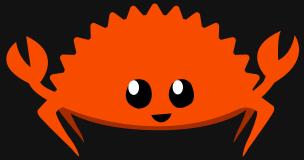
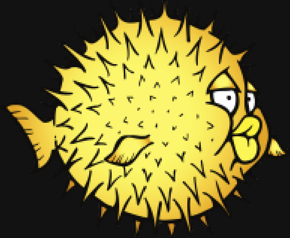
Malcolm Still

- 👨‍💻 Senior Software Engineer @ Swordbreaker
- https://github.com/malcolmstill

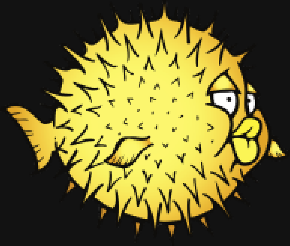




The crab and the pufferfish

Applying OpenBSD's Secure Software Design Pattern in Rust

OpenBSD



- Unix-like operating system
    - Specifically a BSD
        - See also: FreeBSD, NetBSD
    - Forked from NetBSD in 1995 by Theo de Raadt
- Security focussed
- https://www.openbsd.org/innovations.html
    - privdrop
        - pledge(2) 2015
        - unveil(2) 2018
    - privsep
        - First example OpenSSH 2002
        - …actually Qmail or Postfix 1998?

privdrop

Start with high privilege then lower privilege during execution

privdrop

Classic approach: change user running process

E.g. process starts as `root` (uid = 0) then subsequently lowers itself to some other user `alice` (uid = 6001)

privdrop

OpenBSD introduces new sandboxing primitives pledge(2) and unveil(2)

unveil(2)

Hide parts of the filesystem from the program

pledge(2)

Promise to only run subset of all available syscalls

Kernel will kill process if any non-promised syscalls are made

pledge(2)

```c
int pledge(const char *promises, const char *execpromises);
```

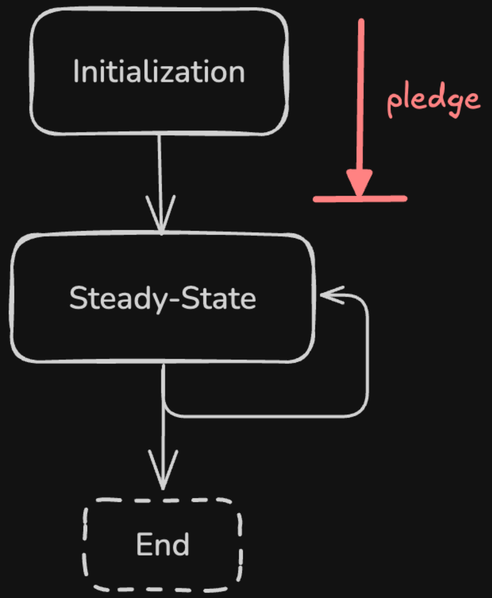
```c
pledge("stdio rpath wpath inet", NULL)
```

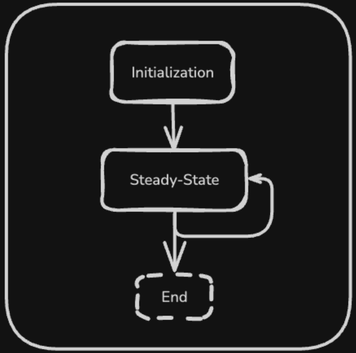
Promises are "bundles" of syscalls

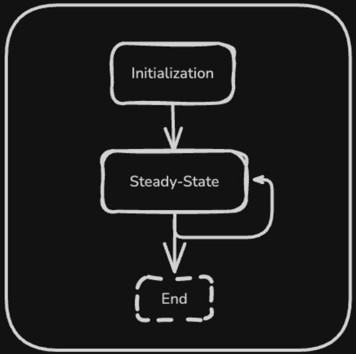
pledge(2)

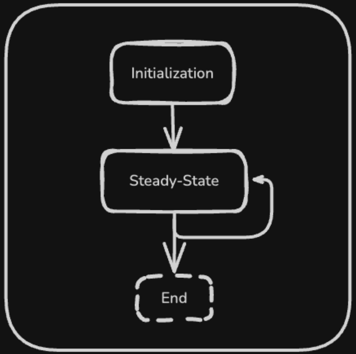
```c
int main(void) {
    if (pledge("stdio rpath", NULL) == -1) err(1, "pledge stdio rpath");

    int fd = open("hello_world.txt", O_RDONLY);
    if (fd == -1) err(1, "open");

    if (pledge("stdio", NULL) == -1) err(1, "pledge stdio");

    char buf[128];
    ssize_t n;
    while ((n = read(fd, buf, sizeof(buf))) > 0) {
      write(STDOUT_FILENO, buf, n);
    }

    close(fd);
    return 0;
}
```

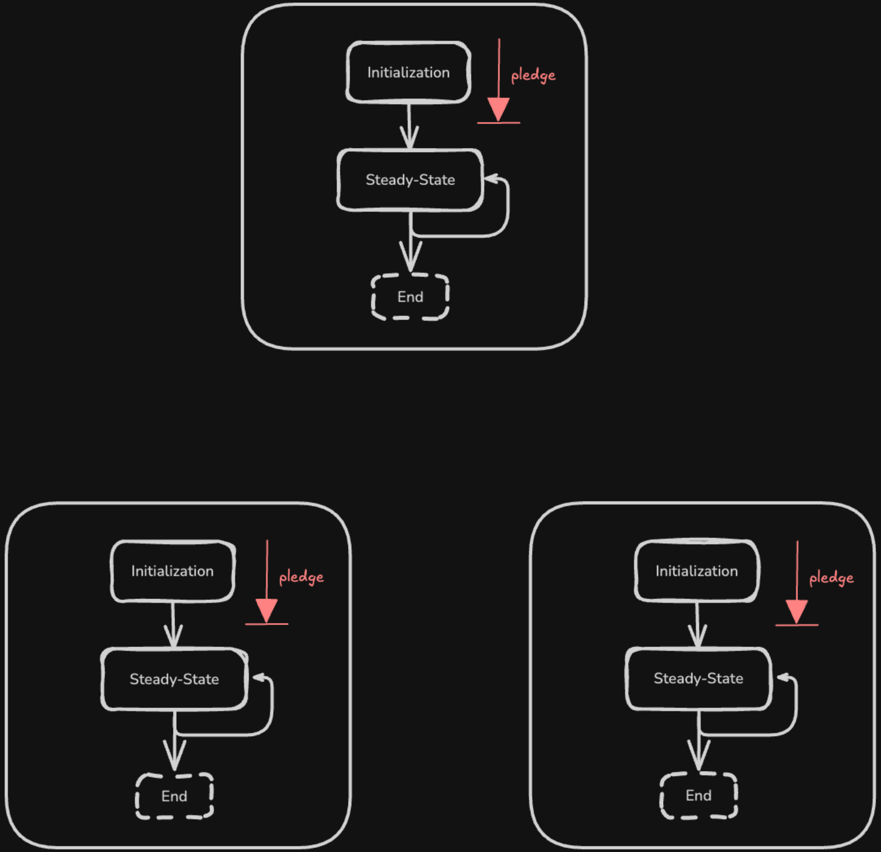
How do we apply pledge / unveil in practice?


Initialization
pledge
Steady-State
End


Initialization
Steady-State
End


Initialization
Steady-State
End


Initialization
Steady-State
End



Initialization

pledge

Steady-State

End

Initialization

pledge

Steady-State

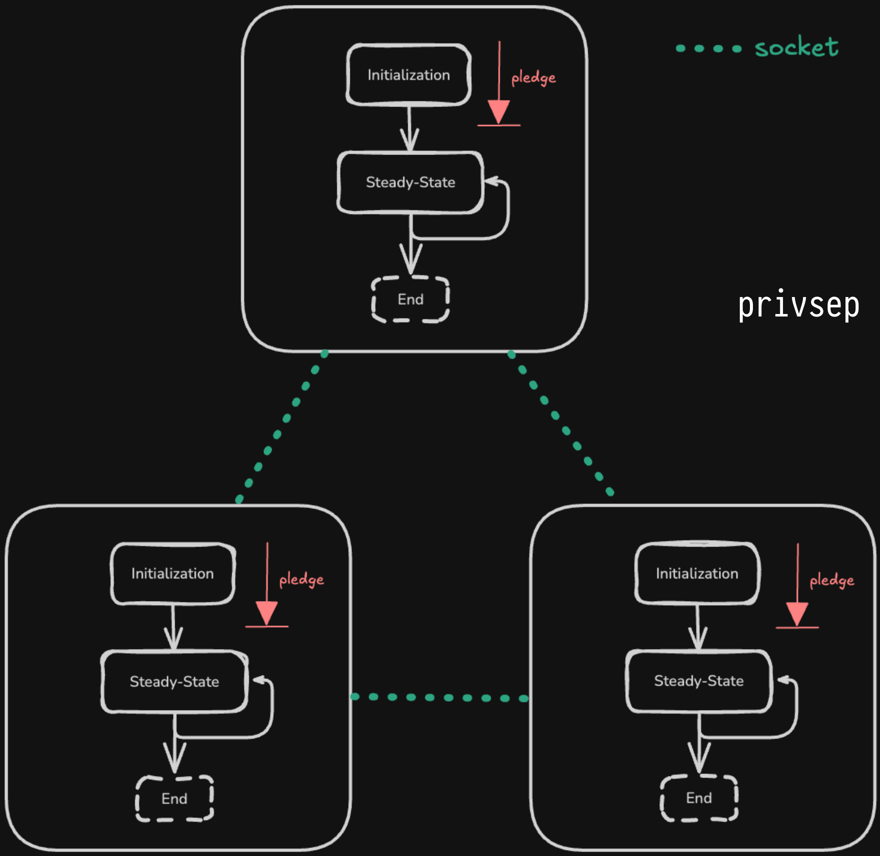
End

Initialization

pledge

Steady-State

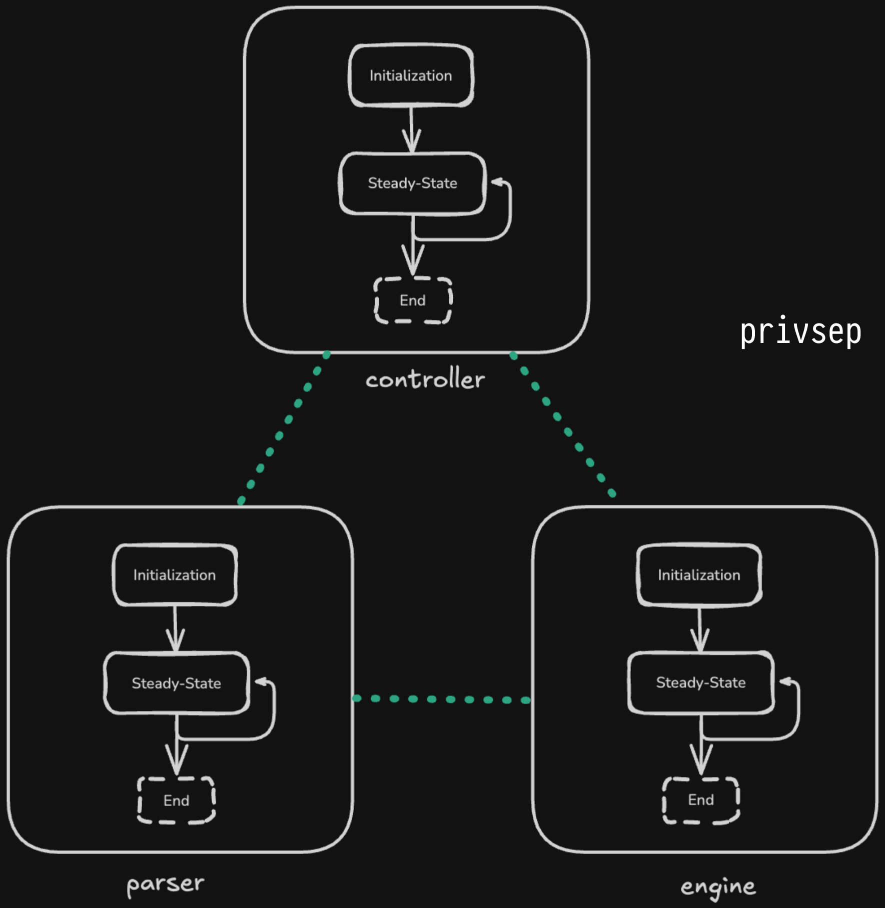
End



• • • • socket
Initialization
pledge
Steady-State
End
Initialization
pledge
Steady-State
End
Initialization
pledge
Steady-State
End

privsep

privsep


Initialization
Steady-State
End
controller
Initialization
Steady-State
End
parser
Initialization
Steady-State
End
engine

What does this pattern look like in rust?

Ingredients

pledge / unveil

- `pledge-rs` crate for `pledge(2)`
- `unveil-rs` crate for `unveil(2)`

privsep

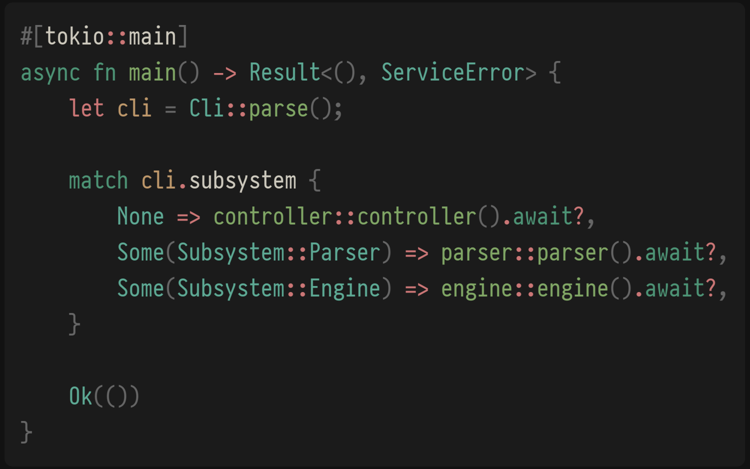
- `std::process::Command` for `fork+exec`
- `clap` crate for top-level subsystem switch

message-oriented channel

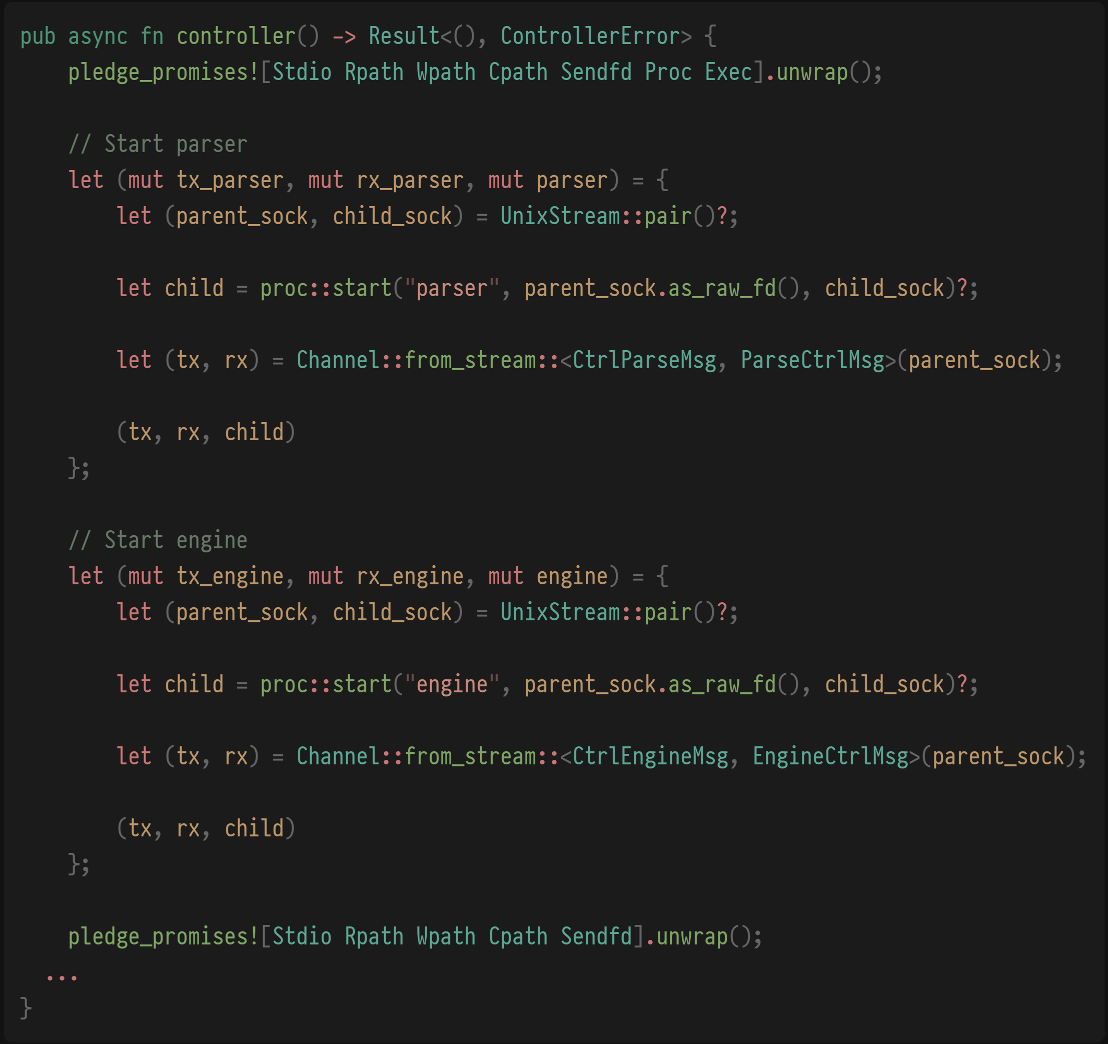
- UnixStream::pair()
- `enum` + `bincode` crate for (de)serilization over unix
- sockets bootstrapped with:
  - File descriptor (FD) inheritence (parent <-> child)
  - FD passing over existing socket (child <-> child)

file descriptors over sockets

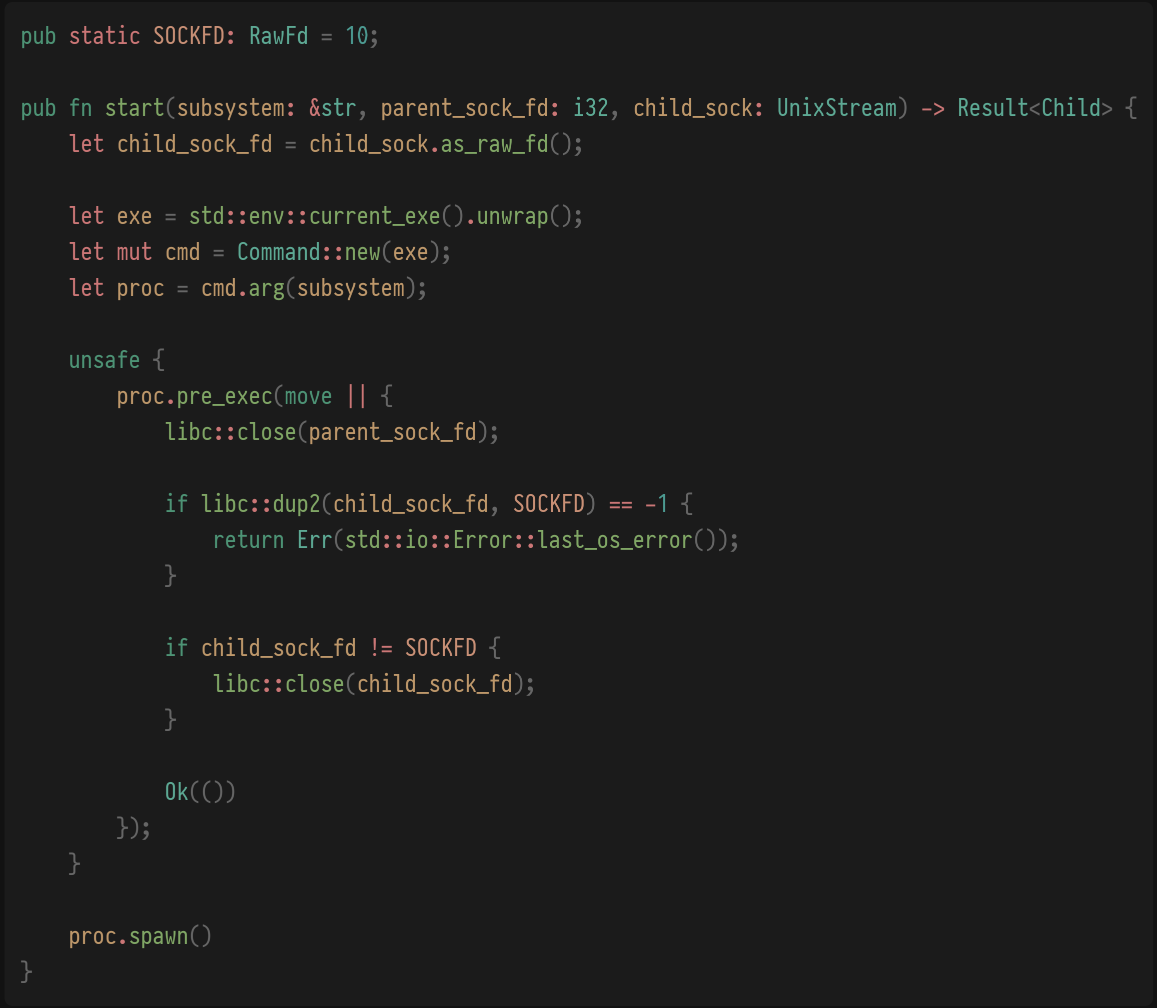
- Unix domain sockets can transfer FDs between processes
  - `sendmsg` / `recvmsg`
- not just putting file descriptor into byte stream…that wouldn’t work!
- `sendfd` crate simplifies the interface for us
  - …after a quick PR https://github.com/standard-ai/sendfd/pull/19!
- For bootstrapping channels, yes, but helps privsep generally



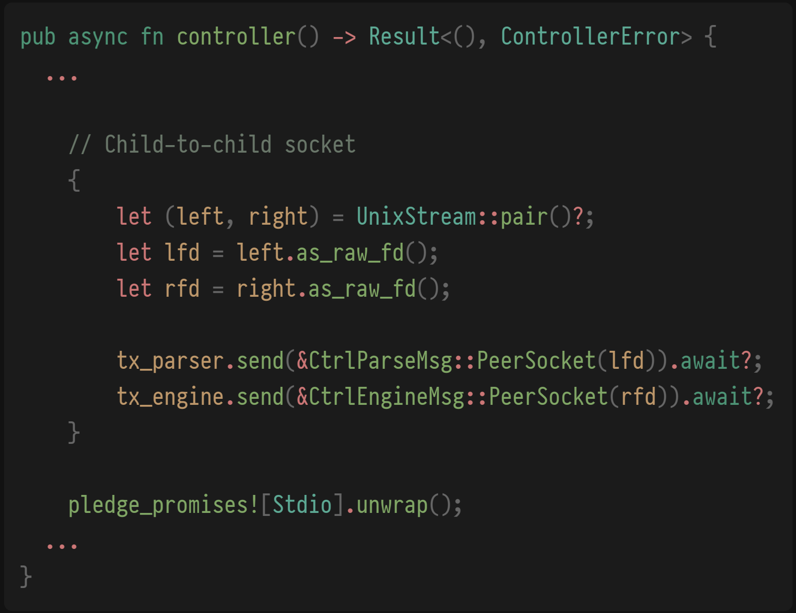
```rust
#[tokio::main]
async fn main() -> Result<(), ServiceError> {
    let cli = Cli::parse();

    match cli.subsystem {
        None => controller::controller().await?,
        Some(Subsystem::Parser) => parser::parser().await?,
        Some(Subsystem::Engine) => engine::engine().await?,
    }

    Ok(())
}
```



```rust
pub async fn controller() -> Result<(), ControllerError> {
    pledge_promises![Stdio Rpath Wpath Cpath Sendfd Proc Exec].unwrap();

    // Start parser
    let (mut tx_parser, mut rx_parser, mut parser) = {
        let (parent_sock, child_sock) = UnixStream::pair()?;

        let child = proc::start("parser", parent_sock.as_raw_fd(), child_sock)?;

        let (tx, rx) = Channel::from_stream::<CtrlParseMsg, ParseCtrlMsg>(parent_sock);

        (tx, rx, child)
    };

    // Start engine
    let (mut tx_engine, mut rx_engine, mut engine) = {
        let (parent_sock, child_sock) = UnixStream::pair()?;

        let child = proc::start("engine", parent_sock.as_raw_fd(), child_sock)?;

        let (tx, rx) = Channel::from_stream::<CtrlEngineMsg, EngineCtrlMsg>(parent_sock);

        (tx, rx, child)
    };

    pledge_promises![Stdio Rpath Wpath Cpath Sendfd].unwrap();
  ...
}
```



```rust
pub static SOCKFD: RawFd = 10;

pub fn start(subsystem: &str, parent_sock_fd: i32, child_sock: UnixStream) -> Result<Child> {
    let child_sock_fd = child_sock.as_raw_fd();

    let exe = std::env::current_exe().unwrap();
    let mut cmd = Command::new(exe);
    let proc = cmd.arg(subsystem);

    unsafe {
        proc.pre_exec(move || {
            libc::close(parent_sock_fd);

            if libc::dup2(child_sock_fd, SOCKFD) == -1 {
                return Err(std::io::Error::last_os_error());
            }

            if child_sock_fd != SOCKFD {
                libc::close(child_sock_fd);
            }

            Ok(())
        });
    }

    proc.spawn()
}
```



```rust
pub async fn controller() -> Result<(), ControllerError> {
  ...

    // Child-to-child socket
    {
        let (left, right) = UnixStream::pair()?;
        let lfd = left.as_raw_fd();
        let rfd = right.as_raw_fd();

        tx_parser.send(&CtrlParseMsg::PeerSocket(lfd)).await?;
        tx_engine.send(&CtrlEngineMsg::PeerSocket(rfd)).await?;
    }

    pledge_promises![Stdio].unwrap();
  ...
}
```

```rust
pub async fn controller() -> Result<(), ControllerError> {
  ...
    pledge_promises![Stdio].unwrap();

    loop {
        tokio::select! {
            msg = rx_parser.recv() => {
              ...
            }
            msg = rx_engine.recv() => {
              ...
            }
            _ = parser.wait() => {
              ...
            }
            _ = engine.wait() => {
              ...
            }
        }
    }

    Ok(())
}
```

```
pub async fn parser() -> Result<(), ParserError> {
    pledge_promises![Stdio Recvfd].unwrap();

    let (mut tx_ctrl, mut rx_ctrl) = Channel::new_from_fd::<ParseCtrlMsg, CtrlParseMsg>(SOCKFD)?;
    let (mut tx_engine, mut rx_engine) = expect_peer_channel(pid, &mut rx_ctrl).await?;

    loop {
        tokio::select! {
            msg = rx_engine.recv() => {
                ...
            }
            msg = rx_ctrl.recv() => {
                match msg? {
                    CtrlParseMsg::Data(data) => {
                        match parse_evaluate_rpn(&data)  {
                            Ok(value) => tx_engine.send(&ParseEngineMsg::NewValue(value)).await?,
                            Err(e) => println!("{NAME}[{pid}]: Bad input: {e:?}"),
                        }
                    },
                    _ => println!("{NAME}[{pid}]: unexpected message"),
                }
            }
        }
    }
```

Do we even need to do this if we’re using rust?

Oops our parser crate is backdoored!

```
echo "SHIBBOLETH cat /root/.ssh/id_ed25519 > pwned.key" | nc -N localhost 8080
```

Pledge to the rescue!

```
     Running `target/debug/privsep-ex1`
controller[47277]: Starting...
controller[47277]: Waiting...
parser[82802]: Starting...
Server listening on 127.0.0.1:8080
parser[82802]: Waiting on peer channel...
parser[82802]: received peer channel fd = 11
engine[80533]: Starting...
engine[80533]: received peer channel fd = 10
Accepted 127.0.0.1:41913
privsep-ex1[82802]: pledge "proc", syscall 2
controller[47277]: Received from parser Err(ConnectionClosedPrematurely)
Error: Engine(Channel(ConnectionClosedPrematurely))
Error: Controller(Channel(ConnectionClosedPrematurely))
```

Conclusions

This is just the principle of least privilege

...so understand the primitives provided by your operating system

Privsep reduces the blast radius of exploitable bugs

...but we may have to restructure our program in a big way

I can’t even stop these people! Once they applied pledge(2) and they know their program’s not perfect, they want to go and restructure it!

– Theo de Raadt

RUN
BSD

runbsd.info

Thank you!

"Cool, but we don’t use OpenBSD"

Sandboxing in other OSes

- Linux seccomp-bpf
    - https://justine.lol/pledge/
- FreeBSD capsicum
- NetBSD secmodel_sandbox (seems experimental)

…but OpenBSD’s approach is particularly ergonomic

Examples

dhcpleased

```
  PID TT  STAT        TIME COMMAND
15709 ??  IU       0:00.31 - /sbin/dhcpleased
33347 ??  Ip       0:00.29 |-- dhcpleased: engine (dhcpleased)
84909 ??  IpU      0:00.42 `-- dhcpleased: frontend (dhcpleased)
```

OpenBSD annotates subsystem with setproctitle(3)

smtpd

```
  PID TT  STAT        TIME COMMAND
41810 ??  Ip       0:00.52 - /usr/sbin/smtpd
11124 ??  Ipc      0:00.60 |-- smtpd: crypto (smtpd)
74895 ??  Ipc      0:00.30 |-- smtpd: control (smtpd)
50559 ??  Ip       0:00.31 |-- smtpd: lookup (smtpd)
27218 ??  Ipc      0:00.85 |-- smtpd: dispatcher (smtpd)
13188 ??  Ipc      0:00.80 |-- smtpd: queue (smtpd)
36552 ??  Ipc      0:00.51 `-- smtpd: scheduler (smtpd)
```

unveil

```c
int unveil(const char *path, const char *permissions);
```

```c
unveil("foo.txt", "rw");
unveil("bar.txt", "rwx");
```

```c
unveil(NULL, NULL);
```