Дополнительные главы практической безопасности

Михаил Воронов

m.voronov@gse.cs.msu.ru

Лекция 5: Rust type system

Agenda

1. Системы типов
2. Rust
3. Система типов Rust
4. Сравнение с C++
5. Unsoundness баги

Системы типов

Системы типов

Type systems

General concepts

- Type safety

Major categories

- Static vs. dynamic
- Manifest vs. inferred
- Nominal vs. structural
- Duck typing

Minor categories

- Dependent
- Flow-sensitive
- Gradual
- Intersection
- Latent
- Refinement
- **Substructural**
- Unique
- Abstract

See also

- Strong and weak typing

Субструктурные системы типов

	Exchange	Weakening	Contraction	Use
Ordered	—	—	—	Exactly once in order
Linear	Allowed	—	—	Exactly once
Affine	Allowed	Allowed	—	At most once
Relevant	Allowed	—	Allowed	At least once
Normal	Allowed	Allowed	Allowed	Arbitrarily

ADT

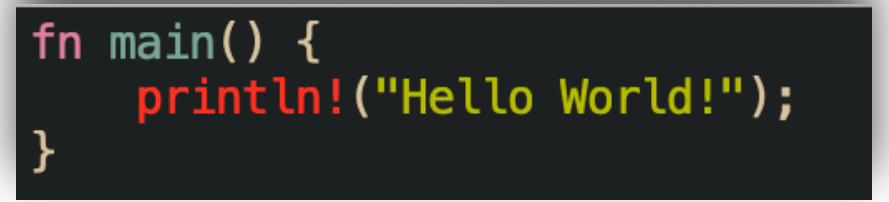
Новые типы обычно создаются с помощью двух операций sum или product.

```
union SumType {
    int32_t n;
    uint16_t s;
    uint8_t c;
};
```

```
struct ProductType {
    int32_t n;
    uint16_t s;
    uint8_t c;
};
```

Rust

Rust by examples



Rust by examples


```rust
use std::mem;

fn analyze_slice(slice: &[i32]) {
    println!("first element of the slice: {}", slice[0]);
    println!("the slice has {} elements", slice.len());
}

fn main() {
    let xs: [i32; 5] = [1, 2, 3, 4, 5];
    let ys: [i32; 500] = [0; 500];
    println!("first element of the array: {}", xs[0]);
    println!("number of elements in array: {}", xs.len());

    println!("array occupies {} bytes", mem::size_of_val(&xs));
    println!("borrow the whole array as a slice");
    analyze_slice(&xs);

    println!("borrow a section of the array as a slice");
    analyze_slice(&ys[1 .. 4]);

    println!("{}", xs[5]);
}
```

Rust by examples

```rust
use std::mem;

fn analyze_slice(slice: &[i32]) {
    println!("first element of the slice: {}", slice[0]);
    println!("the slice has {} elements", slice.len());
}

fn main() {
    let xs: [i32; 5] = [1, 2, 3, 4, 5];
    let ys: [i32; 500] = [0; 500];
    println!("first element of the array: {}", xs[0]);
    println!("number of elements in array: {}", xs.len());

    println!("array occupies {} bytes", mem::size_of_val(&xs));
    println!("borrow the whole array as a slice");
    analyze_slice(&xs);

    println!("borrow a section of the array as a slice");
    analyze_slice(&ys[1 .. 4]);

    println!("{}", xs[5]);
}
```

```
[12:14] :additional_chapters | rustc hello_world.rs
error: this operation will panic at runtime
  --> hello_world.rs:21:20
   |
21 |     println!("{}", xs[5]);
   |                    ^^^^^ index out of bounds: the length is 5 but the index is 5
   |
   = note: `#[deny(unconditional_panic)]` on by default

error: aborting due to previous error
```

Rust by examples


```rust
#[derive(Debug)]
struct Person {
    name: String,
    age: u8,
}

struct Unit;

struct Pair(i32, f32);

struct Point {
    x: f32,
    y: f32,
}

#[allow(dead_code)]
struct Rectangle {
    top_left: Point,
    bottom_right: Point,
}
```

Rust by examples


```rust
enum WebEvent {
    PageLoad,
    PageUnload,
    KeyPress(char),
    Paste(String),
    Click { x: i64, y: i64 },
}

fn inspect(event: WebEvent) {
    match event {
        WebEvent::PageLoad => println!("page loaded"),
        WebEvent::PageUnload => println!("page unloaded"),
        WebEvent::KeyPress(c) => println!("pressed '{}'.", c),
        WebEvent::Paste(s) => println!("pasted \"{}\".", s),
        WebEvent::Click { x, y } => {
            println!("clicked at x={}, y={}.", x, y);
        },
    }
}
```

Rust by examples: ownership


```rust
fn destroy_box(c: Box<i32>) {
    println!("Destroying a box that contains {}", c);
}

fn main() {
    let x = 5u32;

    let y = x;

    println!("x is {}, and y is {}", x, y);

    let a = Box::new(5i32);

    println!("a contains: {}", a);

    let b = a;

    //println!("a contains: {}", a);

    destroy_box(b);

    //println!("b contains: {}", b);
}
```

Rust by examples: mutability


```
fn main() {
    let immutable_box = Box::new(5u32);

    println!("immutable_box contains {}", immutable_box);

    //*immutable_box = 4;

    let mut mutable_box = immutable_box;

    println!("mutable_box contains {}", mutable_box);

    *mutable_box = 4;

    println!("mutable_box now contains {}", mutable_box);
}
```

Rust by examples: borrowing


```rust
fn eat_box_i32(boxed_i32: Box<i32>) {
    println!("Destroying box that contains {}", boxed_i32);
}

fn borrow_i32(borrowed_i32: &i32) {
    println!("This int is: {}", borrowed_i32);
}

fn main() {
    let boxed_i32 = Box::new(5_i32);
    let stacked_i32 = 6_i32;

    borrow_i32(&boxed_i32);
    borrow_i32(&stacked_i32);

    {
        // Take a reference to the data contained inside the box
        let _ref_to_i32: &i32 = &boxed_i32;

        // Error!
        // Can't destroy `boxed_i32` while the inner value is borrowed later in scope.
        eat_box_i32(boxed_i32);

        borrow_i32(_ref_to_i32);
    }

    eat_box_i32(boxed_i32);
}
```

Rust by examples: aliasing


```rust
struct Point { x: i32, y: i32, z: i32 }

fn main() {
    let mut point = Point { x: 0, y: 0, z: 0 };

    let borrowed_point = &point;
    let another_borrow = &point;

    println!("Point has coordinates: ({}, {}, {})",
                borrowed_point.x, another_borrow.y, point.z);

    // Error! Can't borrow `point` as mutable because it's currently
    // borrowed as immutable.
    let mutable_borrow = &mut point;

    println!("Point has coordinates: ({}, {}, {})",
                borrowed_point.x, another_borrow.y, point.z);

    let mutable_borrow = &mut point;

    mutable_borrow.x = 5;
    mutable_borrow.y = 2;
    mutable_borrow.z = 1;

    // let y = &point.y;

    // println!("Point Z coordinate is {}", point.z);

    println!("Point has coordinates: ({}, {}, {})",
                mutable_borrow.x, mutable_borrow.y, mutable_borrow.z);

    let new_borrowed_point = &point;
    println!("Point now has coordinates: ({}, {}, {})",
             new_borrowed_point.x, new_borrowed_point.y, new_borrowed_point.z);
}
```

Rust

Rust vs C++


```
std::vector<int> v { 10, 11 };
int *vptr = &v[1];
v.push_back(12);
std::cout << *vptr;
```

Rust vs C++

- Silent wraparound in C

```
int main() {
    unsigned int a = 4;
    a = a - 3;
    printf("%u\n",a-2);
}
mashimaro <~>  9:35AM % ./a.out
4294967295
```

-

```
fn main() {
    let mut a:u32 = 4;
    a = a - 3;
    println!("{}", a - 2);
}
```

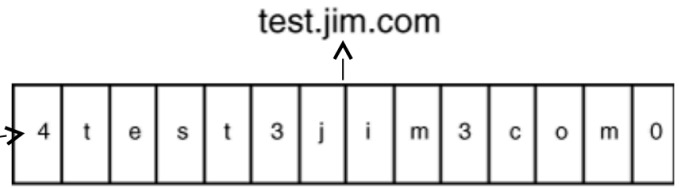
```
rustc 1.15.1 (021bd294c 2017-02-08)
thread 'main' panicked at 'attempt to subtract with overflow',
stack backtrace:
```

www-verimag.imag.fr/~mounier/Enseignement/Software_Security/19RustVsC.pdf

Rust vs C++

- `count` read as byte, then `count` bytes concatenated to `nameStr`

```
char *indx;
int count;
char nameStr[MAX_LEN]; //256
...
memset(nameStr, '\0', sizeof(nameStr));
...
indx = (char *)(pkt + rr_offset);
count = (char)*indx;
while (count){
  (char *)indx++;
  strncat(nameStr, (char *)indx, count);
  indx += count;
  count = (char)*indx;
  strncat(nameStr, ".", sizeof(nameStr) - strlen(nameStr));
}
nameStr[strlen(nameStr)-1] = '\0';
```



What if count = 128?

Sign extended then used in strncat

Type mismatch in Rust

```
char *strncat(char *dest, const char *src, size_t n);
```

http://www.informit.com/articles/article.aspx?p=686170&seqNum=6

www-verimag.imag.fr/~mounier/Enseignement/Software_Security/19RustVsC.pdf

C++ buggy snippet

2002 FreeBSD `getpeername()` bug (B&O Ch. 2)

- Kernel code to copy hostname into user buffer
  - `copy_from_kernel()` call takes signed `int` for size from user
  - `memcpy` call uses unsigned `size_t`
- What if adversary gives a length of "-1" for his buffer size?

```
#define KSIZE 1024
char kbuf[KSIZE]
void *memcpy(void *dest, void *src, size_t n);

int copy_from_kernel(void *user_dest, int maxlen){
        /* Attempt to set len=min(KSIZE, maxlen) */
    int len = KSIZE < maxlen ? KSIZE : maxlen;
        memcpy(user_dest, kbuf, len);
        return len;
}
```

Type mismatch in Rust

`(KSIZE < -1)` is false, so `len = -1`

`memcpy` casts `-1` to $2^{32}-1$

Unauthorized kernel memory copied out

www-verimag.imag.fr/~mounier/Enseignement/Software_Security/19RustVsC.pdf

Rust: soundness баги

Ожидания от Rust:

Реальность:

Rust: unsoundness баги

Soundness is a type system concept (actually originating from the study of logics) and means that the type system is "correct" in the sense that well-typed programs actually have the desired properties. For Rust, **this means well-typed programs cannot cause Undefined Behaviour**. This promise only extends to safe code however; for unsafe code, it is up to the programmer to uphold this contract.

Accordingly, we say that a library (or an individual function) is *sound* if it is impossible for safe code to cause Undefined Behavior using its public API. Conversely, the library/function is *unsound* if safe code *can* cause Undefined Behavior.

github.com/rust-lang/unsafe-code-guidelines/blob/636d140ce9c74ffc4d1fc082bef0771f238f64c9/reference/src/glossary.md#soundness-of-code--of-a-library

Rust: unsoundness баги

Filters | is:open label:"I-unsound 💥" | Labels 368 | Milestones 3 | New issue

Clear current search query, filters, and sorts

64 Open ✓ 225 Closed — Author | Label | Projects | Milestones | Assignee | Sort

- **"C-unwind" ABI is unsound with "-Cpanic=abort"** C-bug F-c_unwind I-unsound P-medium T-compiler requires-nightly — 2
  #83116 opened 9 days ago by RalfJung
- **Regressions with large (2-4GB) stack arrays on large stacks** A-LLVM C-bug I-unsound P-medium T-compiler regression-from-stable-to-nightly — 8
  #83060 opened 11 days ago by joshtriplett
- **Bounds check fail/integer overflow in unicode_data::skip_search via cc when building with LTO** A-LLVM C-bug I-unsound ICEBreaker-LLVM P-critical regression-from-stable-to-nightly — 19
  #82890 opened 15 days ago by saethlin 1.52.0
- **`fn() -> Out` is a valid type for unsized types `Out`, and it implements `FnOnce<(), Output = Out>`** A-associated-items A-closures A-traits A-typesystem C-bug F-unboxed_closures I-nominated I-unsound P-high T-lang — 16
  #82633 opened 23 days ago by steffahn
- **repr(C) is unsound on MSVC targets** A-ffi C-bug I-unsound O-windows O-windows-msvc P-high T-lang — 103
  #81996 opened on 11 Feb by mahkoh
- **Safe code (ui/issues/issue-69225-SCEVAddExpr-wrap-flag.rs) miscompiles under llvm9 + codegen-units=1** 1 — 3
  A-LLVM C-bug I-unsound P-medium T-compiler regression-untriaged
  #81946 opened on 10 Feb by the8472

CTF: sandbox escape


```
extern crate code;
use code::code;

fn main() {
    // hidden
    code();
}
```

CTF: hints

1. unsafe

2. __libc_start_main

```
#[no_mangle]
#[link_section=".text"]
pub static __libc_start_main : [u8; _] = [
  ...
];
```

3. Pin unsoundness (internals.rust-lang.org/t/unsoundness-in-pin/11311)

4. Coherence bug (github.com/rust-lang/rust/issues/57893)

5. Any other unsoundness bug from the Rust compiler repo