bounds-checking?

so far: mistake is no bounds checking

run input function without telling it how much space

so, we avoid this by checking sizes, right?

common problem: bugs in size checking code

integer overflow (or underflow)

integer overflow example (1)

```
item *load_items(int len) {
  int total_size = len * sizeof(item);
  if (total_size >= LIMIT) {
    return NULL;
  }
  item *items = malloc(total_size);
  for (int i = 0; i < len; ++i) {
    int failed = read_item(&items[i]);
    if (failed) {
      free(items);
      return NULL;
    }
  }
  return items;
}
```

len = 0x4000 0001
sizeof(item) = 0x10

total_size =
0x4 0000 0010

integer overflow example (1)

```
item *load_items(int len) {
  int total_size = len * sizeof(item);
  if (total_size >= LIMIT) {
    return NULL;
  }
  item *items = malloc(total_size);
  for (int i = 0; i < len; ++i) {
    int failed = read_item(&items[i]);
    if (failed) {
      free(items);
      return NULL;
    }
  }
  return items;
}
```

len = 0x4000 0001
sizeof(item) = 0x10

total_size =
0x4 0000 0010

integer overflow example (2)

```
/* adapted from https://project-zero.issues.chromium.org/issues/42
   Windows Kernel bug! */
char *FormatNumber(char *source, short source_len) {
    unsigned short dest_size = source_len * 6;
    char *dest = malloc(dest_size);
    char *p = dest;
    for (unsigned short i = 0; < source_len; i += 1) {
        *p++ = "0123456789ABCDEF"[*source >> 4];
        *p++ = "0123456789ABCDEF"[*source & 0xF];
        *p++ = '␣';
        source++;
    }
}
```

integer under/overflow: real example

part of another Google Chrome exploit by Pinkie Pie:

```
// In graphics command processing code:
uint32 ComputeMaxResults(size_t size_of_buffer) {
    return (size_of_buffer - sizeof(uint32)) / sizeof(T);
}
size_t ComputeSize(size_t num_results) {
    return sizeof(T) * num_results + sizeof(uint32);
}
// exploit: size_of_buffer < sizeof(uint32)
```

result: write 8 bytes after buffer

- sometimes overwrites data pointer

via https://blog.chromium.org/2012/05/tale-of-two-pwnies-part-1.html

exercise

assume return address immediately after items array:
Q: what first input number to provide?
Q: how to encode return address replacement `0x12345678`?

```
void vulnerable() {
  int items[100];
  int count;
  bool success =
    try_read_input(&count);
  if (!success) { ... }
  if (count * sizeof() >= sizeof(items)) {
    printf("cannot␣handle␣that␣many\n"); return;
  }
  for (int i = 0; i < count; i += 1) {
    if (!try_read_input(&items[i])) {
      printf("preature␣end␣of␣input\n"); return;
    }
  }
  process_items(items);
}
```

overflow and undefined behavior

C standard: some things are *undefined behavior*

C compiler can do anything in those cases

signed integer overflow is undefined

unsigned integer overflow must wrap around

-fsanitize=undefined

```
int x = INT_MAX -1; int y = 5; printf("%d\n", x * y);
```

compile with `-fsanitize=undefined`:

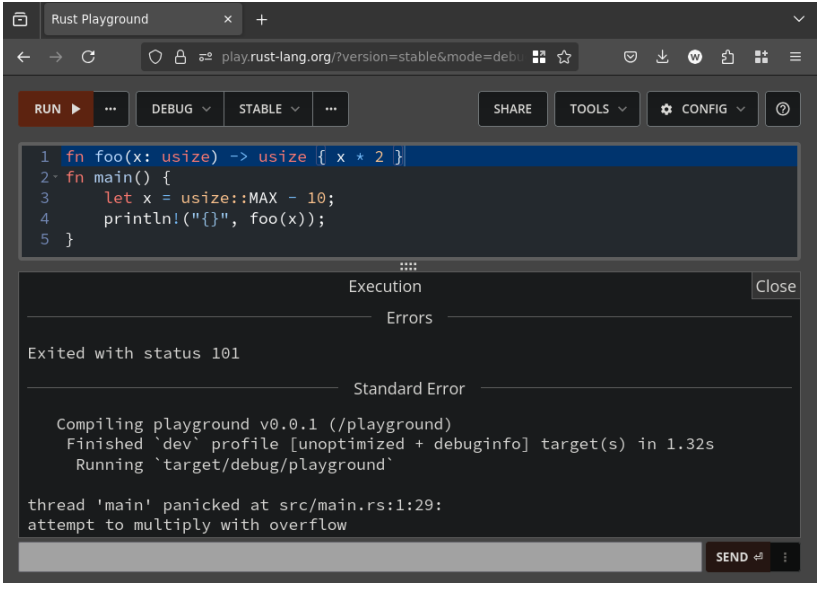
```
test.c:...: runtime error: signed integer
overflow: 2147483646 * 5 cannot be represented in
type 'int
```

compile with `-ftrapv`: `Aborted (core dumped)`

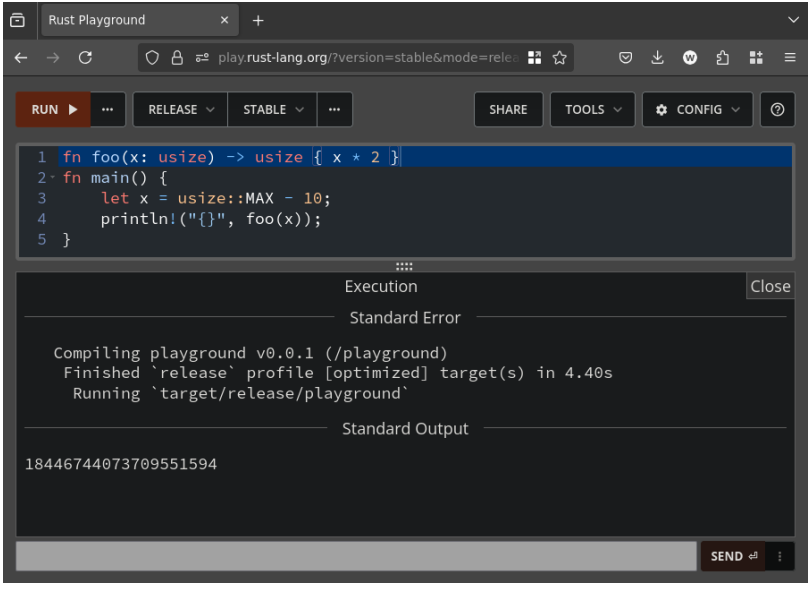
```
unsigned x = INT_MAX -1; unsigned y = 5; printf("%u\n", x * y
```

compile with `-fsanitize=undefined` or `-ftrapv`: NO ERROR

in Rust (1)


Rust Playground
play.rust-lang.org/?version=stable&mode=debu
RUN
DEBUG
STABLE
SHARE
TOOLS
CONFIG
1 fn foo(x: usize) -> usize { x * 2 }
2 fn main() {
3 let x = usize::MAX - 10;
4 println!("{}", foo(x));
5 }
Execution
Close
Errors
Exited with status 101
Standard Error
Compiling playground v0.0.1 (/playground)
Finished `dev` profile [unoptimized + debuginfo] target(s) in 1.32s
Running `target/debug/playground`
thread 'main' panicked at src/main.rs:1:29:
attempt to multiply with overflow
SEND

in Rust (2)


Rust Playground
play.rust-lang.org/?version=stable&mode=relea
RUN
RELEASE
STABLE
SHARE
TOOLS
CONFIG
1 fn foo(x: usize) -> usize { x * 2 }
2 fn main() {
3 let x = usize::MAX - 10;
4 println!("{}", foo(x));
5 }
Execution
Close
Standard Error
Compiling playground v0.0.1 (/playground)
Finished `release` profile [optimized] target(s) in 4.40s
Running `target/release/playground`
Standard Output
18446744073709551594
SEND

in Rust (3)

```
let x = usize::MAX - 10; let y = 10usize;
println!("{} {}", x.saturating_mul(2), y.saturating_mul(2));
// 18446744073709551615 20
    // 18446744073709551615==usize::MAX
println!("{:?} {:?}", x.checked_mul(2), y.checked_mul(2));
// None Some(30)
println!("{} {}", x.wrapping_mul(2), y.wrapping_mul(2));
// 18446744073709551594 20
    // 18446744073709551594==usize::MAX-21
```

backup slides