MirageOS: minimizing the attack surface and vectors of network services

Hannes Mehnert, robur.io, hannesm@mastodon.social

BSDCan, 18th May 2019, Ottawa

About me

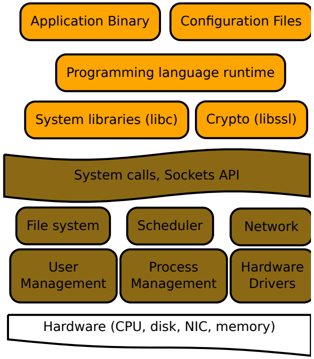
- Hacker interested in communication infrastructure, network and security protocols
- Using FreeBSD since 4.6
- Academic research on formal verification, programming language semantics, security
- PhD (2013, ITU Copenhagen) incremental verification of the correctness of object-oriented software using Coq and higher-order separation logic
- PostDoc (2014-2017) at University of Cambridge: MirageOS and formal model of TCP/IP and the Unix Sockets API in HOL4
- 2018 started a non-profit robur.io in Berlin with the goal to deploy MirageOS
- Running my own communication infrastructure (eMail, DNS, ..) since ~2000

Motivation

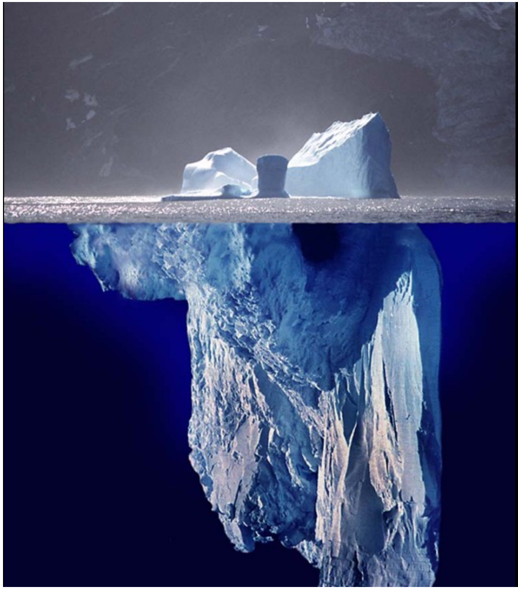
Memory corruption issues are the root-cause of 68% of listed CVEs.

Ben Hawkes analysed 108 CVE since start of Google's Project Zero in 2014, `0day "In the Wild"` `https://googleprojectzero.blogspot.com/p/0day.html` (published 2019-05-15)

Stack



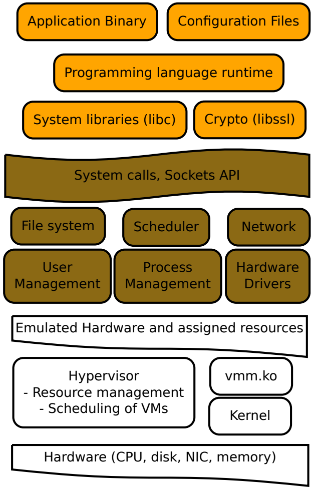
- Common Unix applications depend on shared libraries, their configuration files.
- Reproducing one application on a separate computer requires all these artifacts.
- Kernel isolates processes from each other.
- Compromise is contained to a single process.
- Escalation by flaws in the system call API (568 in `sys/syscall.h`)



Code **you** care about

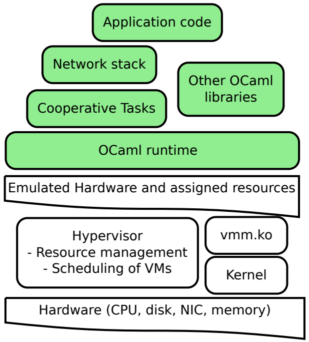
Code **the OS** insists you need

Hypervisor stack



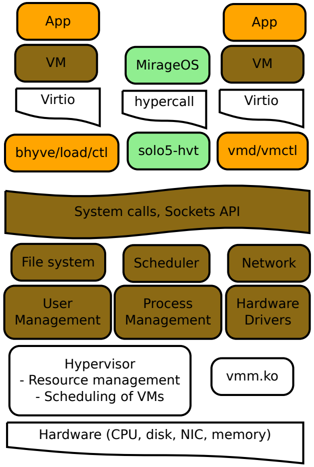
- The hypervisor manages the physical machine and assigns resources to virtual machines. Hardware is emulated.
- Scheduling at every layer
- Hypervisor isolates virtual machines from each other

MirageOS unikernel - library operating system



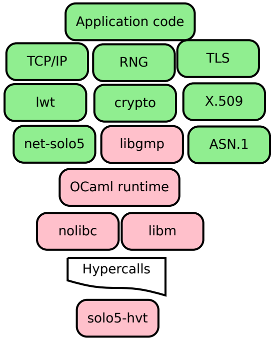
- Each MirageOS unikernel is a separate virtual machine
- Each is tailored for its single service at compilation time
- Using OCaml libraries
- Single address space
- Single core
- Programming language guarantees memory safety
- Cooperative tasks (no interrupts)

MirageOS Hypervisor integration



- Custom `solo5-hvt` monitor process
- Less than 3000 lines of code
- Sets up memory, loads statically linked ELF binary, sets up VCPU
- Boot: `jmp 0x100000` (passing command line arguments)
- Hypercalls: block device (read, write, info), network device (read, write, info)
- Console write, wall clock, monotonic clock, abort, exit, yield

Virtual Machine in detail



- `nolibc`: `dtoa`, `dlmalloc`, `strcmp`, `strlen`: ~8000 lines C (6200 malloc)
- `libm`: openlibm (FreeBSD libm repackaged by Julialang) ~22000 lines C
- OCaml runtime: ~25000 lines C

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)

OCaml

- Multi-paradigm memory-managed programming language
- Expressive type system with type inference
- Strong type system, no downcasts or pointer arithmetics
- Unique module system
- Compiles to native machine code, types are erased during compilation
- Used by industry and academia for proof assistants, compilers
- Developed since more than 25 years at INRIA

MirageOS OCaml style

- No objects
- Immutable data as much as sensible
- Value passing style: state and data in, state and reply out
- Errors are explicitly declared in the API, no exceptions
- Concurrent programming with promises using the `lwt` library
- Declarative code - easy to understand and reason about
- Expressing invariants (e.g. read-only buffer) in the type system

Example: MirageOS CLOCK interface

```
module type PCLOCK = sig
  val now_d_ps : unit -> int * int64
  (** [now_d_ps ()] is [(d, ps)] representing the POSIX time occuring
      at [d] * 86'400e12 + [ps] POSIX picoseconds from the epoch
      1970-01-01 00:00:00 UTC. [ps] is in the range
      [0;86_399_999_999_999_999L]. *)
end
```

- Implementation depends on target
- On Unix `gettimeofday` is used
- On hypervisors the CPU instruction `RDTSC` is used

Example: Interfaces and implementation

- As MirageOS unikernel developer, you program against the interface!
- Write once, run anywhere ;)

```
module Main (T : TIME) (P : PCLOCK) = struct
 let start _time _pclock =
  let rec speak () =
   let now = Ptime.v (P.now_d_ps ()) in
   let pp_ts = Ptime.pp_rfc3339 () in
   Format.printf "It is %a\n%!" pp_ts now;
   T.sleep_ns (Duration.of_sec 1) >>= fun () ->
   speak ()
  in
  speak ()
end
```

Example: Running a unikernel

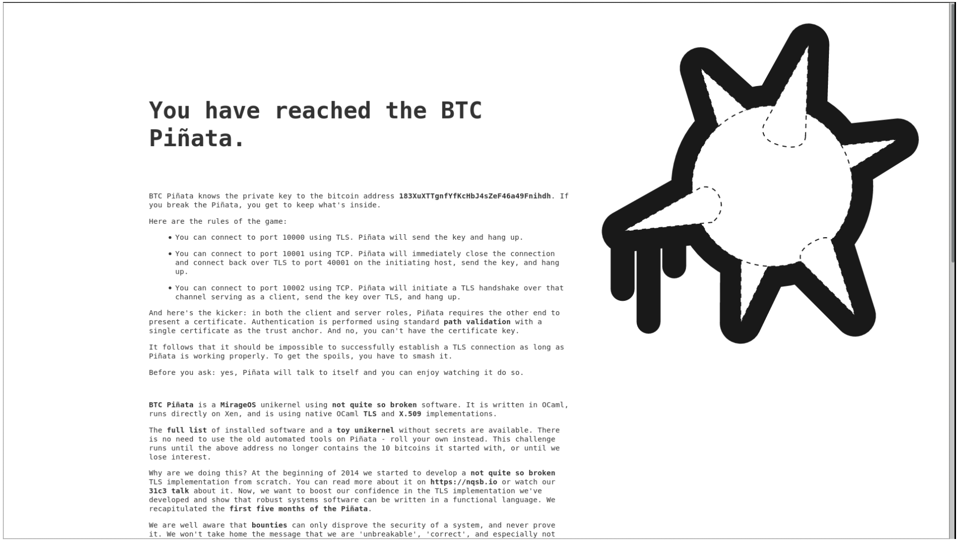
```
$ mirage configure #configures (defaults to Unix target)
$ make depend #installs necessary dependencies
$ mirage build #compiles the unikernel, producing an ELF executable
$ ./speaking_clock #executes it

$ mirage configure -t hvt #configures for Hardware Virtualized Tender
$ make depend #installs necessary dependencies
$ mirage build #compiles, output: monitor (solo5-hvt) and VM image (.hvt)
$ ./solo5-hvt speaking_clock.hvt  #executes it
```

Supported MirageOS targets

- Unix binary (development and testing)
- Architectures: x86-64, arm64, ESP32, soon RISC-V
- Hypervisor: Xen, KVM, FreeBSD BHyve, OpenBSD VMM, Virtio
- Muen separation kernel (Ada/SPARK)
- Genode operating system framework (Nova microkernel, L4)
- Linux binary with strict seccomp filters

Unikernel: Bitcoin Piñata



You have reached the BTC Piñata.

BTC Piñata knows the private key to the bitcoin address **183XuXTTgnfYfKcHbJ4sZeF46a49Fnihdh**. 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.

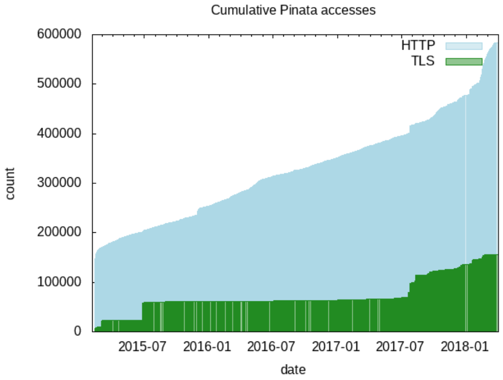
The **full list** of installed software and a **toy unikernel** without secrets are available. There is no need to use the old automated tools on Piñata - roll your own instead. This challenge runs until the above address no longer contains the 10 bitcoins it started with, or until we lose interest.

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**.

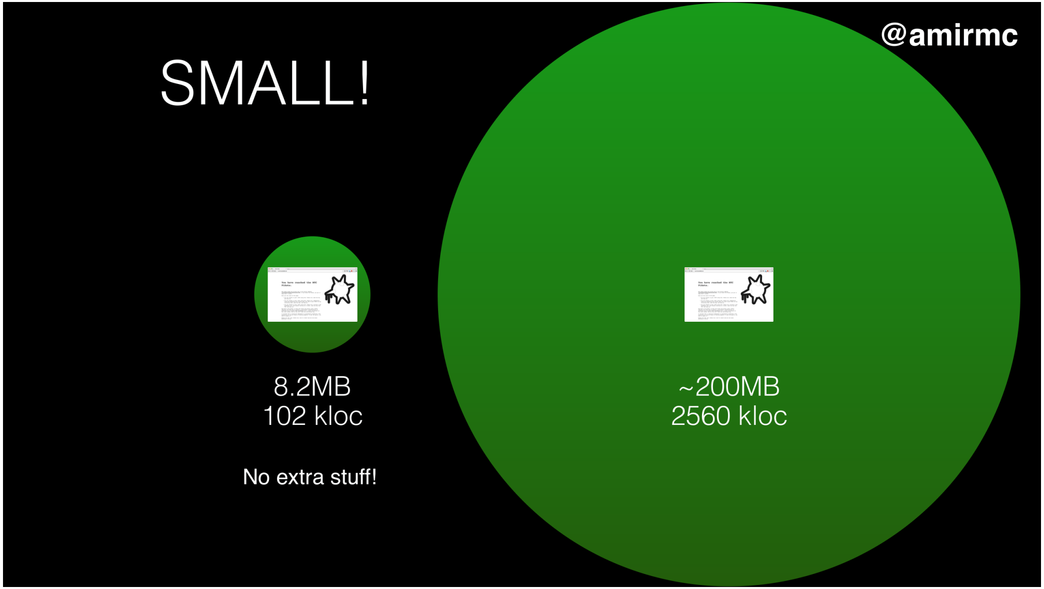
We are well aware that **bounties** can only disprove the security of a system, and never prove it. We won't take home the message that we are 'unbreakable', 'correct', and especially not

Unikernel: Bitcoin Piñata

- 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
- Contained 10 BTC until March 2018



Trusted computing base - Bitcoin Piñata



@amirmc

SMALL!

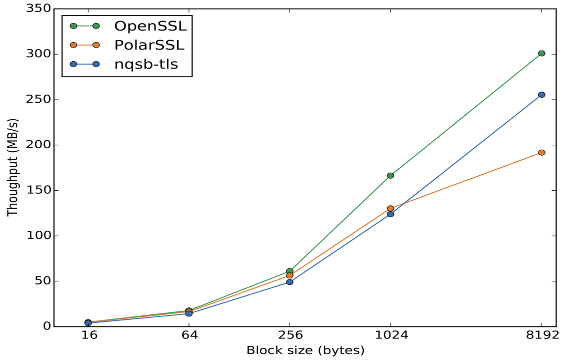
8.2MB
102 kloc

No extra stuff!

~200MB
2560 kloc

Performance of nqsb-TLS (2015, on a laptop)

- Throughput



- Handshakes (number per second)

	nqsb	OpenSSL	Polar
RSA	698	723	672
DHE-RSA	601	515	367

MirageOS libraries

http://

DNS
DOMAIN NAME SERVICES

DHCP

TCP/IP

BGP

TLS

git

OpenPGP

Let's Encrypt

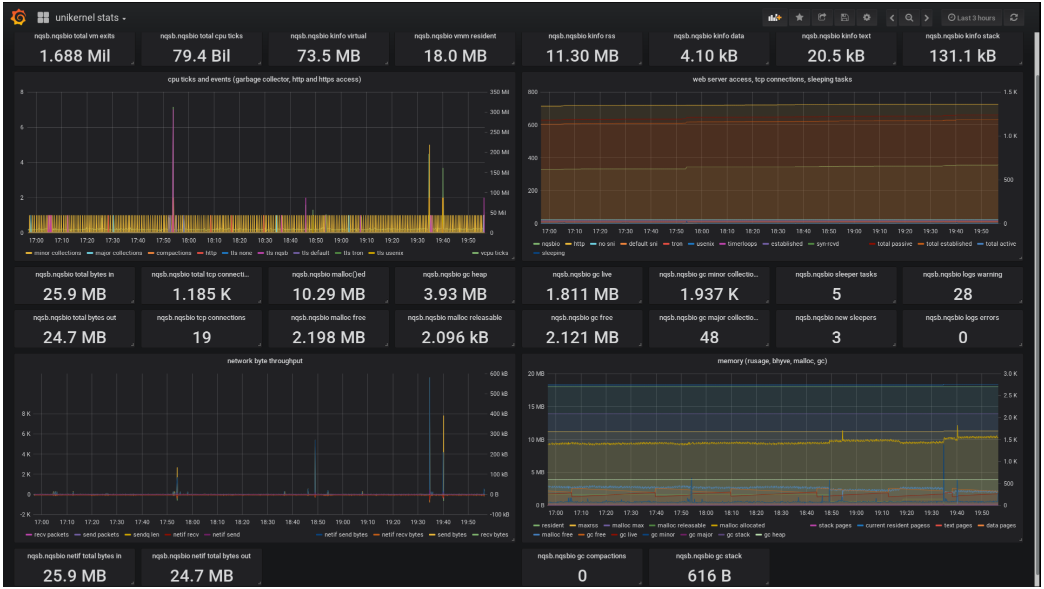
Prometheus

>_
SSH

syslog

SNMP

Monitoring



unikernel stats

Last 3 hours

nqsb.nqsbio total vm exits	nqsb.nqsbio total cpu ticks	nqsb.nqsbio kinfo virtual	nqsb.nqsbio vmm resident
1.688 Mil	79.4 Bil	73.5 MB	18.0 MB
nqsb.nqsbio kinfo rss	nqsb.nqsbio kinfo data	nqsb.nqsbio kinfo text	nqsb.nqsbio kinfo stack
11.30 MB	4.10 kB	20.5 kB	131.1 kB

nqsb.nqsbio total bytes in	nqsb.nqsbio total tcp connecti...	nqsb.nqsbio malloc()ed	nqsb.nqsbio gc heap
25.9 MB	1.185 K	10.29 MB	3.93 MB
nqsb.nqsbio gc live	nqsb.nqsbio gc minor collectio...	nqsb.nqsbio sleeper tasks	nqsb.nqsbio logs warning
1.811 MB	1.937 K	5	28
nqsb.nqsbio total bytes out	nqsb.nqsbio tcp connections	nqsb.nqsbio malloc free	nqsb.nqsbio malloc releasable
24.7 MB	19	2.198 MB	2.096 kB
nqsb.nqsbio gc free	nqsb.nqsbio gc major collectio...	nqsb.nqsbio new sleepers	nqsb.nqsbio logs errors
2.121 MB	48	3	0

nqsb.nqsbio netif total bytes in	nqsb.nqsbio netif total bytes out	nqsb.nqsbio gc compactions	nqsb.nqsbio gc stack
25.9 MB	24.7 MB	0	616 B

Unikernel: DNS server

- Replies to DNS requests on port 53
- Zone data as a trie is kept in memory
- Storage in a git remote in zone file format
- Configuration (ip address, git remote, syslog, ..) via boot arguments
- No block device

Unikernel: DNS server

- Replies to DNS requests on port 53
- Zone data as a trie is kept in memory
- Storage in a git remote in zone file format
- Configuration (ip address, git remote, syslog, ..) via boot arguments
- No block device

- Git pre-commit hook validates changes: SOA was increased, . . .
- Git post-commit hook notifies (RFC 1996)

Unikernel: DNS server

- Replies to DNS requests on port 53
- Zone data as a trie is kept in memory
- Storage in a git remote in zone file format
- Configuration (ip address, git remote, syslog, ..) via boot arguments
- No block device

- Git pre-commit hook validates changes: SOA was increased, . . .
- Git post-commit hook notifies (RFC 1996)

- TSIG support HMAC-authentication - RFC 2845
- Modification also via (authenticated) nsupdate (RFC 2136)
- Successful nsupdate implies a `git push` to the repository

Unikernel: DNS server

- Replies to DNS requests on port 53
- Zone data as a trie is kept in memory
- Storage in a git remote in zone file format
- Configuration (ip address, git remote, syslog, ..) via boot arguments
- No block device

- Git pre-commit hook validates changes: SOA was increased, ...
- Git post-commit hook notifies (RFC 1996)

- TSIG support HMAC-authentication - RFC 2845
- Modification also via (authenticated) nsupdate (RFC 2136)
- Successful nsupdate implies a `git push` to the repository

- Secondaries are notified on update, and start zone transfer (AXFR/IXFR)
- VM image size roughly 8MB (including IP stack, DNS, git)

Existing MirageOS unikernels

- Various web sites including mirage.io and nqsb.io
- Content management system `Canopy` that serves Markdown files from a git remote
- OpenVPN client (work in progress)
- QubesOS firewall (run-time rule changes under development)
- CalDAV server (test deployment since November 2018)
- DNS server (primary, secondary, let's encrypt certificate issuance)
- DNS resolver (work in progress)
- DHCP server
- SMTP server (work in progress)
- Albatross orchestration system, deploying unikernels via TLS handshake (client certs)

Security

- Minimized attack surface
- Avoiding common attack vectors, such as memory corruptions
- Defense in depth (in progress): W ^ X mapping (Linux only), stack guard, ASLR
- Formal verification on the horizon
- Supply chain: signed library releases, reproducible builds

Signed library releases (wip)

- Library authors sign their releases
- A quorum of repository maintainers delegates packages to authors
- Impact of a private key compromise or loss is contained to their packages
- If a quorum of maintainers is compromised, game over
- Rollback, mix-and-match attacks mitigated by snapshot service
- Freeze, slow retrieval attacks mitigated by timestamp service
- Using `update framework` (Cappos NYU) with augmentation proposal TAP8

Reproducible builds

- Simple idea: compiling the source should produce identical output
- Temporary files names, timestamps, build path are problematic
- Environment (C compiler, libc, ..) needs to be specified as input
- The OCaml compiler and runtime are reproducible, MirageOS unikernels mostly
- reproducible-builds.org

Call for help: how to destroy a BHyve VM?

- Currently we use `sysctl "hw.vmm.destroy"` in `atexit`
- Requires root privileges :|

Call for help: how to protect memory (W ^ X)

- We `mmap` memory in the host system process
- And `mprotect` to set the protection bits
- The guest mapping is not changed by this (still writable and executable)
- `https://github.com/Solo5/solo5/issues/303`
- We are keen to test any solution (different syscall, ..)

Testing and formal verification

- Unit testing, quickcheck, fuzz-based testing
- Model checking with TLA+ was used for libraries
- Interactive proof assistant Coq generates OCaml code
- Dependently typed Agda generates OCaml code
- CFML is a proof system specifically for OCaml

Community

- Research at University of Cambridge since 2008 (ongoing student projects, etc.)
- Bi-annual hack retreats since 2016 (7 days, ~35 participants, next September 23rd - 28th)
- Dogfooding our unikernels (DHCP, DNS)
- Open source contributors worldwide
- Docker for Mac and Docker for Windows use MirageOS libraries
- Running 15 unikernels on my servers since a year (DNS, Webserver, CalDAV)

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

Li Proverbe au Vilain, around 1190

Conclusion

- Radical approach to operating system development
- Security from the ground up
- Reduced complexity
- Reasonable performance
- Boots in milliseconds
- Permissively licensed (BSD/MIT)
- More information at mirage.io
- We at robur.io provide commercial MirageOS development as non-profit company
- Contact me: EMail hannes@mehnert.org
- Mastodon: hannesm@mastodon.social

Selected related talks

- At FOSDEM 2019 about Solo5 by Martin Lucina
  `https://fosdem.org/2019/schedule/event/solo5_unikernels/`
- At 34C3 (2017) by Mindy Preston
  `https://media.ccc.de/v/34c3-8949-library_operating_systems`
- At Lambda World 2018 by Romain Calascibetta
  `https://www.youtube.com/watch?v=urG5BjvjW18`
- At Esper (2015) by Anil Madhavapeddy
  `https://www.youtube.com/watch?v=bC7rTUEZfmI`