...

- problems with pointer arguments
- too many system calls

Linux system call: open

```
open("foo.txt", O_RDONLY);
```

parameters:

- system call number: 2 ("`open`")
- argument 1: 0x7fffe983 (address of string "foo.txt")
- argument 2: 0 (value of "`O_RDONLY`")

very problematic to filter using BPF interface