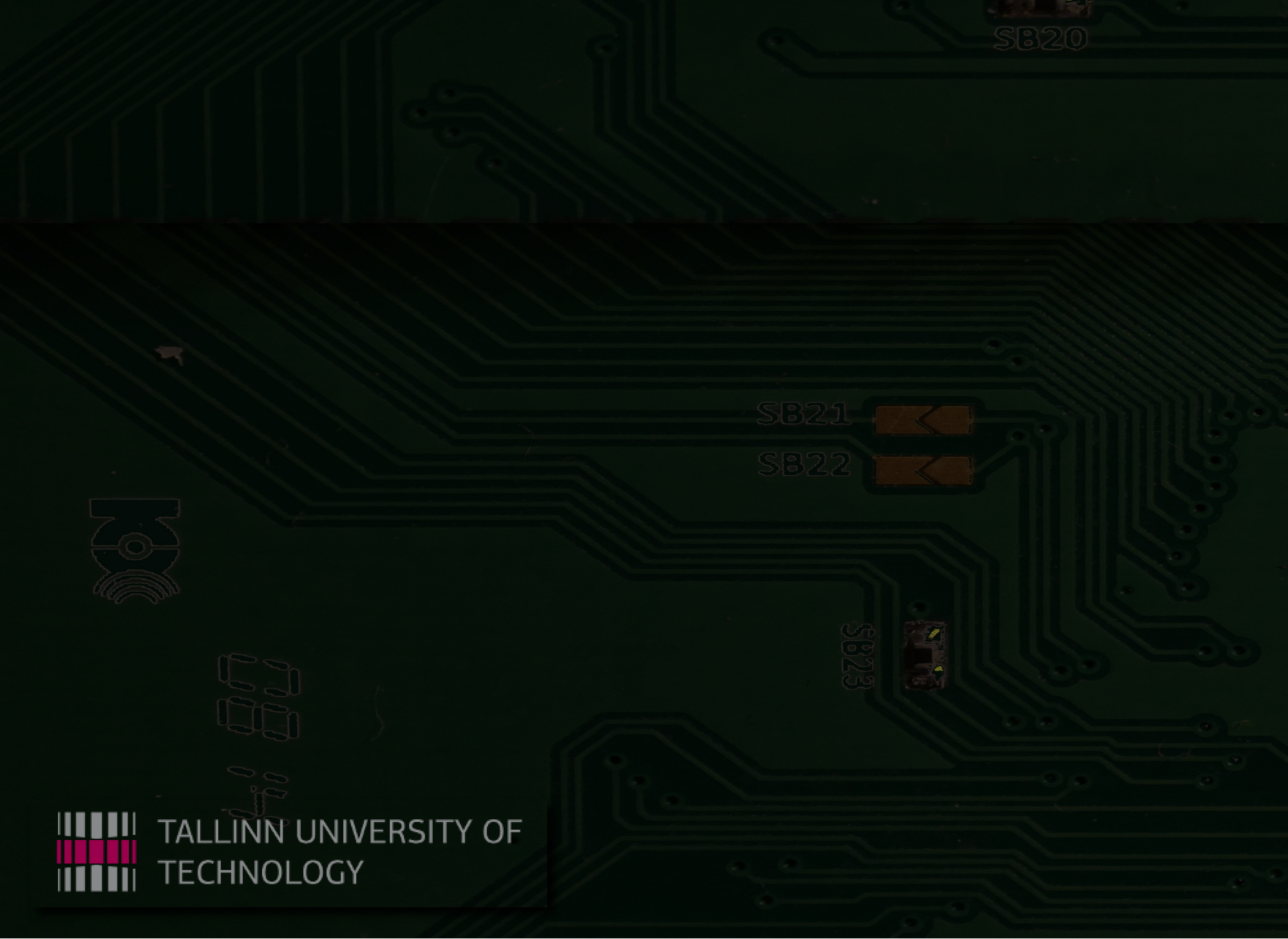
SECURE PROGRAMMING OF RESOURCE CONSTRAINED DEVICES

Jens Getreu

2018-01-16

TALLINN UNIVERSITY OF TECHNOLOGY



TALLINN UNIVERSITY OF
TECHNOLOGY

AGENDA

1. Resource Constrained Devices
2. The Heartbleed vulnerability
3. The Rust Programming Language
4. Conclusion and recommendations

RESOURCE CONSTRAINED DEVICES

TALLINN UNIVERSITY OF TECHNOLOGY

DEFINITION

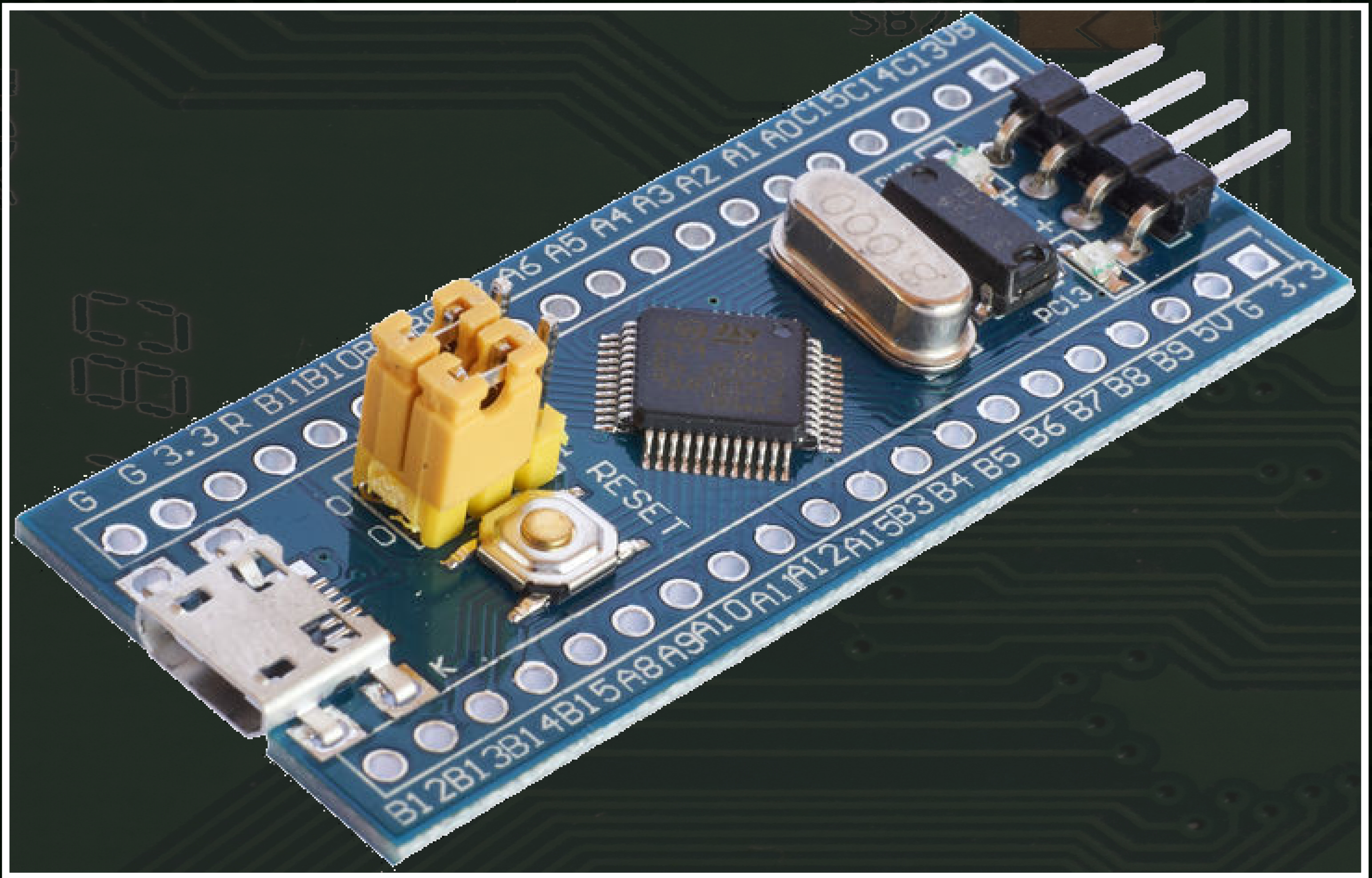
Resource constrained device

is a computer with very limited processing and storage capabilities, designed for low energy consumption.

Examples

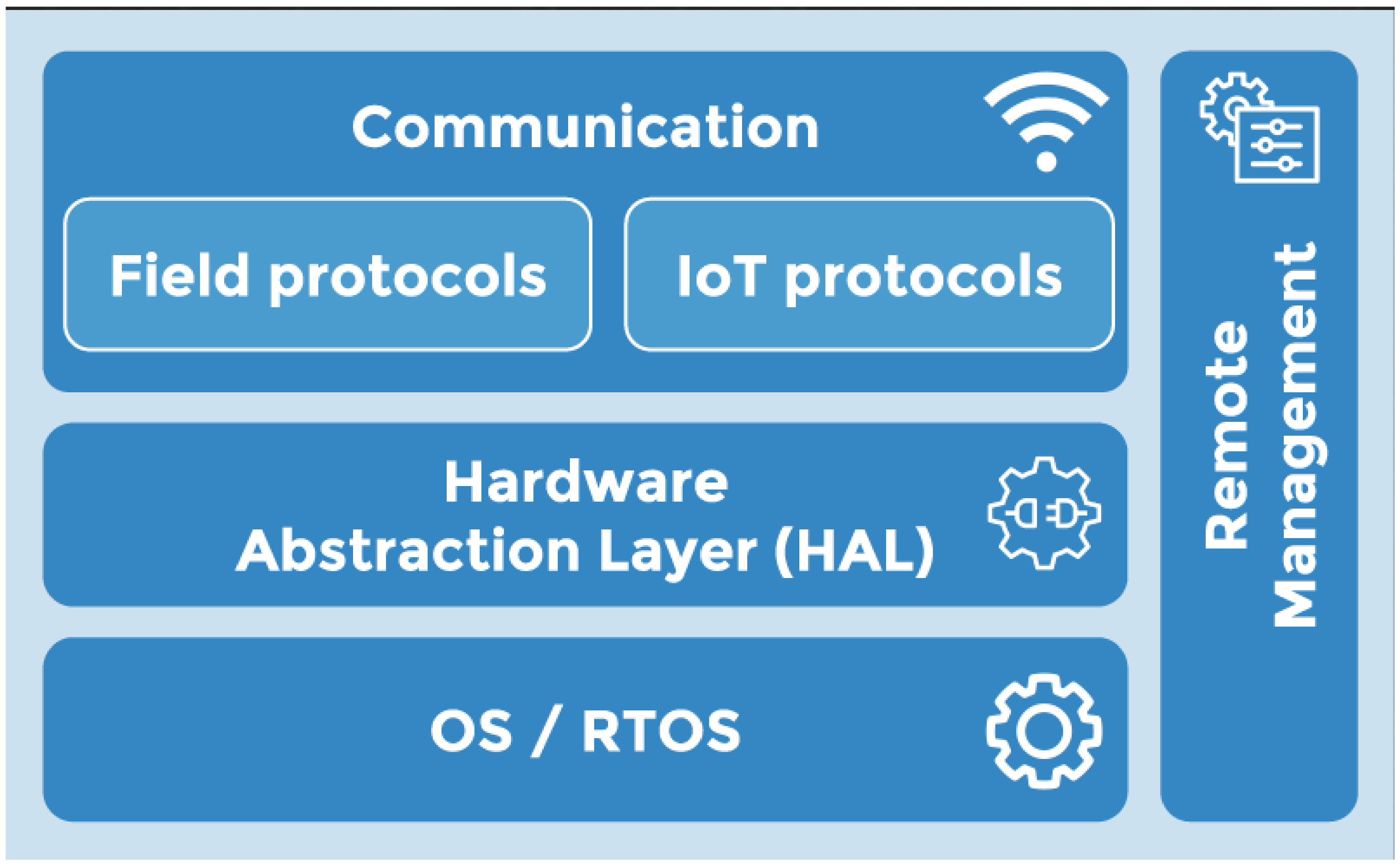
- Wireless Sensors
- The “Things” in the Internet of Things

HARDWARE



Example: STM32F103C8T6 Blue-Pill

SOFTWARE ARCHITECTURE



Resource Constrained Device

SPECIAL REQUIREMENTS

meta-functional attributes

RESOURCE CONSTRAINED DEVICES ARE VULNERABLE

Attacks

Mirai (2016) / IoT reaper / IoTroop / Heartbleed (2014)

Causes

- RCD are as complex
- Internet connectivity does not generate excess profit.

  -> Devices are poorly configured and highly insecure
- C/C++ do not provide *memory and thread safety*

THE HEARTBLEED VULNERABILITY

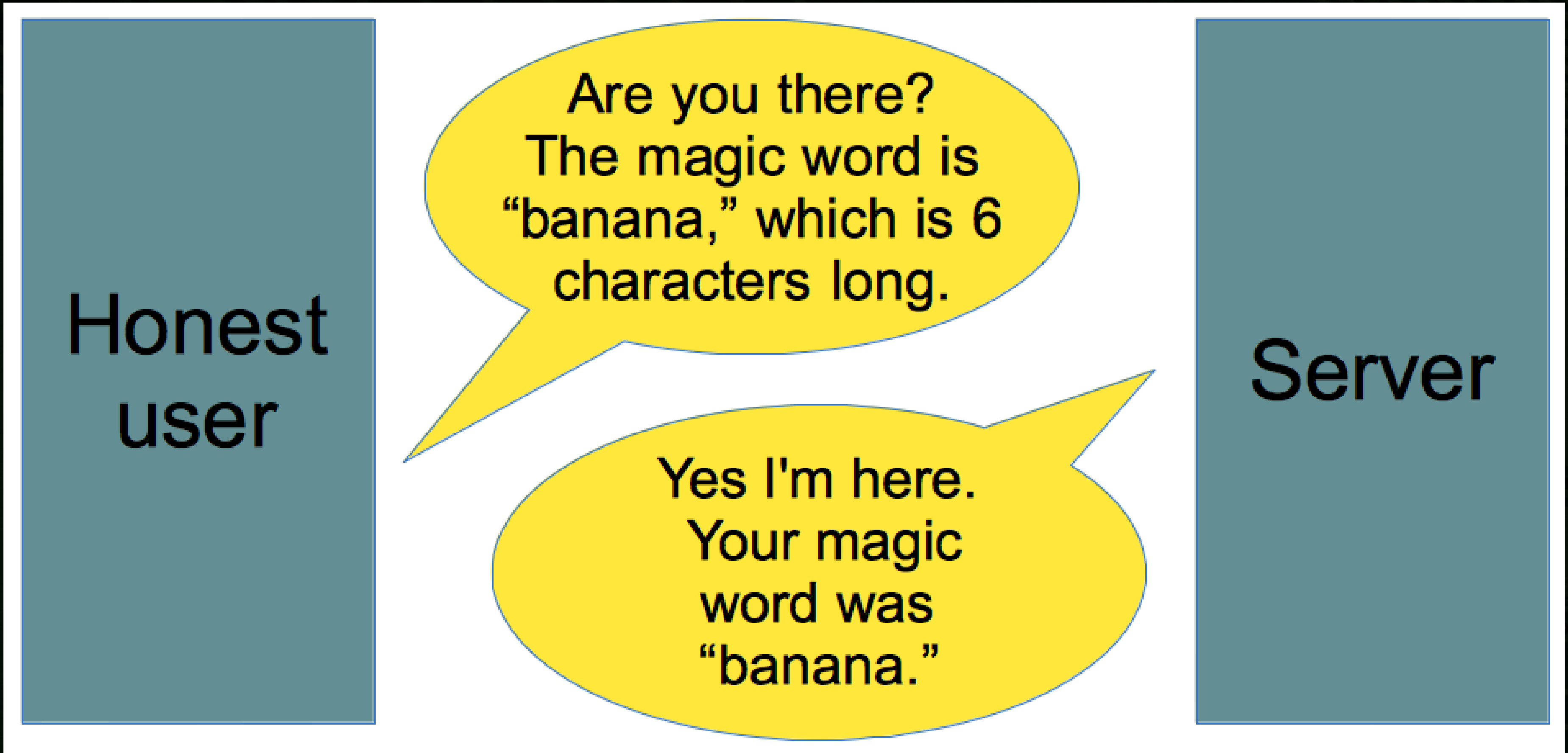
TALLINN UNIVERSITY OF TECHNOLOGY

MEMORY SAFETY RELATED VULNERABILITIES

2/3 of all Linux kernel vulnerabilities are memory safety related.

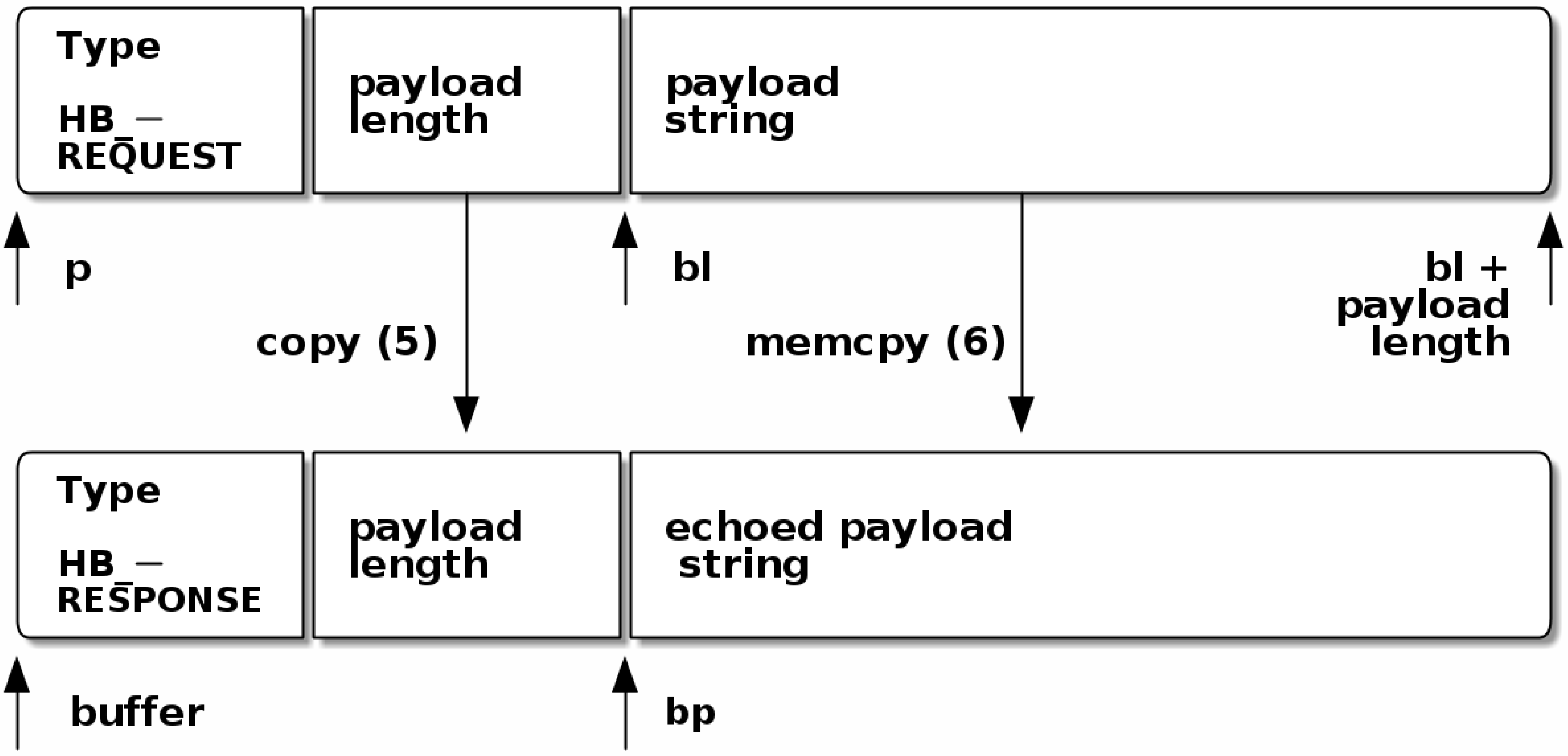
CWE ID	Name
120	Buffer Copy without Checking Size of Input
125	Out-of-bounds Read
126	Buffer Over-read
122	Heap-based Buffer Overflow
401	Memory Leak

THE HEARTBEAT TLS EXTENSION 1



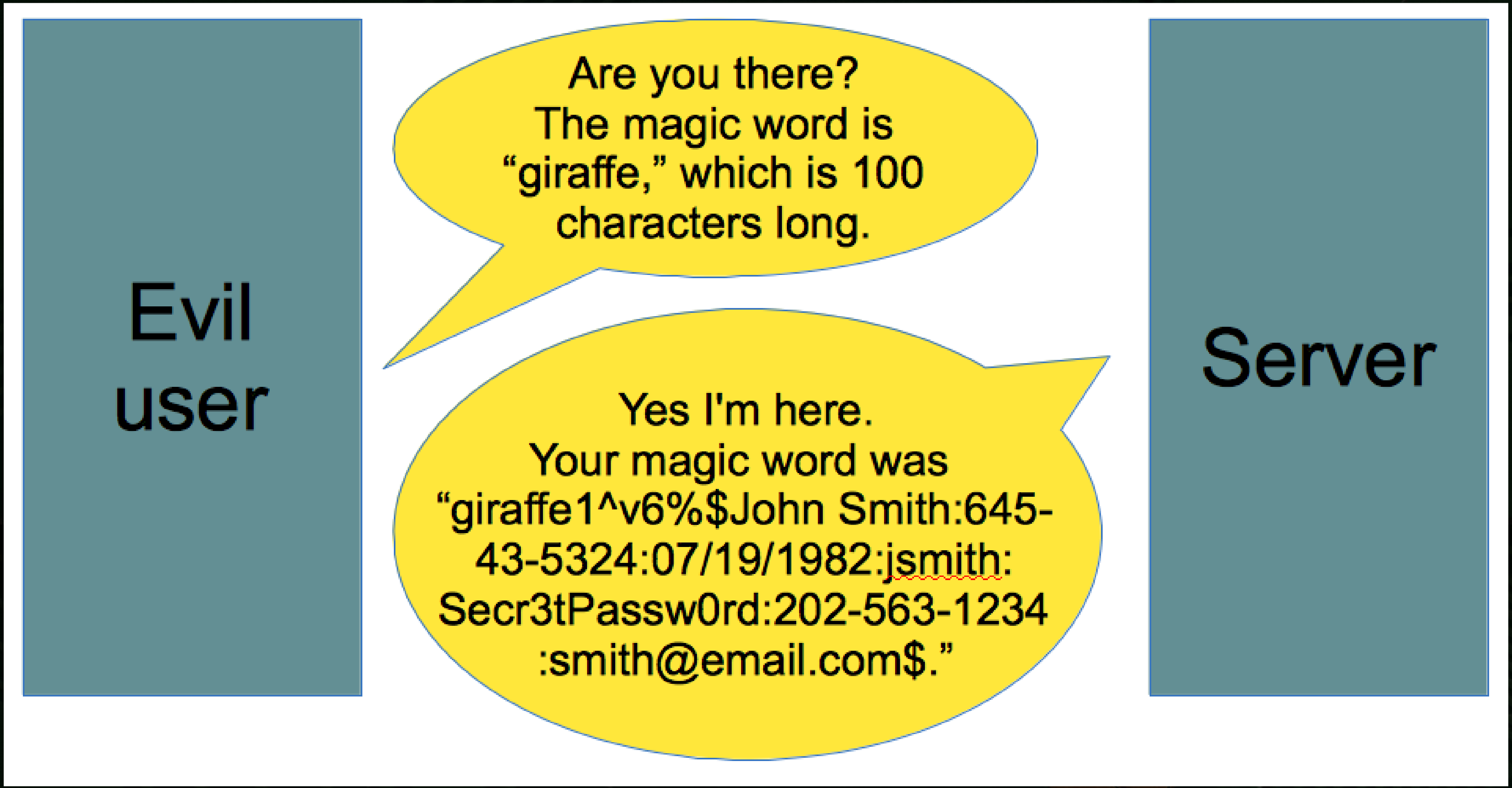
TLS Heartbeat protocol

THE HEARTBEAT TLS EXTENSION 2



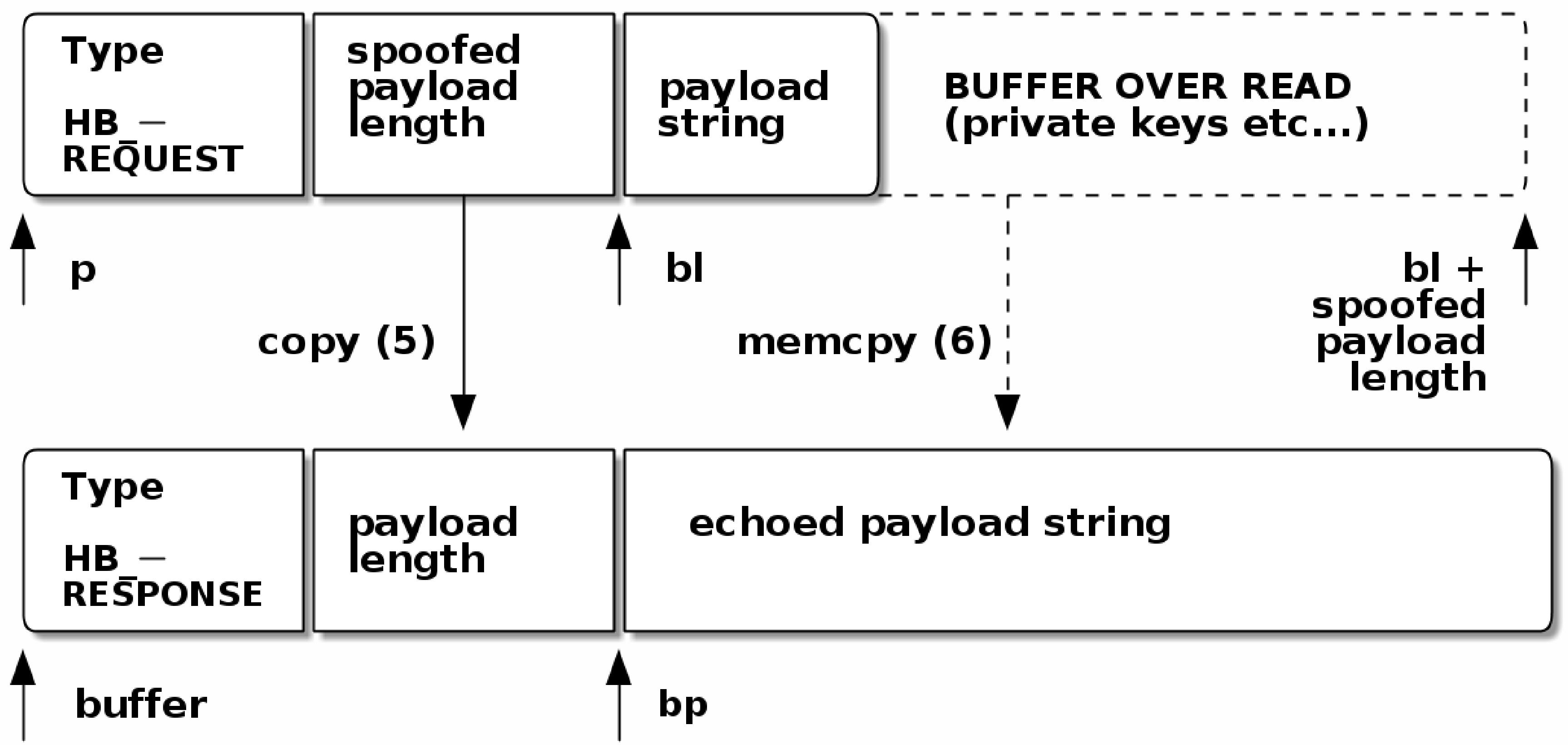
TLS Heartbeat protocol

THE HEARTBLEED VULNERABILITY 1



Heartbleed vunerability

THE HEARTBLEED VULNERABILITY 2



VULNERABLE C CODE

```c
unsigned char *p = &s->s3->rrec.data[0], *pl;
unsigned short hbtype;
unsigned int payload;
unsigned int padding = 16;

hbtype = *p++;      //<1>
n2s(p, payload);    //<2>
pl = p;

//... folded lines ...

if (hbtype == TLS1_HB_REQUEST)
        {
        unsigned char *buffer, *bp;
        Cint r;

        buffer = OPENSSL_malloc(1 + 2 + payload + padding); //<3>
```

THE RUST PROGRAMMING LANGUAGE

TALLINN UNIVERSITY OF TECHNOLOGY

FEATURES

- guaranteed memory safety
- zero-cost abstractions
- threads without data races

References: Firefox 57, Maidsafe, Parity-Bitcoin-Client

COULD HEARTBLEED HAVE HAPPENED WITH RUST?

```
fn tls1_process_heartbeat (s: Ssl) -> Result<(), isize> {
  const PADDING: usize = 16;

  let p = s.s3.rrec;
  let hbtype:u8 = p[0];
  let payload:usize = ((p[1] as usize) << 8) + p[2] as usize; // <1>

  let mut buffer: Vec<u8> = Vec::with_capacity(1+2+payload+PADDING);
  buffer.push(TLS1_HB_RESPONSE);
  buffer.extend(p[1..1+2].iter().cloned());                    // <2>
  buffer.extend(p[3..3+payload].iter().cloned());              // <3>

  let mut rng = rand::thread_rng();                            // <4>
  buffer.extend( (0..PADDING).map(|_|rng.gen::<u8>())
        .collect::<Vec<u8>>() );
```

TALLINN UNIVERSITY OF TECHNOLOGY

HEARTBLEED EXPLOIT PACKAGE

```
let s: Ssl = Ssl {
    s3 : Rrec{
            rrec:  &[TLS1_HB_REQUEST, 1, 1, 14, 15, 16, 17,
                     18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
                     28, 29, 30, 31, 32]
    }
};
tls1_process_heartbeat(s).unwrap();
}
```

SYSTEM RESPONSE AFTER HEARTBLEED ATTACK

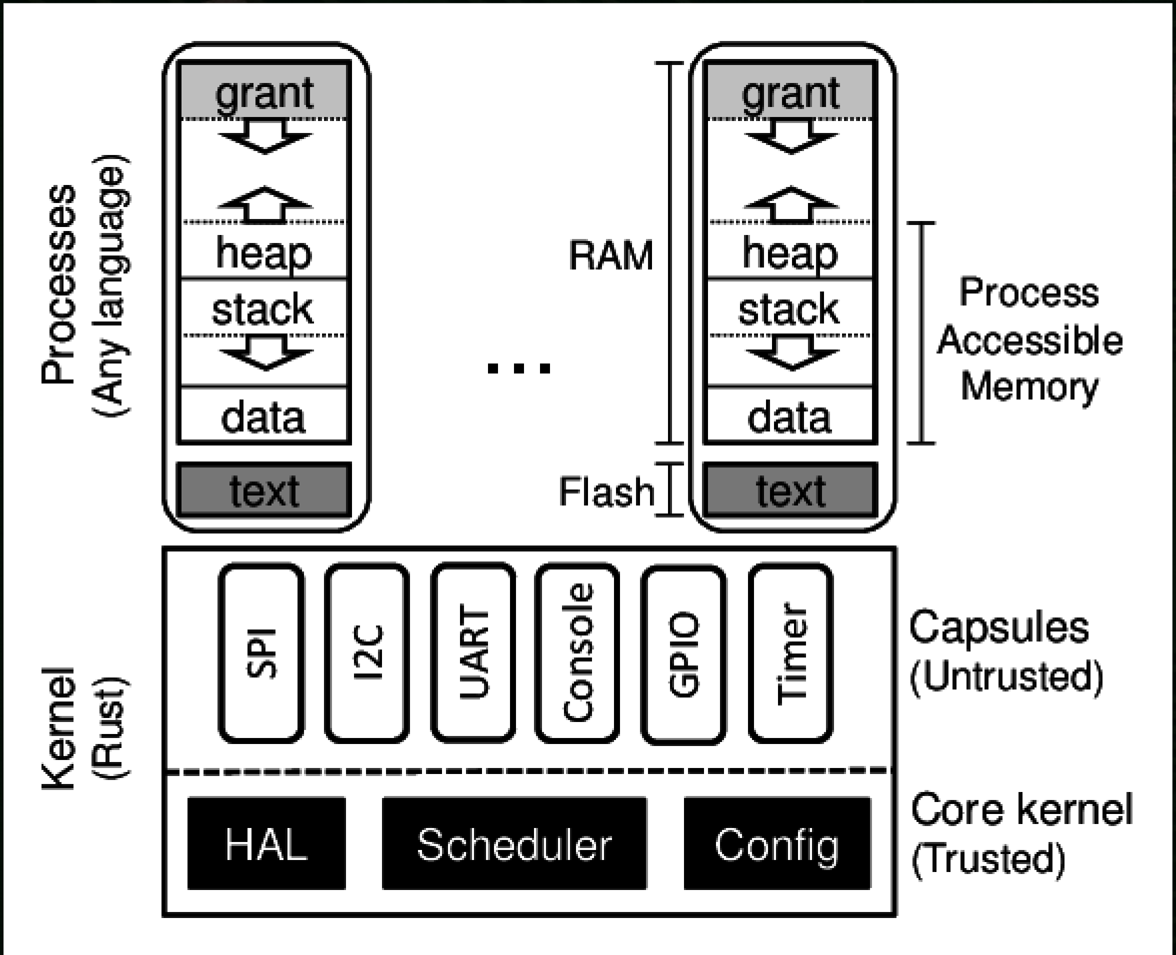
```
thread '<main>' panicked at 'assertion failed:
index.end <= self.len()',
Process didn't exit successfully:
`target/release/heartbeat` (exit code: 101)
```

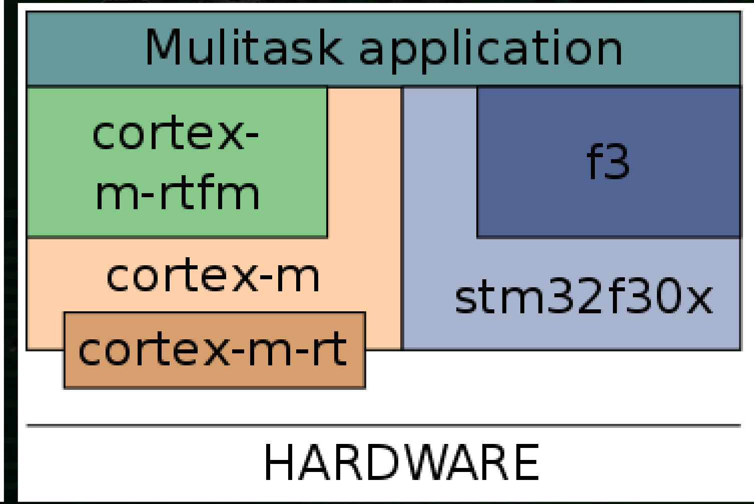
TALLINN UNIVERSITY OF TECHNOLOGY

RESOURCE SHARING IN RUST

Resource sharing type	Aliasing	Mutation	Example
move ownership	no	yes	`let a = b`
shared borrow	yes	no	`let a = &b`
mutable borrow	no	yes	`let a = &mut b;`

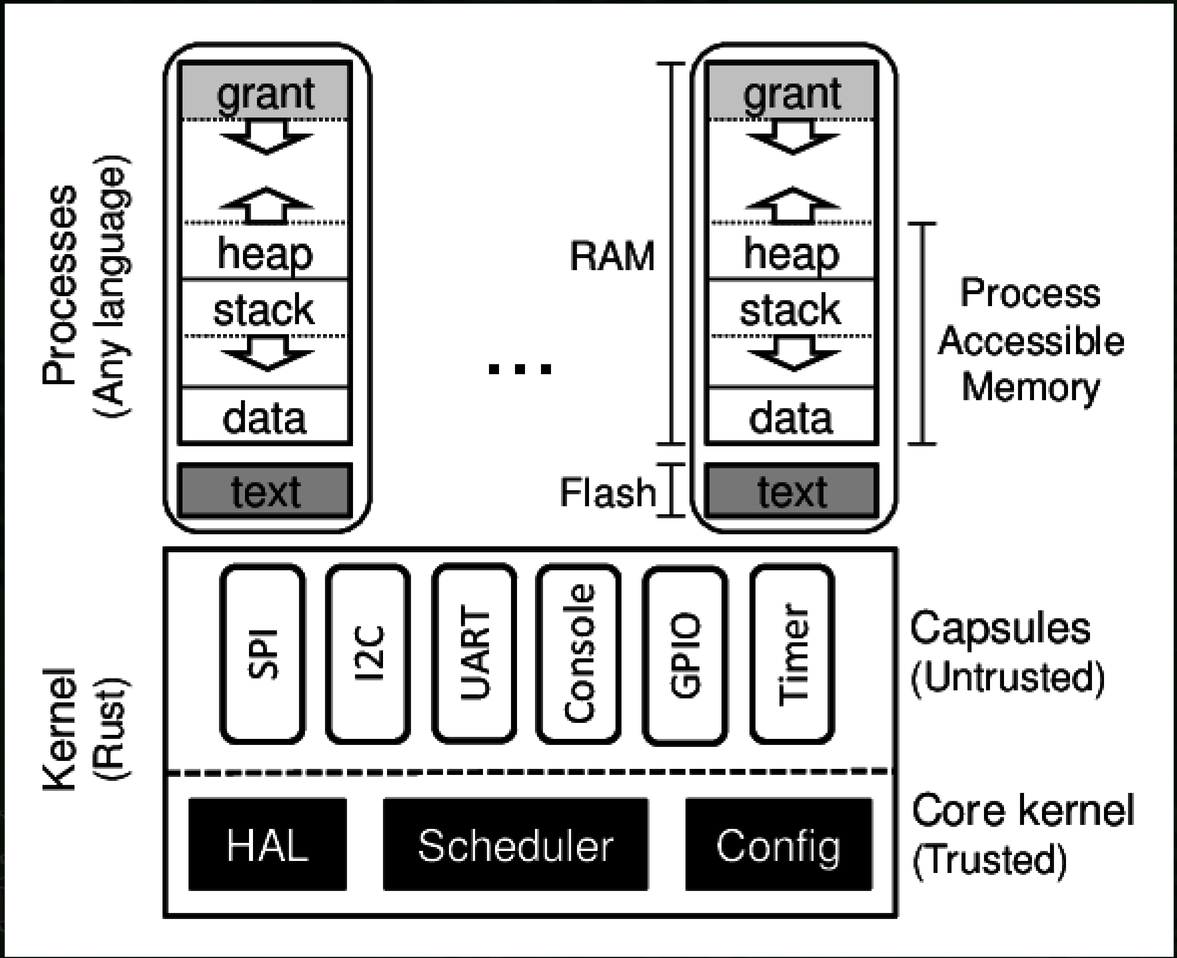
RUST OPERATING SYSTEMS





TockOS versus RTFM

TOCK-OS



TockOS architecture

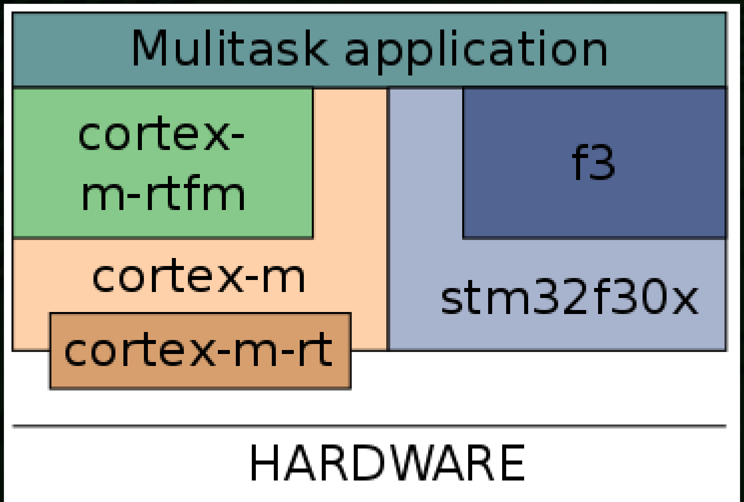
TOCK-OS PRIMITIVES

```
struct App {
    count: u32,
    tx_callback: Callback,
    rx_callback: Callback,
    app_read: Option<AppSlice<Shared,u8>>,
    app_write: Option<AppSlice<Shared,u8>>,
}
pub struct Driver {
    app: TakeCell<App>,
}

new_app () {
    ...
    driver.app.map(|app| {app.count = app.count + 1});
}
```

TALLINN UNIVERSITY OF TECHNOLOGY

REAL TIME FOR THE MASSES



RTFM architecture

RTFM PRIMITIVES

```
threshold.raise(
    &SHARED, |threshold| {
        let shared = SHARED.access(priority, threshold);
        shared.mode.set(Mode::Bounce)
    }
);
```

RUST IN EMBEDDED SYSTEMS

Challenges

- secure concurrency
  - (lightweight) threads
  - interrupt driven
- “zero zero” cost abstractions
- yet only few drivers available
- yet only few platforms are supported
- no `std`-library

CONCLUSION AND RECOMMENDATION

LIMITATIONS

Rust for Resource Constrained Devices:
Technology is mature, ready for production.

- only few drivers are available
- only few platforms are supported

Doable, typical amount of lines of code 10k
(vs Linux Kernel 4.14: 25 Mio lines)

OPPORTUNITIES

Rust eradicates memory safety related vulnerabilities, improves systematically the security of

- field sensors
- *consumer IoT*

Contribute to **F**ree and **O**pen **S**ource **S**oftware.

THANK YOU!

TALLINN UNIVERSITY OF
TECHNOLOGY

REFERENCES

TALLINN UNIVERSITY OF TECHNOLOGY

ARTICLES

1. A. Levy, B. Campbell, P. Dutta, B. Ghena, P. Levis, and P. Pannuto, “The Case for Writing a Kernel in Rust,” in Proceedings of APSys ’17, Mumbai, India, 2017, p. 7.
2. M. Antonakakis et al., “Understanding the Mirai Botnet,” in Proceedings of the 26th USENIX Security Symposium, Vancouver, 2017, pp. 1093–1110.
3. R. Clayton, “A New IoT Botnet Storm is Coming,” Check Point Research, 19-Oct-2017. [Online]. Available. [Accessed: 19-Dec-2017].

PICTURE CREDITS 1

1. “File:STM32 Blue Pill perspective.jpg - STM32duino wiki.” [Online]. Available [Accessed: 12-Dec-2017]

2. Eclipse IoT Working Group, “The Three Software Stacks Required for IoT Architectures - IoT software requirements and how to implement them using open source technology.” The Eclipse Foundation, Sep-2016 [Online]. Available.

3. A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 1, pp. 11–33, 2004.

PICTURE CREDITS 2

1. T. B. Lee, “How does the Heartbleed attack work?,” Vox, 10-Apr-2014. [Online]. Available. [Accessed: 14-Jan-2018].
2. “Tock Design,” *[The] Tock Embedded Operating System*. [Online]. Available. [Accessed: 14-Dec-2017]
3. J. Aparicio, “Fearless concurrency in your microcontroller,” *Embedded in Rust.* 09-May-2017 [Online]. Available. [Accessed: 17-May-2017]