RustNL GOSIM 2024 EUROPE

Do you know…

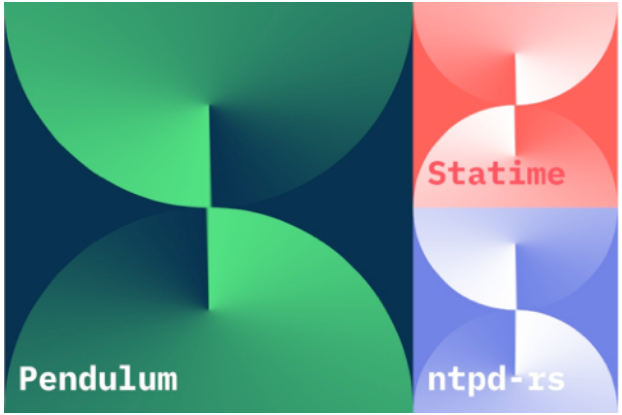
… WHO WROTE YOUR SOFTWARE?

Marc Schoolderman
Tweede golf

May 6th, 2024

Software must become safer

- Fewer vulnerabilities
- A more reliable Internet
- Resilient critical infrastructure




communities
research
One
often
offers
subject
Internet
known
information
site
allow
world
fact
resources
data
easily
members
person
sites
Linux
tool
guides
find
email
GIMP
Scribus
product
Greenstone
without
different
need
tools
interface
WordPress
option
provide
used
digital
another
Source
application
make
web
like
easy
active
way
project
time
also
manage
simply
people
think
amazing
developers
just
catalog
CAPTION
every
system
Firefox
community
operating
look
creating
support
book
developed
even
software
libraries
many
using
Library
staff
online
options
made
proprietary
popular
cost
share
freedom
browser
right
websites
means
important
systems
librarians
figure
source
patrons
see
keep
projects
much
help
search
development
FIGURE
number
one
around
free
open
code
process
Evergreen
website
applications
computer
always
management
available
might
services
SIDEBAR
use
Open
making
work
OPAC
collections
several
first
ILS
library's
access
new
Ubuntu
know
Koha
makes
offer
content
allows
Drupal
users
release
create
page
want
Joomla
running
worldwide
security
others
library


source
open
library
libraries
software
community
many
Library
applications
Open
SIDEBAR
CAPTION
application
without
make
web
like
system
Firefox
use
free
access
patrons
see
figure
systems
librarians
projects
Source
also
time
just
tool
find
Greenstone
WordPress
Scribus
product
digital
another
site
need
data
tools
information
Internet
offers
One
website
Ubuntu
Koha
right
online
staff
means
development
code
computer
OPAC
work
one
people
think
used
way
using
popular
keep
much
worldwide
users
collections
security
allows
content
makes
making
want
create
page
release
process
management
developed
look
creating
different
fact
resources
members
tweede golf


source
open
library
software
community
many
libraries
applications
CAPTION
SIDEBAR
Open
use
patrons
figure
access
new
free
one
work
computer
OPAC
Firefox
system
web
like
make
need
without
application
Internet
Ubuntu
Koha
website
offer
making
using
Library
tools
data
site
information
members
tool
find
time
think
people
right
online
staff
developed
means
Evergreen
Joomla
Drupal
security
FIGURE
code
services
collections
worldwide
popular
freedom
projects
keep
browser
WordPress
Greenstone
Scribus
GIMP
Linux
guides
product
digital
another
easily
resources
fact
allow
world
different
amazing
support
creating
look
process
management
release
create
page
want
makes
content
allows
always
might
running
others
computer
development
librarians
patrons
see
number
around
several
first
users
provide
option
manage
simply
even
options
proprietary
Source
interface
person
sites
subject
research
communities
known
often
offers
every
important
search
help
websites
ILS
library's
tweede golf

tweede golf

Ferris mascot: Karen Rustad Tölva
CC0

Word cloud: Nicole C. Baratta
CC-BY-NC-SA 2.0

Who wrote `sudo-rs`?

Our team

- Developers: 👨🏻‍💻 👨🏼‍💻 👨🏻‍💻 👨🏽‍💻

Who wrote `sudo-rs`?

Our team

- Developers: 👨‍💻 👨🏼‍💻 👨‍💻 👨🏽‍💻
- Outside contributors: 💁🏻‍♀️ 🙋🏻 🙋🏾 🙋🏻 🕵️‍♂️ 🙋🏿 🙋🏻 🙋🏽 🙋🏻 🙋🏽‍♀️ 👮‍♂️ 🙋🏻 🙋🏻 🧜‍♀️

Who wrote `sudo-rs`?

OUR TEAM

- **Developers:** 👨🏻‍💻 👨🏼‍💻 👨🏻‍💻 👨🏽‍💻
- **Outside contributors:** 🙋🏼‍♀️ 🙋🏻‍♂️ 🙋🏾‍♂️ 🙋🏻‍♂️ 🕵️‍♂️ 🙋🏿‍♂️ 🙋🏻‍♂️ 🙋🏽‍♂️ 🙋🏻‍♂️ 🙋🏽‍♀️ 👨‍✈️ 🙋🏻‍♂️ 🙋🏻‍♂️ 👩‍🎤

NOT OUR TEAM

- **11 Developers** 🤷‍♂️🤷🏻‍♀️🤷‍♂️🤷🏼‍♂️🤷‍♂️🤷🏾‍♂️🤷‍♂️ 🤷‍♂️🤷🏼‍♀️🤷‍♂️🤷‍♂️
- **1 Bot** 🤖
- **4 GitHub teams** …🤷‍♂️🤷‍♂️🤷‍♂️🤷‍♂️🤷‍♂️🤷‍♂️🧑‍🎤🤷‍♂️🤷‍♂️🤷‍♂️🧑‍🎤🤷‍♂️ 🤷‍♂️🤷‍♂️🤷‍♂️🦹🤷‍♂️🤷‍♂️🤷‍♂️🤷‍♂️🤷‍♂️🧑‍✈️🤷‍♂️🤷‍♂️👩‍🎤🤷‍♂️🕵️🤷‍♂️🤷‍♂️🤷‍♂️🤷‍♂️🤷‍♂️🥷🤷‍♂️ ??

Who wrote `sudo-rs`?


```
The following individuals can publish updates for your dependencies:

 1. alexcrichton via crates: glob, libc, log
 2. huonw via crates: glob, libc, log
 3. rust-lang-owner via crates: glob, libc, log
 4. JohnTitor via crates: libc
 5. KodrAus via crates: log
 6. gnzlbg via crates: libc
 7. joshtriplett via crates: libc
 8. sfackler via crates: log
```

```
$ cargo supply-chain publishers
```

How many lines of code is `sudo-rs`?

OUR TEAM

- `sudo-rs`: **20.320 lines**

NOT OUR TEAM

- `log`: **5594 lines**
- `glob`: **2160 lines**
- `libc`: **121.914 lines (bindings)**

```
$ cargo vet
```

How much of `sudo-rs` is our work?

Conclusions:

- **At least** 13.5% of the lines of active code
- **At most** 33% of the people involved
- `sudo-rs` has **minimal** dependencies (best-case scenario)

Running example: pet project “`cargo pulse`”

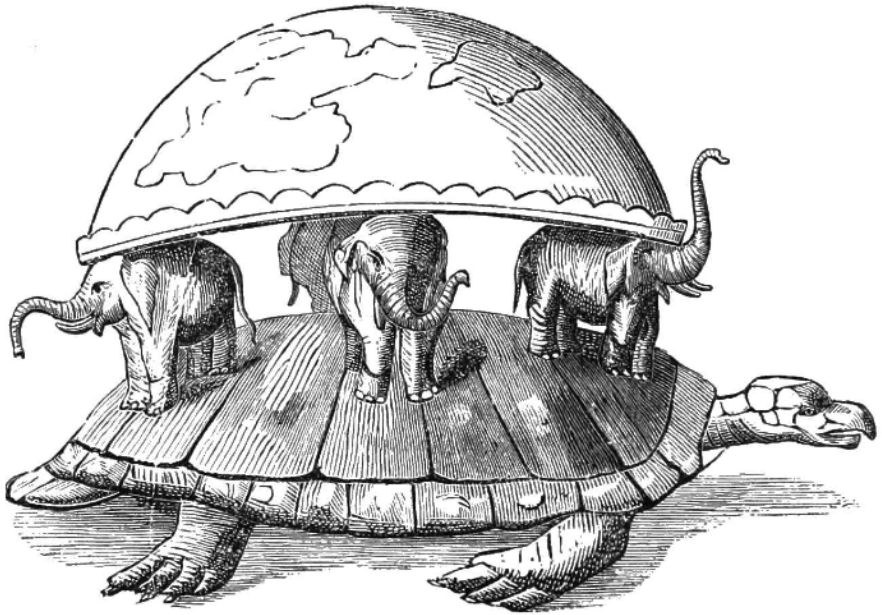
ME:

- **1** contributor
- 181 lines of Rust
- **7 dependencies**

NOT ME:

- 99 contributors, 28 teams
- 4.2 million lines of code,
  1.2 million not audited

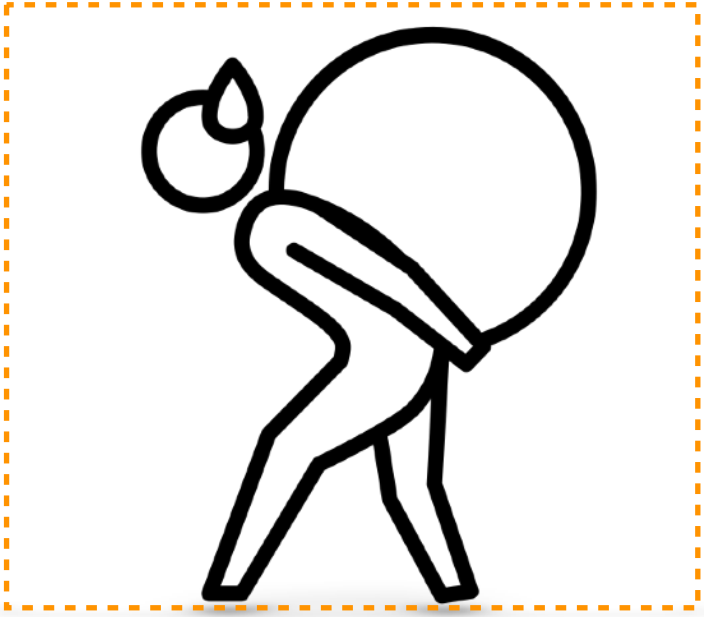
- **194 indirect dependencies**



Burden Problem

Trust Problem

Burden Problem



Trust Problem

Burden: Version management

- **Easy-to-Package?**

```
🔴 cargo-pulse v0.1.0 (/Users/squell/cargo-pulse)
   ├── async-trait v0.1.80 (0.1.77 in debian)
   ├── cargo_metadata v0.15.4 (in debian)
   ├── chrono v0.4.38 (0.4.31 in debian)
   ├── colored v2.1.0 (in debian)
🔴 ├── crates_io_api v0.8.2
   |   ├── chrono v0.4.38 (0.4.31 in debian)
   |   ├── futures v0.3.30 (in debian)
   |   ├── reqwest v0.11.27 (0.11.24 in debian)
   |   ├── serde v1.0.199 (1.0.195 in debian)
   |   ├── serde_derive v1.0.199 (1.0.195 in debian)
   |   ├── serde_json v1.0.116 (1.0.111 in debian)
   |   ├── serde_path_to_error v0.1.16 (0.1.9 in debian)
   |   ├── tokio v1.37.0 (1.35.1 in debian)
   |   └── url v2.5.0 (in debian)
   ├── octocrab v0.31.2 (in debian)
   └── tokio v1.37.0 (1.35.1 in debian)
```

```
$ cargo debstatus
```

Burden: Version management

- **Duplicate, Incompatible Versions**
- `Cargo.toml` **up to date?**
  - (*Compatibility with other versions*)
- `Cargo.lock` **usually not included**
  - (*Versions known to work*)

```
⌛ base64 v0.13.1 (outdated, 0.21.7 in debian)
⌛ └── pem v1.1.1 (outdated, 3.0.3 in debian)
      └── jsonwebtoken v8.3.0 (in debian)

  base64 v0.21.7 (in debian)

  bitflags v1.3.2 (in debian)

  bitflags v2.5.0 (2.4.2 in debian)

⌛ ring v0.16.20 (outdated, 0.17.5 in debian)
   └── jsonwebtoken v8.3.0 (in debian)

  ring v0.17.8 (0.17.5 in debian)

  syn v1.0.109 (in debian)

  syn v2.0.60 (2.0.48 in debian)

⌛ untrusted v0.7.1 (outdated, 0.9.0 in debian)
⌛ └── ring v0.16.20 (outdated, 0.17.5 in debian)
      └── jsonwebtoken v8.3.0 (in debian)

  untrusted v0.9.0 (in debian)
```

```
$ cargo tree -d
```

Burden: License management

- **Publish as `Apache-2.0-OR-MIT`?**
- **Distribute binaries?**
- **Respect copyleft licenses!**

```
(Apache-2.0 OR MIT) AND BSD-3-Clause (1)
(MIT OR Apache-2.0) AND Unicode-DFS-2016 (1)
0BSD OR Apache-2.0 OR MIT (1)
Apache-2.0 (2)
Apache-2.0 OR BSL-1.0 (1)
Apache-2.0 OR ISC OR MIT (5)
Apache-2.0 OR MIT (144)
Apache-2.0 OR MIT OR Zlib (3)
Custom License File (2)
ISC (4)
MIT (29)
MIT OR Unlicense (1)
MPL-2.0 (1)
```

```
$ cargo license
```

Burden Problem

Trust Problem

Trust: Code Quality

- Undefined Behaviour (UB) impossible in normal Rust
- `unsafe` Rust: “trust me, I’m a real programmer”
- Pet project: 181 lines of safe Rust

  … but 20’000+ `unsafe` expressions under the hood

```
?  │   ├── snafu 0.7.5
☢  │   │   ├── backtrace 0.3.69
☢  │   │   │   ├── addr2line 0.21.0
☢  │   │   │   │   ├── gimli 0.28.0
☢  │   │   │   │   ├── object 0.32.1
☢  │   │   │   │   │   └── memchr 2.6.4
?  │   │   │   │   └── rustc-demangle 0.1.23
?  │   │   │   ├── cfg-if 1.0.0
☢  │   │   │   ├── libc 0.2.149
🔓 │   │   │   ├── miniz_oxide 0.7.1
🔓 │   │   │   │   └── adler 1.0.2
```

```
$ cargo geiger
```

Trust: Vulnerabilities

- **We know how to solve this!**
- But do all developers:
  - report vulnerabilities?
  - admit embarrassing mistakes?
  - actively update versions?

RUSTSEC
The Rust Security Advisory Database

```
Crate:      rustls
Version:    0.21.10
ID:         RUSTSEC-2024-0336
Dependency tree:
rustls 0.21.10
├── tokio-rustls 0.24.1
│   └── hyper-rustls 0.24.2
│       └── octocrab 0.29.3
│           └── cargo-pulse 0.1.0
└── hyper-rustls 0.24.2

Crate:      simple_asn1
Version:    0.6.0
ID:         RUSTSEC-2021-0125
Dependency tree:
simple_asn1 0.6.0
└── jsonwebtoken 8.3.0
    └── octocrab 0.29.3
        └── cargo-pulse 0.1.0

error: 2 vulnerabilities found!
```

```
$ cargo audit, cargo deny
```

Trust: Build time security

- Dependencies come with build scripts, not sandboxed!
  - ➡ `build.rs`

```
$ cargo build
   Compiling cargo-pulse v0.1.0 (/home/cargo-pulse)
   Compiling proc-macro2 v1.0.69
   Compiling unicode-ident v1.0.12
   Compiling libc v0.2.149
   Compiling backdoor v0.1.0
[sudo: authenticate] Password:
```

- Known problem in PyPI (Python) and npm (Node.js) repositories

  https://jfrog.com/webinar/identifying-and-avoiding-malicious-packages-2/

Trust: Who is our Source of Truth?

- Do we know *"Josh Triplett", "John Titor", "gnzlbg", … ?*
  - ➡ **How** do we know we are getting our code from them?
  - ➡ Can we **trust** them to protect their credentials?
  - ➡ Do they **respond** to incidents?
  - ➡ Can they be **coerced** to do something?

XZ Utils logo by "Jia Tan"

Trust: No proper authentication for dependencies

- **SSL** certificates only authenticates the package repository
- **Signed commits** are weakly authenticated (“this is my SSH key”)
- **OpenPGP** unpopular, “Web of Trust” is broken
- Linux Foundation project: **https://trustoverip.org/**

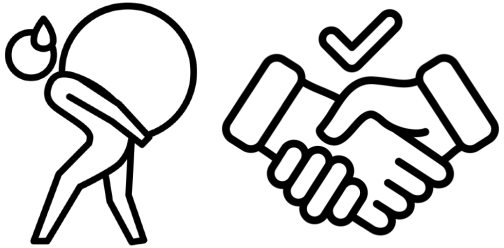
TRUST Over IP FOUNDATION

What to do?

What not to do?

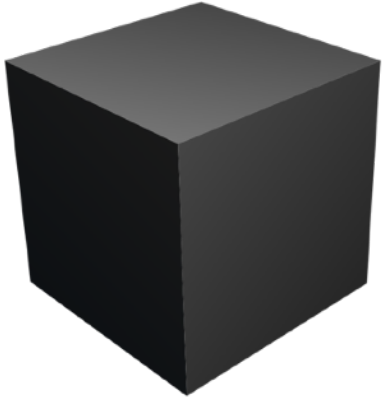
This is not a problem we can “fix”

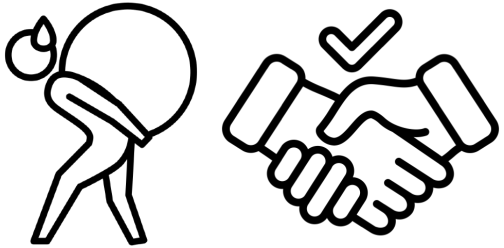
- **Modern software is complex**
- ***Many-and-small* dependencies**
  - ➡ Large “Software Bill of Materials”
  - ➡ Allows analysis and risk management
- ***Few-but-large* dependencies, big standard library**
  - ➡ Software bloat
  - ➡ Hard to change bad design choices



Bad solution: avoiding Rust or Open source

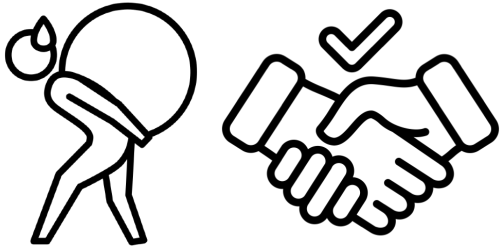
- “Trust problem” is universal, Rust helps keep it under control
- Proprietary software **obscures** problems





Bad solution: duplicating code

- Locks you out of bug fixes
- No vulnerability reporting
- Loses licensing information



Low hanging fruit

- Critically evaluate need for dependencies, check `crates.io` statistics
- Give your users a choice (feature flags)
- Participate in RUSTSEC
- Work towards a standard set of “common dependencies”
- Learn from Linux distributions: reproducible builds, hygiene checks, …

  - https://tweedegolf.nl/en/blog/104/dealing-with-dependencies-in-rust
  - https://www.memorysafety.org/blog/reducing-dependencies-in-sudo/

Security is a trade-off

Security is a trade-off

THANK YOU

More information available at:

https://tweedegolf.nl/en/blog

GOSIM 2024 EUROPE