MirageOS - developing operating systems in OCaml

Combining systems programming with functional programming

Hannes Mehnert, https://hannes.robur.coop

Fun OCaml, 16th September 2024, Berlin

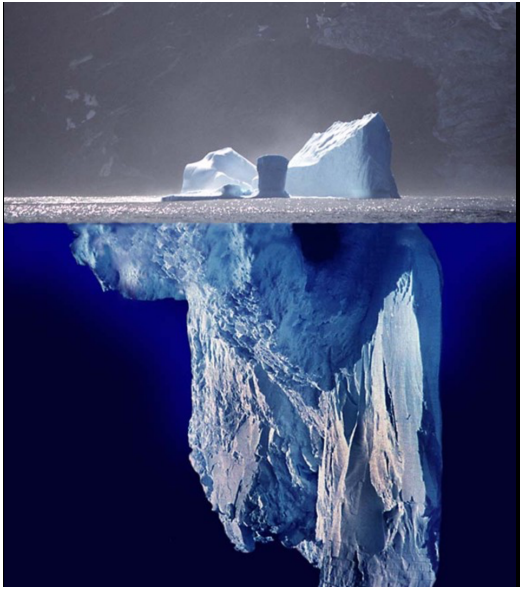
Contents

What is an operating system?

- Communication with hardware, providing abstraction to software
- Provide an interface for the user (graphical, terminal, remote)
- Allows to execute processes: browser, editor, HTTP server
- Includes user management, file system, resource management
- Access to the Internet (network device, TCP/IP stack, ...)

Conventional OS

- Several OS exist: Windows, macOS, Linux, BSDs
- All written in unsafe programming language C
- Programming mistakes often lead to security issues that are exploitable
- All are general purpose: run many applications
- Enormous legacy code basis



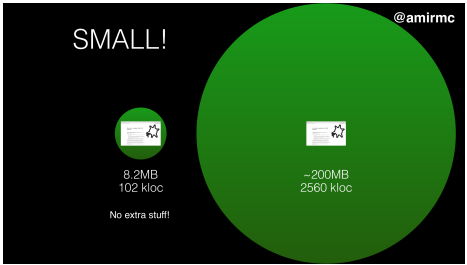
Code **you** care about

Code **the OS** insists you need

- Started in 2013 at University of Cambridge (3 developers from XenSource),
- More than 150 opam packages,
- Used in Qubes OS, Docker Desktop (Windows & macOS), Nitrokey NetHSM,
- 12 hack retreats happened, 20 - 30 participants, 1 week in Marrakesh,
- Today at least 6 people are working full time on MirageOS

Goals of MirageOS

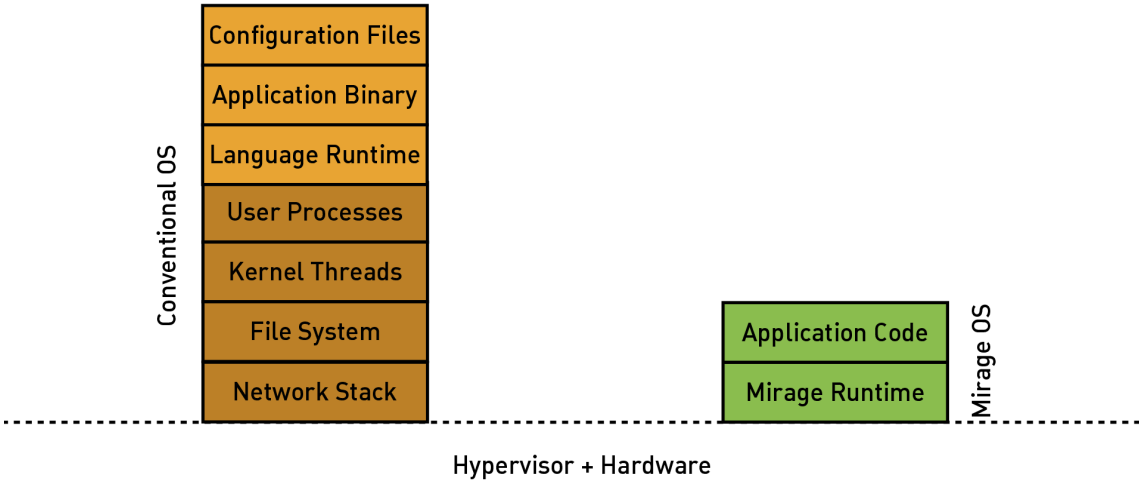
- Radical approach to operating system development,
- Security from the grounds up (25x - 100x less code): fewer attack vectors (memory safety), less attack surface,
- Reducing configuration and run-time complexity,
- Drastically reduced carbon footprint (10x less CPU cycles, 25x less memory).



Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.

Antoine de Saint-Exupery (1900 - 1944)

MirageOS stack



- MirageOS is completely developed in OCaml
- Uses a value-passing OCaml style ("sans IO", explicity IO)
- MirageOS packages are usable with Unix (Lwt/miou/async/eio): tls
- Lwt event loop is used

What is a MirageOS unikernel?

- Statically compiled ELF binary
- Executed as a virtual machine
- Solo5 is the host system process ("tender")
- MirageOS unikernel executes in KVM (VMM/BHyve), Xen, Virtio, muen, Linux seccomp, and unix
- Network and block device are explicitly given (no implicit access)

Hello world (unikernel.ml)

```
open Lwt.Infix
let hello =
  Cmdliner.Arg.(value & opt string "Hello World!"
                  (info ~doc:"How to say hello." [ "hello" ]))

module Hello (Time : Mirage_time.S) = struct
  let start _time hello =
    let rec loop = function
      | 0 -> Lwt.return_unit
      | n ->
          Logs.info (fun f -> f "%s" hello);
          Time.sleep_ns (Duration.of_sec 1) >>= fun () ->
          loop (n - 1)
    in
    loop 4
end
```

Hello world (config.ml)

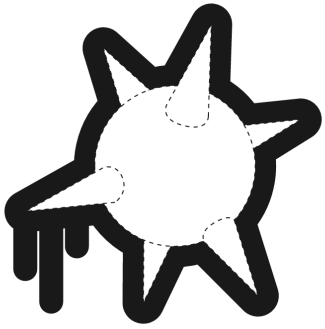
```
(* mirage >= 4.5.0 & < 4.8.0 *)
open Mirage

let runtime_args = [ runtime_arg ~pos:__POS__ "Unikernel.hello" ]
let packages = [ package "duration" ]
let main = main ~runtime_args ~packages "Unikernel.Hello" (time @-> job)
let () = register "hello-key" [ main $ default_time ]
```

Hello world (run it)

```
$ mirage configure -t hvt
$ make
$ solo5-hvt -- dist/hello-key.hvt --hello="FunOCaml"
```

You have reached the BTC Piñata.



BTC Piñata knows the private key to the bitcoin address **183XuXTTgmfYfKcHbJ4sZeF46a49Fnihdh**. If you break the Piñata, you get to keep what's inside.

Here are the rules of the game:

- You can connect to port 10000 using TLS. Piñata will send the key and hang up.
- You can connect to port 10001 using TCP. Piñata will immediately close the connection and connect back over TLS to port 40001 on the initiating host, send the key, and hang up.
- You can connect to port 10002 using TCP. Piñata will initiate a TLS handshake over that channel serving as a client, send the key over TLS, and hang up.

And here's the kicker: in both the client and server roles, Piñata requires the other end to present a certificate. Authentication is performed using standard **path validation** with a single certificate as the trust anchor. And no, you can't have the certificate key.

It follows that it should be impossible to successfully establish a TLS connection as long as Piñata is working properly. To get the spoils, you have to smash it.

Before you ask: yes, Piñata will talk to itself and you can enjoy watching it do so.

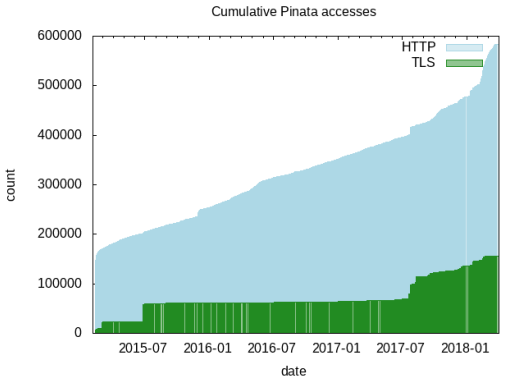
BTC Piñata is a **MirageOS** unikernel using **not quite so broken** software. It is written in OCaml, runs directly on Xen, and is using native OCaml **TLS** and **X.509** implementations.

The **full list** of installed software and a **toy unikernel** without secrets are available. There is no need to use the old automated tools on Piñata - roll your own instead. This challenge runs until the above address no longer contains the 10 bitcoins it started with, or until we lose interest.

Why are we doing this? At the beginning of 2014 we started to develop a **not quite so broken** TLS implementation from scratch. You can read more about it on **https://nqsb.io** or watch our **31c3 talk** about it. Now, we want to boost our confidence in the TLS implementation we've developed and show that robust systems software can be written in a functional language. We recapitulated the **first five months of the Piñata**.

We are well aware that **bounties** can only disprove the security of a system, and never prove it. We won't take home the message that we are 'unbreakable', 'correct', and especially not

- Marketing of our from-scratch TLS implementation
- Transparent and self-serving security bait
- Web server which contains a private key for a Bitcoin wallet
- If a peer authenticates (using TLS and client certificates), it sends the private key
- Online since February 2015 with 10 BTC until March 2018
- The Piñata was not hacked, the BTC were only borrowed and reused in other projects



Size of Bitcoin Piñata unikernel vs openssl on Linux

@amirmc

SMALL!

8.2MB
102 kloc

No extra stuff!

~200MB
2560 kloc

- QubesOS is a "reasonably secure operating system"
- Uses Xen for isolation of workspaces and applications (i.e. pdf reader)
- Qubes-Mirage-firewall is a small replacement for the Linux-based firewall in OCaml
- Instead of 300MB, only consumes 32MB resident memory
- `https://github.com/mirage/qubes-mirage-firewall`

- https://github.com/mirage
- https://github.com/robur-coop

Rome ne s'est pas faite en un jour (Rome wasn't built in a day)

Li Proverbe au Vilain, around 1190

- Robur collective
- Tarides & Parsimoni (Irmin, SpaceOS)
- Freelance contributors and contractors
- Open source developers

What is robur?

- We run our code (name service, website, ...) as unikernels
- A collective of OCaml programmers working towards getting MirageOS deployed (at the moment we're four people)
- Self-organized - working without hierarchies
- Funded via public money (EU NGI (e.g. NLnet), Prototypefund (Germany)), commercial contracts (Semgrep, Tarides, OCaml Labs), donations
- Needs-based payments
- Gift economy
- Part of a non-profit company based in Germany
- `https://robur.coop`

- Just finished MirageVPN (an OpenVPN compatible implementation)
  `https://blog.robur.coop`
- DNSVizor - a DNS and DHCP replacement (a la dnsmasq)
  `https://github.com/robur-coop/dnsvizor`
- PTT, developing mailing lists and web archive
- MirageOS Taler Exchange (MTE)
- A lot on performance recently (see mirage-crypto and tls release notes)
- Improving developer experience: removing more boilerplate and functors
- OCaml 5: effects (and multicore), with unikraft integration

Call to action

- Join our 2025 retreat `https://retreat.mirage.io`
- Find documentation and community on `https://mirage.io`
- Tomorrow 15:00 in Apollo room: hands-on MirageOS
- Get in touch team@robur.coop

More MirageOS unikernels

- Unipi (web server) content in git `https://github.com/robur-coop/unipi`
- Opam mirror `https://git.robur.coop/robur/opam-mirror`
- SMTP (eMail) stack `https://github.com/mirage/ptt/`
- `ssh-agent` for Qubes `https://github.com/reynir/qubes-mirage-ssh-agent`
- Web site: `https://mirage.io` (`https://github.com/mirage/mirage-www`)
- DHCP server `https://github.com/mirage/charrua`
- OpenVPN client and server `https://github.com/robur-coop/miragevpn`
- Pastebin clone `https://github.com/dinosaure/pasteur`
- Pong game `https://github.com/cfcs/PongOS`
- Z machine (Zork) via telnet `https://github.com/mato/flathead`

Selected related talks

- At radical networks 2019 about QubesOS firewall by Stefanie Schirmer
  `https://livestream.com/internetsociety/radnets19/videos/197991963`
- At FOSDEM 2019 about Solo5 by Martin Lucina
  `https://fosdem.org/2019/schedule/event/solo5_unikernels/`
- At Lambda World 2018 by Romain Calascibetta
  `https://www.youtube.com/watch?v=urG5BjvjW18`
- At 36th Chaos Communication Congress by Hannes Mehnert
  `https://media.ccc.de/v/36c3-11172-leaving_legacy_behind`

- Goal: compile the source multiple times should produce bit-wise identical output
- Temporary files names, timestamps, etc. may cause issues
- Our MirageOS unikernels are reproducible!
- And we have tooling to check reproducibility
- `https://hannes.robur.coop/Posts/ReproducibleOPAM`
- `https://builds.robur.coop`