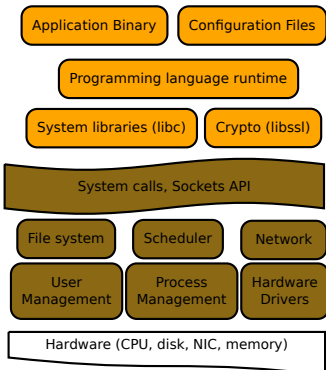
Leaving legacy behind

Reducing carbon footprint of network services with MirageOS unikernels

Hannes Mehnert, https://hannes.nqsb.io

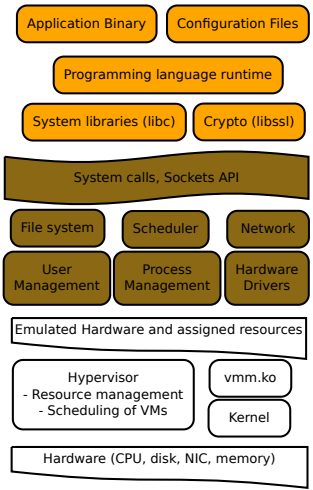
36c3, 27th December 2019, Leipzig

Stack

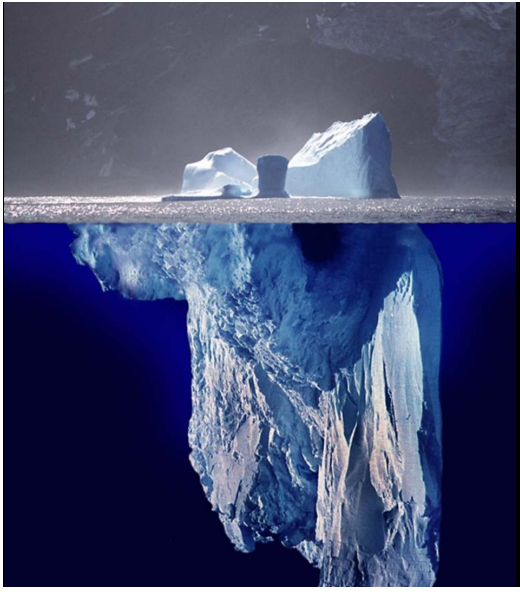


- Unix applications depend on libraries and configuration files.
- Kernel isolates processes from each other using virtual memory.
- Compromise is contained to a single process.
- Privilege escalation by flaws in the system call API (568 in `sys/syscall.h`).

Hypervisor stack



- The hypervisor manages the resources, and splits them across virtual machines. Hardware is emulated.
- Scheduling is done by the hypervisor (VMs) and by each virtual machines (processes).
- The hypervisor isolates virtual machines from each other.



Code **you** care about

Code **the OS** insists you need

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

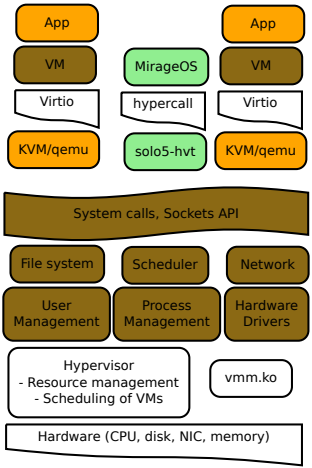
Ben Hawkes analysed 108 CVE since start of Google's Project Zero in 2014, `0day "In the Wild"` `https://googleprojectzero.blogspot.com/p/0day.html` (published 2019-05-15)

Goals

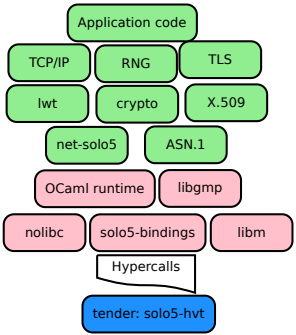
- Reducing attack vectors (memory safety),
- Reducing attack surface,
- Reducing run-time complexity,
- Reducing the carbon footprint.

MirageOS unikernel - library operating system

- Each service (i.e. DNS resolver, web server) is a separate MirageOS unikernel
- Functional programming language OCaml with automated memory management
- Only the libraries needed are compiled into the binary
- Libraries are developed independently and reused across unikernels
- Cooperative tasks, no interrupts
- Single address space, single core, single process
- No user management, no process management, no file system, no virtual memory
- Executed as virtual machine



- Custom `solo5-hvt` monitor process in the host system
- Sets up memory, loads unikernel image, sets up VCPU
- Boot: `jmp 0x100000`
- Hypercalls: main, yield, argv, clock, console, block device, network interface



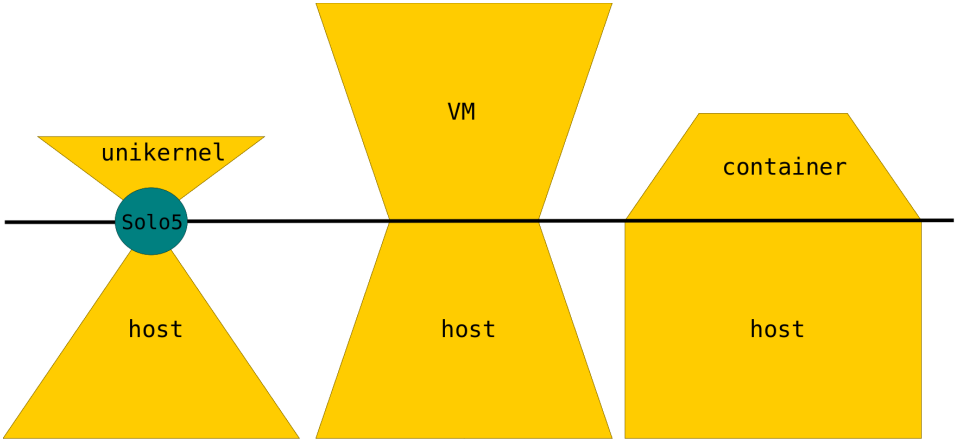
C code in a MirageOS unikernel

- OCaml runtime: ~25 kloc
- `nolibc`: `malloc`, `strcmp`, ...: ~8 kloc
- `solo5-bindings`: ~2 kloc
- `libm`: openlibm ~22 kloc

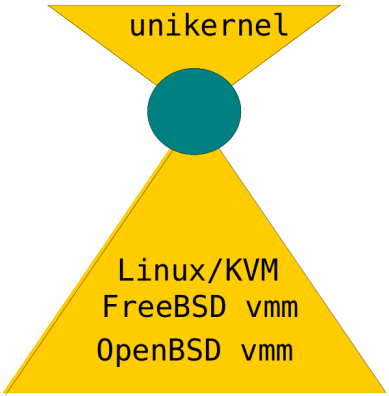
Solo5 - sandboxed execution environment for unikernels

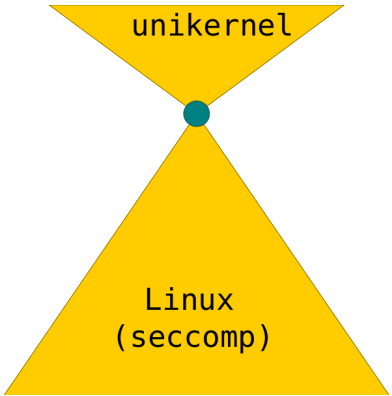
- Resources (memory, network interface, block devices) are allocated statically
- Minimalist hypercall interface (14 functions)
- Bindings for various targets (KVM, Genode, Virtio, seccomp)
- Sandboxed host system tender where applicable
- `https://github.com/solo5/solo5`

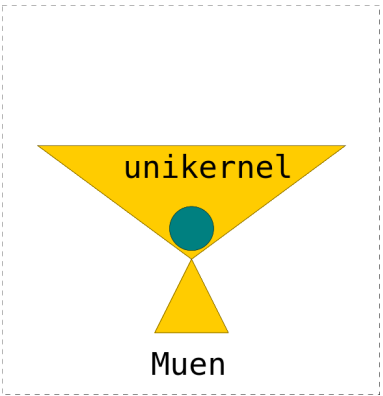
Solo5 - compared to virtual machine and container



Solo5 - hardware virtualised tender and sandboxed process tender







- Muen is a tiny separation kernel developed in Ada
- Using SPARK to guarantee memory isolation
- Static resource management (communication channels, memory, devices)

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

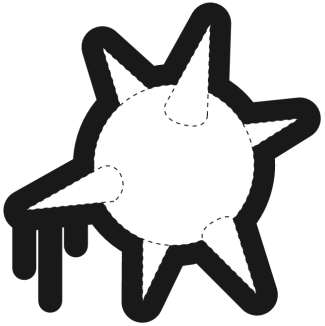
Antoine de Saint-Exupery (1900 - 1944)

- Multi-paradigm programming language initially released in 1996
- Declarative code is the goal
- Focus on the problem, do not distract with boilerplate
- Abstractions (variables, functions, higher order functions)
- Expressive static type system spots violations at build time
- Type inference allows concise code
- Types are erased during compilation
- Compiles to native machine code

- Each value is defined in a module (e.g. filename or explicit `module`)
- Each module has an interface, its signature
- Functors are compile-time functions from module to module, and allow parametrization, i.e. `Map.Make(String)`
- In MirageOS, each resource (network interface) has a signature, and target-specific (virtio, xen, spt) implementations

- Using immutable data whenever sensible
- Value passing style: state and data in, state and reply out
- Errors are explicitly declared in the API, and have to be handled by the caller
- Concurrent programming with promises
- Ability to express strong invariants (read-only buffer) in the type system

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.

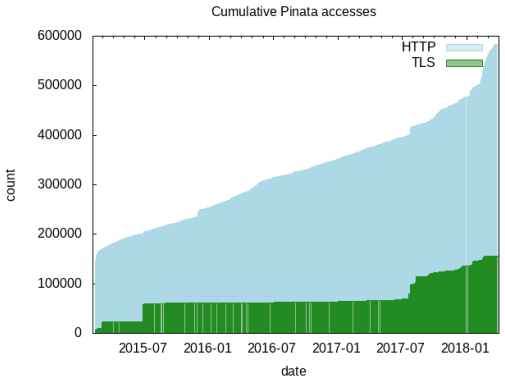
BTC Piñata is a **MirageOS** unikernel using **not quite so broken** software. It is written in OCaml, runs directly on Xen, and is using native OCaml **TLS** and **X.509** implementations.

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

- 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
- See `http://ownme.ipredator.se`



Size of Bitcoin Piñata unikernel vs openssl on Linux

@amirmc

SMALL!

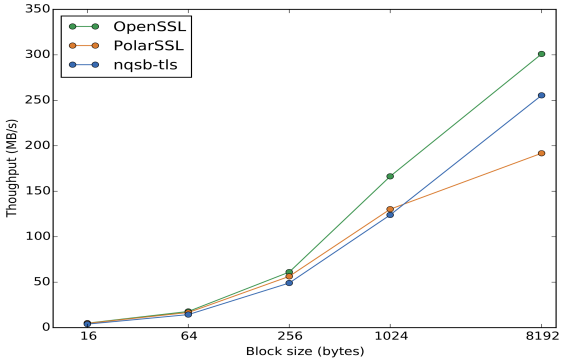
8.2MB
102 kloc

No extra stuff!

~200MB
2560 kloc

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

- Throughput



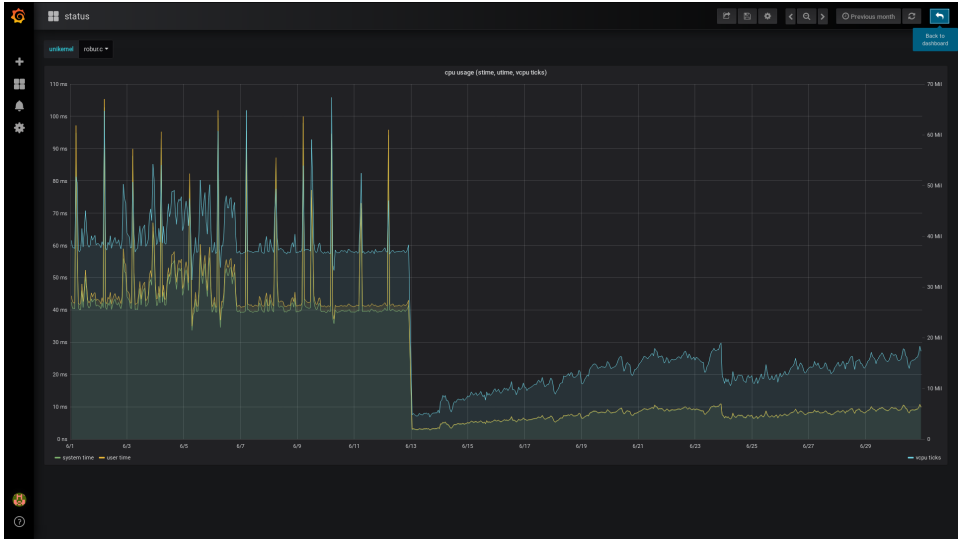
- Handshakes (number per second)

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

- Developed in 2018 with a grant from Prototypefund
- Interoperable with widely used clients (Android, Linux, macOS)
- Stores data in a remote git repository
- Image size ~10MB (HTTP server, WebDAV, CalDAV, ics, git, IP stack)
- CalDavZAP integration, a calendar in JavaScript
- Demo server at `https://calendar.robur.coop` (data repository `https://git.robur.io/?p=calendar-data.git`)
- Source `https://github.com/roburio/caldav`

Resource consumption

CPU consumption



Case study authoritative DNS server

- Domain Name System, used for translating hostnames into Internet addresses
- Authoritative server replies to DNS requests
- Data (zone) is kept in memory, no block device
- Storage in a git remote in zone file format
- Configuration (ip address, git remote, syslog, ..) via boot arguments

Case study authoritative DNS server

- Domain Name System, used for translating hostnames into Internet addresses
- Authoritative server replies to DNS requests
- Data (zone) is kept in memory, no block device
- Storage in a git remote in zone file format
- Configuration (ip address, git remote, syslog, ..) via boot arguments

- Modification via git commit and push, sending a notify (RFC 1996) to server
- Or DNS update (RFC 2136), authenticated with DNS-TSIG (RFC 2845)
- Successful nsupdate will `git push` by the server to the repository
- Other servers are notified on update, and start zone transfer (AXFR RFC 5936, incremental IXFR RFC 1995)

Case study authoritative DNS server

- Domain Name System, used for translating hostnames into Internet addresses
- Authoritative server replies to DNS requests
- Data (zone) is kept in memory, no block device
- Storage in a git remote in zone file format
- Configuration (ip address, git remote, syslog, ..) via boot arguments

- Modification via git commit and push, sending a notify (RFC 1996) to server
- Or DNS update (RFC 2136), authenticated with DNS-TSIG (RFC 2845)
- Successful nsupdate will `git push` by the server to the repository
- Other servers are notified on update, and start zone transfer (AXFR RFC 5936, incremental IXFR RFC 1995)

- Image size ~9MB (IP stack, DNS, git, ssh)
- Let's encrypt integration, signing requests and certificates stored in DNS
- `https://hannes.nqsb.io/Posts/DnsServer`

- QubesOS is a "reasonable 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
- Support for dynamic rules for Qubes 4.0 is under review
- `https://github.com/mirage/qubes-mirage-firewall`

- https://github.com/mirage
- https://github.com/roburio

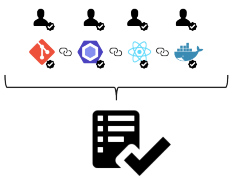
More MirageOS unikernels

- A picture viewer `https://github.com/cfcs/eye-of-mirage`
- `ssh-agent` for Qubes `https://github.com/reynir/qubes-mirage-ssh-agent`
- Web sites: `https://mirage.io`, `https://nqsb.io`
- `Canopy` serves markdown content from a git repository as website, `https://github.com/Engil/Canopy`
- DHCP server `https://github.com/mirage/charrua`
- OpenVPN client and server `https://github.com/roburio/openvpn`
- Pastebin clone `https://github.com/dinosaure/pasteur`
- Pong game `https://github.com/cfcs/PongOS`
- Z machine (Zork) via telnet `https://github.com/mato/flathead`
- `https://github.com/roburio/unikernels`

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

Supply chain security (wip)

- OCaml package authors should sign their releases
- A quorum of repository maintainers can delegate a package to authors
- Key compromise impact is contained to the delegated packages of the author
- 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
- See `https://github.com/hannesm/jackline-opam`



Conventional orchestration systems

- Lack decent integration of MirageOS
- `mirage` generates a `libvirt.xml` for each unikernel
- Also `.xl` and `.xe` for Xen unikernels

Albatross

- Minimal orchestration system for MirageOS unikernels, with optional remote deployment
- Metrics, console access, multi-tenant supported (resource policies in CA chain)
- A unikernel image stored in a TLS client certificate can be deployed remotely
- `https://hannes.nqsb.io/Posts/VMM`

- Research at University of Cambridge since 2008 (ongoing student projects, etc.)
- Bi-annual hack retreats `http://retreat.mirage.io`
- Dogfooding our unikernels (DHCP, DNS)
- Open source contributors from all over the world
- Docker for Mac and Docker for Windows use MirageOS libraries

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 (25x - 100x less code)
- Drastically reduced carbon footprint (10x less CPU, 25x less memory)
- Reasonable performance, boots in milliseconds
- Thanks to everybody involved working on this technology stack :D
- We at `https://robur.coop` develop full-stack MirageOS unikernels

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`