Unikernels: Paths to Production & Current Research Trends

Hugo Lefeuvre

The University of Manchester

ASPLOS 2022 Unikraft Tutorial, *March 1st*

A Decade of Unikernels...

~2012: A. Kantee's thesis on Rump kernels

A tad later, Mirage (ASPLOS'13), coined the term "unikernel".

A Decade of Unikernels...

~2012: A. Kantee's thesis on Rump kernels

A tad later, Mirage (ASPLOS'13), coined the term "unikernel".

Now, quite a large body of work

A Decade of Unikernels...

~2012: A. Kantee's thesis on Rump kernels

A tad later, Mirage (ASPLOS'13), coined the term "unikernel".

Now, quite a large body of work

ClickOS (NSDI'24) Mirage (ASPLOS'13)

Specialization / Performance

Unikraft (EuroSys'21)

A Decade of Unikernels...

~2012: A. Kantee's thesis on Rump kernels

A tad later, Mirage (ASPLOS'13), coined the term "unikernel".

Now, quite a large body of work

ClickOS (NSDI'24)

Mirage (ASPLOS'13)

Specialization / Performance

Unikraft (EuroSys'21)

LightVM (SOSP'17)

Firecracker (NSDI'21)

USETL (ApSys)

Scalability

Solo5 (HotCloud/SoCC)

A Decade of Unikernels...

~2012: A. Kantee's thesis on Rump kernels

A tad later, Mirage (ASPLOS'13), coined the term "unikernel".

Now, quite a large body of work

ClickOS (NSDI'24) Mirage (ASPLOS'13)

Specialization / Performance

Unikraft (EuroSys'21)

LightVM (SOSP'17)

Firecracker (NSDI'21) USETL (ApSys)

Scalability

Solo5 (HotCloud/SoCC)

EbbRT (OSDI'16)

Lupine Linux (EuroSys'20)

Compatibility

OSv (ATC'14) HermiTux (VEE'19)

A Decade of Unikernels...

So, a lot of interest, a lot of nice ideas.

A Decade of Unikernels...

So, a lot of interest, a lot of nice ideas.

And *yet*, in practice, we don't see unikernels in production (even in the cloud) in 2022.

A Decade of Unikernels...

So, a lot of interest, a lot of nice ideas.

And *yet*, in practice, we don't see unikernels in production (even in the cloud) in 2022.

Why?

A Decade of Unikernels...

So, a lot of interest, a lot of nice ideas.

And *yet*, in practice, we don't see unikernels in production (even in the cloud) in 2022.

Why?

Current unikernels are just (mostly academic) **research prototypes**

Unikernels are Research Prototypes

Since they are research prototypes, none **managed (or even tried!) to**:

Unikernels are Research Prototypes

Since they are research prototypes, none **managed (or even tried!) to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows

Unikernels are Research Prototypes

Since they are research prototypes, none **managed (or even tried!) to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows
- take security seriously
- meet production-grade testing standards
- good debuggability

Unikernels are Research Prototypes

Since they are research prototypes, none **managed (or even tried!) to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows
- take security seriously
- meet production-grade testing standards
- good debuggability

Now, if we want to see unikernels in production one day, one project must show that it can be done because some of these items are not trivial. **We want this project to be Unikraft.**

Unikernels are Research Prototypes

Since they are research prototypes, none **managed (or even tried!) to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows
- take security seriously
- meet production-grade testing standards
- good debuggability

Now, if we want to see unikernels in production one day, one project must show that it can be done because some of these items are not trivial. **We want this project to be Unikraft.**

Unikraft is not there yet. But it's progressing, and we hope to see it reaching full maturity in the coming year

Unikernels are Research Prototypes

Since they are research prototypes, none **managed (or even tried!) to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows
- take security seriously
- meet production-grade testing standards
- good debuggability

Now, if we want to see unikernels in production one day, one project must show that it can be done because some of these items are not trivial. **We want this project to be Unikraft.**

Unikraft is not there yet. But it's progressing, and we hope to see it reaching full maturity in the coming year

In the Works for Unikraft

Integration and Scalability

Security and Stability

In the Works for Unikraft

Integration and Scalability

Security and Stability

- Making Unikraft fit for classical deployment workflows (Kubernetes, etc.)

In the Works for Unikraft

Integration and Scalability

- Making Unikraft fit for classical deployment workflows (Kubernetes, etc.)
- Making Unikraft just as debuggable as any userland application

Security and Stability

In the Works for Unikraft

Integration and Scalability

Security and Stability

- Making Unikraft fit for classical deployment workflows (Kubernetes, etc.)
- Making Unikraft just as debuggable as any userland application
- Open question: real world high-density: achieving 100s of unikernels per host?

In the Works for Unikraft

Integration and Scalability

- Making Unikraft fit for classical deployment workflows (Kubernetes, etc.)
- Making Unikraft just as debuggable as any userland application
- Open question: real world high-density: achieving 100s of unikernels per host?

Security and Stability

- Matching the security/hardening features of mainstream OSes

In the Works for Unikraft

Integration and Scalability

- Making Unikraft fit for classical deployment workflows (Kubernetes, etc.)
- Making Unikraft just as debuggable as any userland application
- Open question: real world high-density: achieving 100s of unikernels per host?

Security and Stability

- Matching the security/hardening features of mainstream OSes
- Production-grade testing and fuzzing of Unikraft

In the Works for Unikraft

Integration and Scalability

- Making Unikraft fit for classical deployment workflows (Kubernetes, etc.)
- Making Unikraft just as debuggable as any userland application
- Open question: real world high-density: achieving 100s of unikernels per host?

Security and Stability

- Matching the security/hardening features of mainstream OSes
- Production-grade testing and fuzzing of Unikraft
- Specialization for the masses: automatic reasoning about Unikraft configurations

In the Works for Unikraft

Integration and Scalability

- Making Unikraft fit for classical deployment workflows (Kubernetes, etc.)
- Making Unikraft just as debuggable as any userland application
- Open question: real world high-density: achieving 100s of unikernels per host?

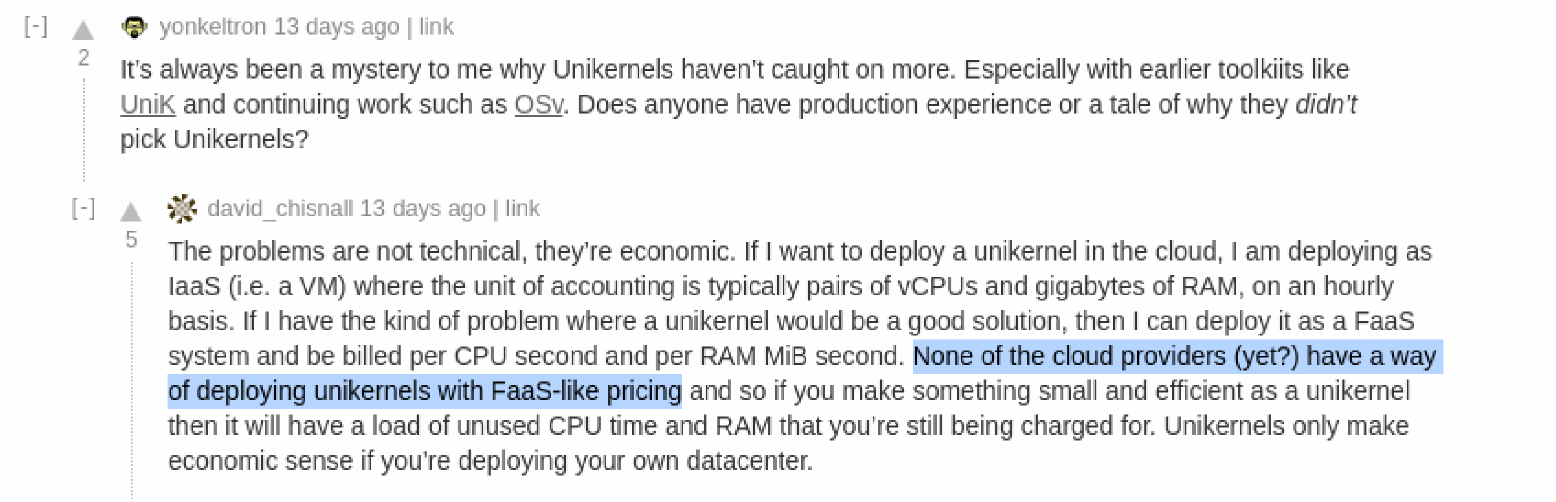
Security and Stability

- Matching the security/hardening features of mainstream OSes
- Production-grade testing and fuzzing of Unikraft
- Specialization for the masses: automatic reasoning about Unikraft configurations
- Beyond the single trust domain: compartmentalizing Unikraft?

Integration & Scalability

Integration and Scalability

Why no unikernels in production?



[-] ▲ 2 yonkeltron 13 days ago | link

It's always been a mystery to me why Unikernels haven't caught on more. Especially with earlier toolkiits like UniK and continuing work such as OSv. Does anyone have production experience or a tale of why they *didn't* pick Unikernels?

[-] ▲ 5 david_chisnall 13 days ago | link

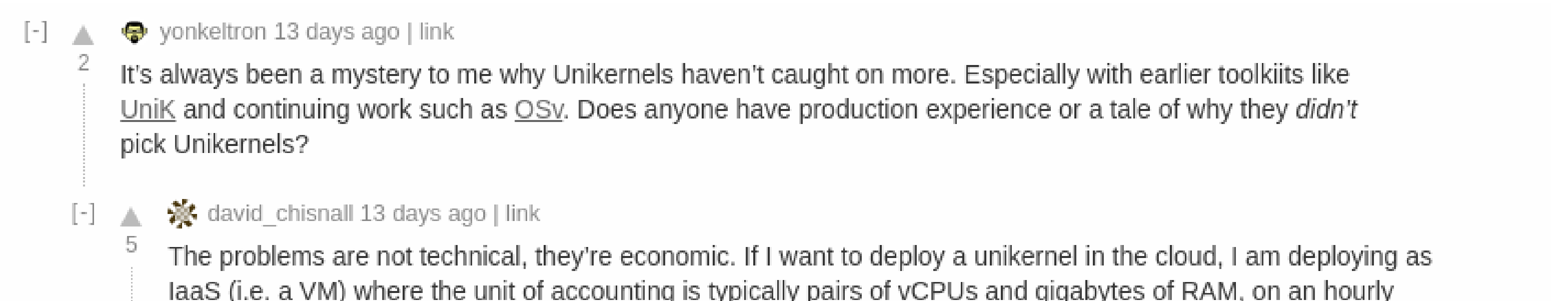
The problems are not technical, they're economic. If I want to deploy a unikernel in the cloud, I am deploying as IaaS (i.e. a VM) where the unit of accounting is typically pairs of vCPUs and gigabytes of RAM, on an hourly basis. If I have the kind of problem where a unikernel would be a good solution, then I can deploy it as a FaaS system and be billed per CPU second and per RAM MiB second. None of the cloud providers (yet?) have a way of deploying unikernels with FaaS-like pricing and so if you make something small and efficient as a unikernel then it will have a load of unused CPU time and RAM that you're still being charged for. Unikernels only make economic sense if you're deploying your own datacenter.

No good integration.

https://lobste.rs/s/cyyx7a/unikraft_fast_secure_open_source (NOT an official Microsoft comment)

Integration and Scalability

Why no unikernels in production?



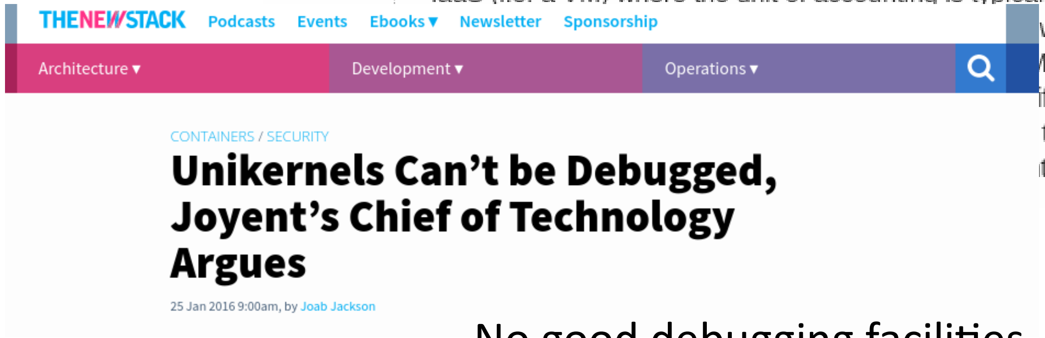
[-] ▲ 2 yonkeltron 13 days ago | link

It's always been a mystery to me why Unikernels haven't caught on more. Especially with earlier toolkiits like UniK and continuing work such as OSv. Does anyone have production experience or a tale of why they *didn't* pick Unikernels?

[-] ▲ 5 david_chisnall 13 days ago | link

The problems are not technical, they're economic. If I want to deploy a unikernel in the cloud, I am deploying as IaaS (i.e. a VM) where the unit of accounting is typically pairs of vCPUs and gigabytes of RAM, on an hourly

would be a good solution, then I can deploy it as a FaaS
iB second. None of the cloud providers (yet?) have a way
f you make something small and efficient as a unikernel
that you're still being charged for. Unikernels only make
ter.



CONTAINERS / SECURITY

Unikernels Can't be Debugged, Joyent's Chief of Technology Argues

25 Jan 2016 9:00am, by Joab Jackson

No good integration.

No good debugging facilities.

https://lobste.rs/s/cyyx7a/unikraft_fast_secure_open_source (NOT an official Microsoft comment)
https://thenewstack.io/good-luck-debugging-unikernels-joyents-chief-technology-says/

In the Works: Integration

People need integration of unikernels into orchestration frameworks to truly leverage their benefits.

In the Works: Integration

People need integration of unikernels into orchestration frameworks to truly leverage their benefits.

- Dynamically and quickly provision new services
- Schedule/Reschedule services based on workload

(etc.)

Typically with

kubernetes

In the Works: Integration

People need integration of unikernels into orchestration frameworks to truly leverage their benefits.

- Dynamically and quickly provision new services
- Schedule/Reschedule services based on workload

(etc.)

Typically with kubernetes

No need to reinvent the wheel: make unikernels fit in these frameworks

In the Works: Integration

People need integration of unikernels into orchestration frameworks to truly leverage their benefits.

- Dynamically and quickly provision new services
- Schedule/Reschedule services based on workload

(etc.)

Typically with

kubernetes

No need to reinvent the wheel: make unikernels fit in these frameworks

We are almost there.

- Integration of unikernels in Kubernetes infrastucture
  - OCI-compliant unikernel runtime interface

A. Jung @ CNCF'21 https://www.youtube.com/watch?v=cV-xawN9_cg

In the Works: Integration

People need integration of unikernels into orchestration frameworks to truly leverage their benefits.

- Dynamically and quickly provision new services
- Schedule/Reschedule services based on workload (etc.)

Typically with

kubernetes

No need to reinvent the wheel: make unikernels fit in these frameworks

We are almost there.

- Integration of unikernels in Kubernetes infrastucture
  - OCI-compliant unikernel runtime interface
- "A Unikernel in OCI Clothing": make unikernels look and feel like containers

A. Jung @ CNCF'21 https://www.youtube.com/watch?v=cV-xawN9_cg

In the Works: Integration

Good Progress

People need integration of unikernels into orchestration frameworks to truly leverage their benefits.

- Dynamically and quickly provision new services
- Schedule/Reschedule services based on workload (etc.)

Typically with kubernetes

No need to reinvent the wheel: make unikernels fit in these frameworks

We are almost there.

- Integration of unikernels in Kubernetes infrastucture
  - OCI-compliant unikernel runtime interface
- "A Unikernel in OCI Clothing": make unikernels look and feel like containers
- More progress to make on the FaaS side?

A. Jung @ CNCF'21 https://www.youtube.com/watch?v=cV-xawN9_cg

In the Works: Debugging

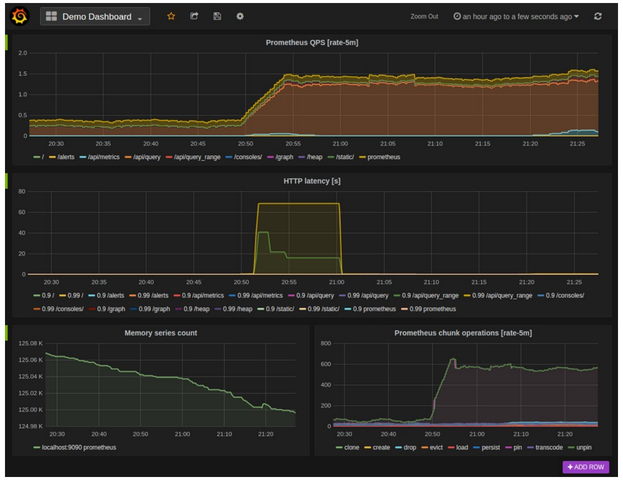
Vast engineering effort towards seamless introspection and debugging

S. Kuenzer, M. Rittinghaus @ FOSDEM'22 https://fosdem.org/2022/schedule/event/skuenzer/

In the Works: Debugging

Vast engineering effort towards seamless introspection and debugging

- Production-grade monitoring with Prometheus:
  - Monitor unikernels like any general-purpose VM
  - Setup alarms when values pass thresholds, etc.

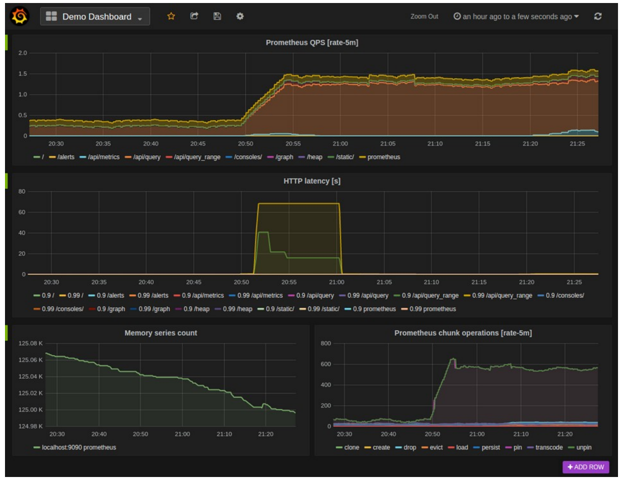


S. Kuenzer, M. Rittinghaus @ FOSDEM'22 https://fosdem.org/2022/schedule/event/skuenzer/

In the Works: Debugging

Vast engineering effort towards seamless introspection and debugging

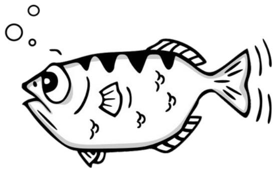
- Production-grade monitoring with Prometheus:
  - Monitor unikernels like any general-purpose VM
  - Setup alarms when values pass thresholds, etc.
- OS-level native GDB debugger support:
  - Debug unikernels like any userland application
  - Support for threads, OS-specific constructs (asserts, kernel crashes, etc.)
- Uniform debugging experience all platforms



S. Kuenzer, M. Rittinghaus @ FOSDEM'22 https://fosdem.org/2022/schedule/event/skuenzer/

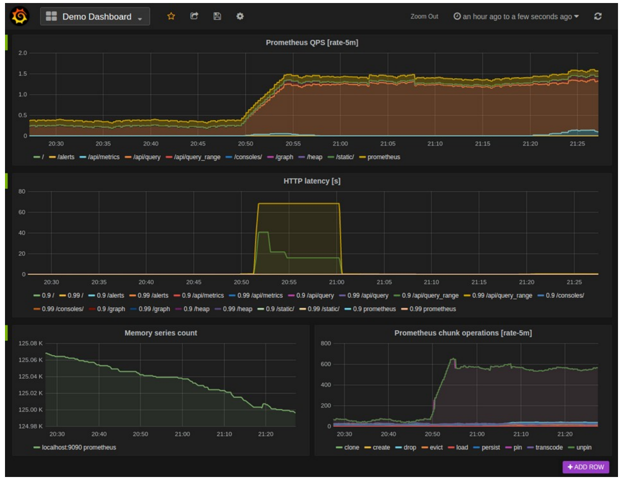
In the Works: Debugging

Good Progress



Vast engineering effort towards seamless introspection and debugging

- Production-grade monitoring with Prometheus:
  - Monitor unikernels like any general-purpose VM
  - Setup alarms when values pass thresholds, etc.
- OS-level native GDB debugger support:
  - Debug unikernels like any userland application
  - Support for threads, OS-specific constructs (asserts, kernel crashes, etc.)
- Uniform debugging experience all platforms



S. Kuenzer, M. Rittinghaus @ FOSDEM'22 https://fosdem.org/2022/schedule/event/skuenzer/

Outstanding Question: Scalability?

Outstanding Question: Scalability?

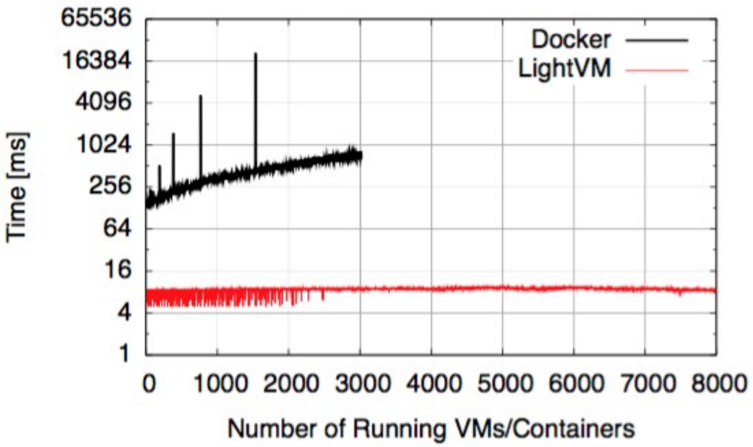
As unikernels are evolving towards production, more challenges are likely to arise

We (as a community) claimed **massive packing** of such VMs on a single host

Outstanding Question: Scalability?

As unikernels are evolving towards production, more challenges are likely to arise

We (as a community) claimed **massive packing** of such VMs on a single host



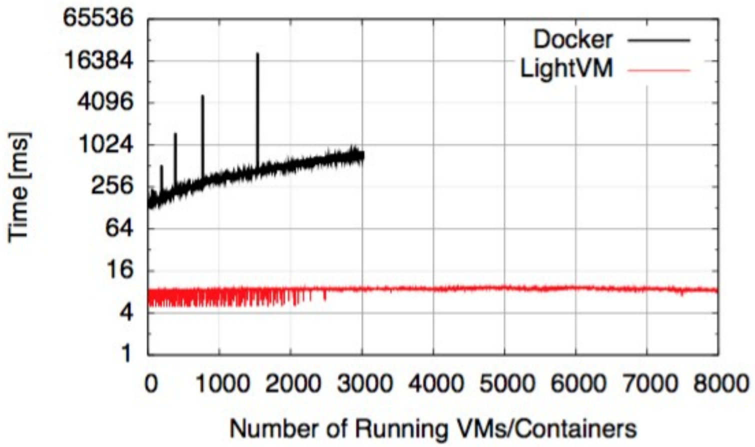
LightVM (SOSP'17) paper: 1000s of noop unikernels on a single host

Outstanding Question: Scalability?

As unikernels are evolving towards production, more challenges are likely to arise

We (as a community) claimed **massive packing** of such VMs on a single host

… but **are we ready to see 50 Nginx instances on a single host?** Let alone 100s?



LightVM (SOSP'17) paper: 1000s of noop unikernels on a single host

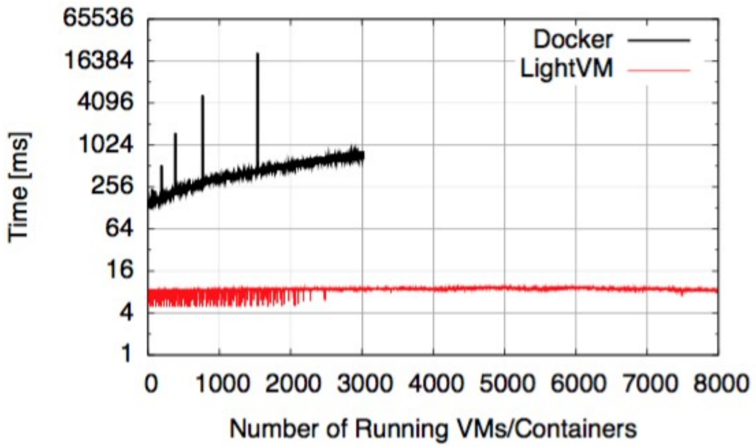
Outstanding Question: Scalability?

As unikernels are evolving towards production, more challenges are likely to arise

We (as a community) claimed **massive packing** of such VMs on a single host

… but **are we ready to see 50 Nginx instances on a single host?** Let alone 100s?

Pretty much all unikernel papers evaluate systems with 1 CPU = 1vCPU static pinning...



LightVM (SOSP'17) paper: 1000s of noop unikernels on a single host

Outstanding Question: Scalability?

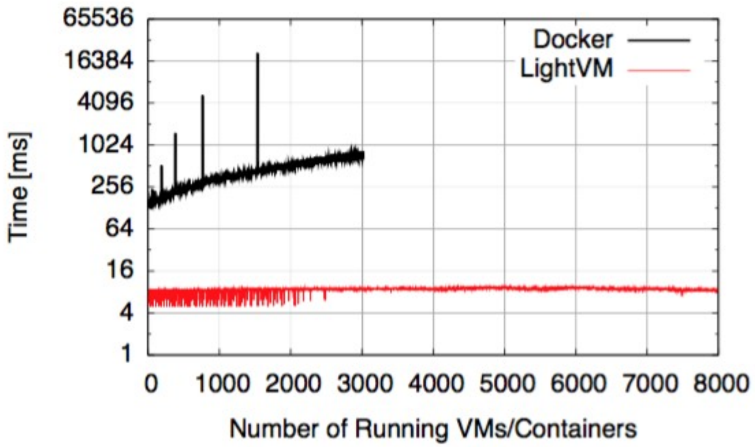
As unikernels are evolving towards production, more challenges are likely to arise

We (as a community) claimed **massive packing** of such VMs on a single host

… but **are we ready to see 50 Nginx instances on a single host?** Let alone 100s?

Pretty much all unikernel papers evaluate systems with 1 CPU = 1vCPU static pinning...

With such density: how do things look on the networking side? Have hypervisors really been thought for this kind of usage?



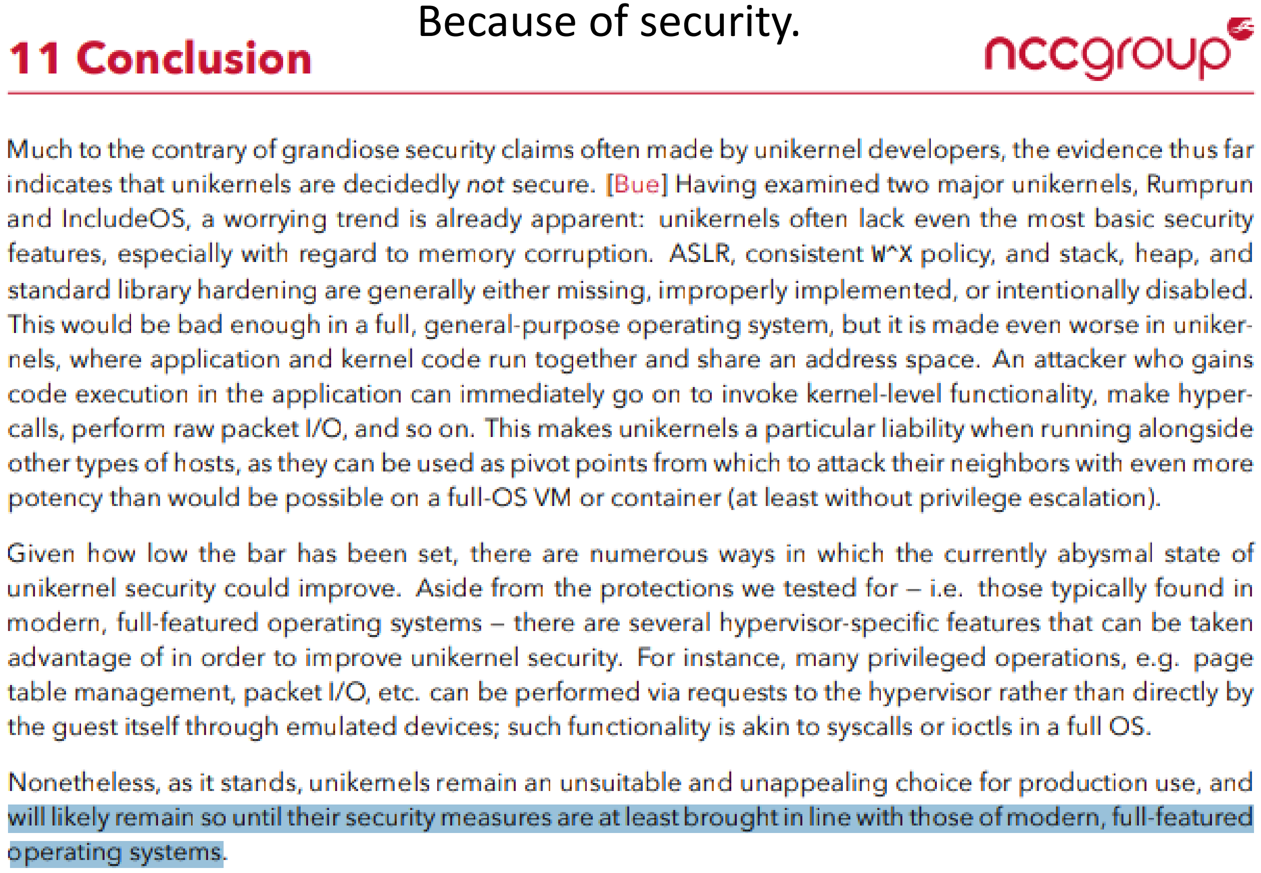
LightVM (SOSP'17) paper: 1000s of noop unikernels on a single host

Security & Stability

Security and Stability

Why no unikernels in production?

Because of security.



11 Conclusion

nccgroup

Much to the contrary of grandiose security claims often made by unikernel developers, the evidence thus far indicates that unikernels are decidedly *not* secure. [Bue] Having examined two major unikernels, Rumprun and IncludeOS, a worrying trend is already apparent: unikernels often lack even the most basic security features, especially with regard to memory corruption. ASLR, consistent `W^X` policy, and stack, heap, and standard library hardening are generally either missing, improperly implemented, or intentionally disabled. This would be bad enough in a full, general-purpose operating system, but it is made even worse in unikernels, where application and kernel code run together and share an address space. An attacker who gains code execution in the application can immediately go on to invoke kernel-level functionality, make hypercalls, perform raw packet I/O, and so on. This makes unikernels a particular liability when running alongside other types of hosts, as they can be used as pivot points from which to attack their neighbors with even more potency than would be possible on a full-OS VM or container (at least without privilege escalation).

Given how low the bar has been set, there are numerous ways in which the currently abysmal state of unikernel security could improve. Aside from the protections we tested for – i.e. those typically found in modern, full-featured operating systems – there are several hypervisor-specific features that can be taken advantage of in order to improve unikernel security. For instance, many privileged operations, e.g. page table management, packet I/O, etc. can be performed via requests to the hypervisor rather than directly by the guest itself through emulated devices; such functionality is akin to syscalls or ioctls in a full OS.

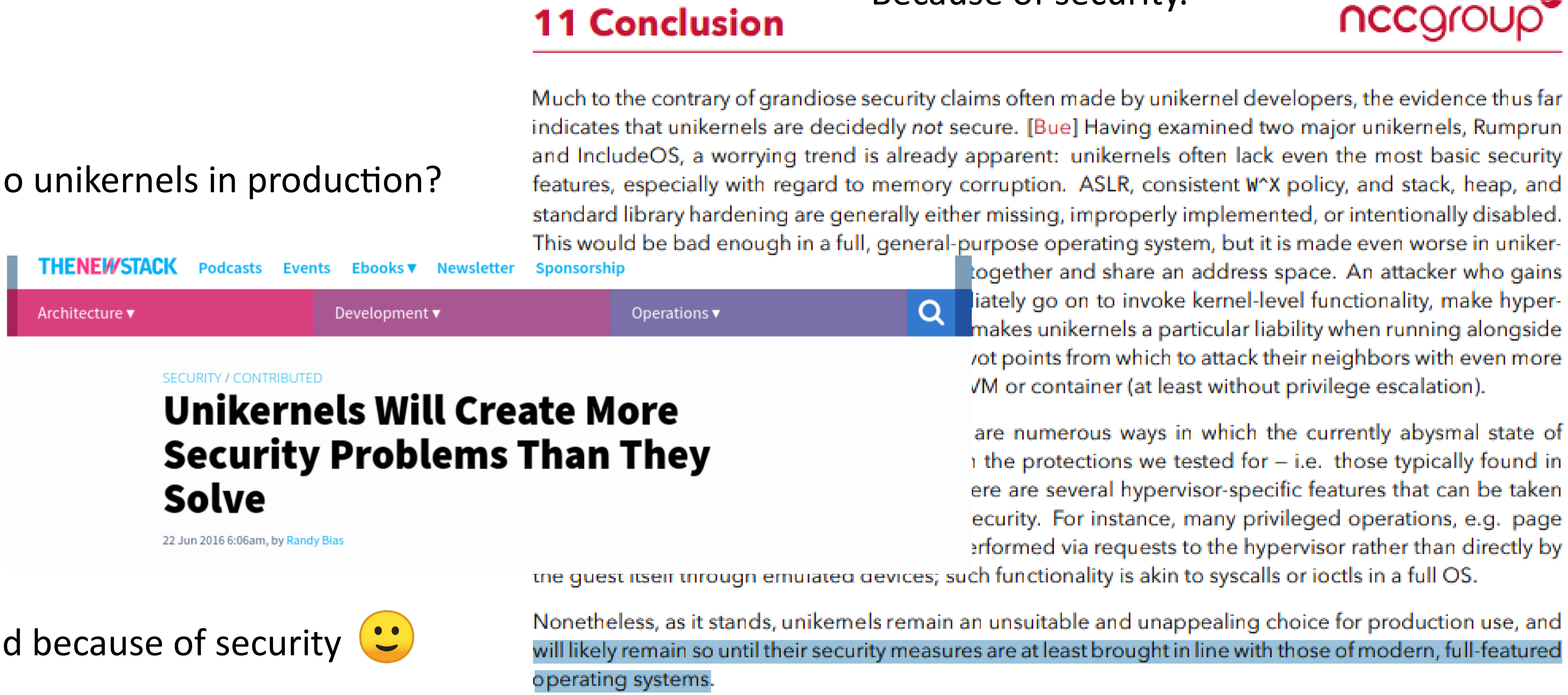
Nonetheless, as it stands, unikemels remain an unsuitable and unappealing choice for production use, and will likely remain so until their security measures are at least brought in line with those of modern, full-featured operating systems.

https://www.nccgroup.com/globalassets/our-research/us/whitepapers/2019/ncc_group-assessing_unikernel_security.pdf

Security and Stability

Because of security.

Why no unikernels in production?



11 Conclusion

Much to the contrary of grandiose security claims often made by unikernel developers, the evidence thus far indicates that unikernels are decidedly *not* secure. [Bue] Having examined two major unikernels, Rumprun and IncludeOS, a worrying trend is already apparent: unikernels often lack even the most basic security features, especially with regard to memory corruption. ASLR, consistent `W^X` policy, and stack, heap, and standard library hardening are generally either missing, improperly implemented, or intentionally disabled. This would be bad enough in a full, general-purpose operating system, but it is made even worse in uniker-

together and share an address space. An attacker who gains
iately go on to invoke kernel-level functionality, make hyper-
makes unikernels a particular liability when running alongside
/ot points from which to attack their neighbors with even more
/M or container (at least without privilege escalation).

are numerous ways in which the currently abysmal state of
ı the protections we tested for – i.e. those typically found in
ere are several hypervisor-specific features that can be taken
ecurity. For instance, many privileged operations, e.g. page
erformed via requests to the hypervisor rather than directly by
the guest itself through emulated devices; such functionality is akin to syscalls or ioctls in a full OS.

Nonetheless, as it stands, unikernels remain an unsuitable and unappealing choice for production use, and will likely remain so until their security measures are at least brought in line with those of modern, full-featured operating systems.

SECURITY / CONTRIBUTED

Unikernels Will Create More Security Problems Than They Solve

22 Jun 2016 6:06am, by Randy Bias

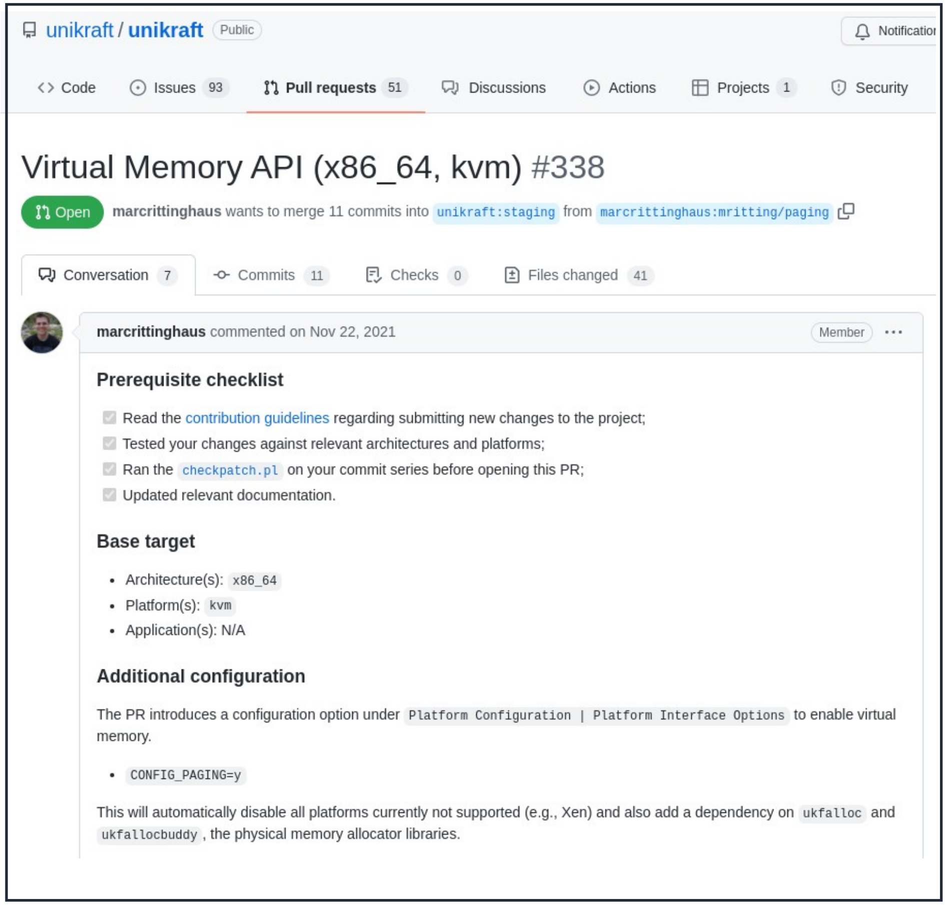
...and because of security 🙂

https://www.nccgroup.com/globalassets/our-research/us/whitepapers/2019/ncc_group-assessing_unikernel_security.pdf
https://thenewstack.io/unikernels-will-create-security-problems-solve/

Getting in Line with Mainstream OSes

Getting in Line with Mainstream OSes

Write or Execute



Virtual Memory API (x86_64, kvm) #338

marcrittinghaus wants to merge 11 commits into `unikraft:staging` from `marcrittinghaus:mritting/paging`

marcrittinghaus commented on Nov 22, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submitting new changes to the project;
- ☑ Tested your changes against relevant architectures and platforms;
- ☑ Ran the `checkpatch.pl` on your commit series before opening this PR;
- ☑ Updated relevant documentation.

Base target

- Architecture(s): `x86_64`
- Platform(s): `kvm`
- Application(s): N/A

Additional configuration

The PR introduces a configuration option under `Platform Configuration | Platform Interface Options` to enable virtual memory.

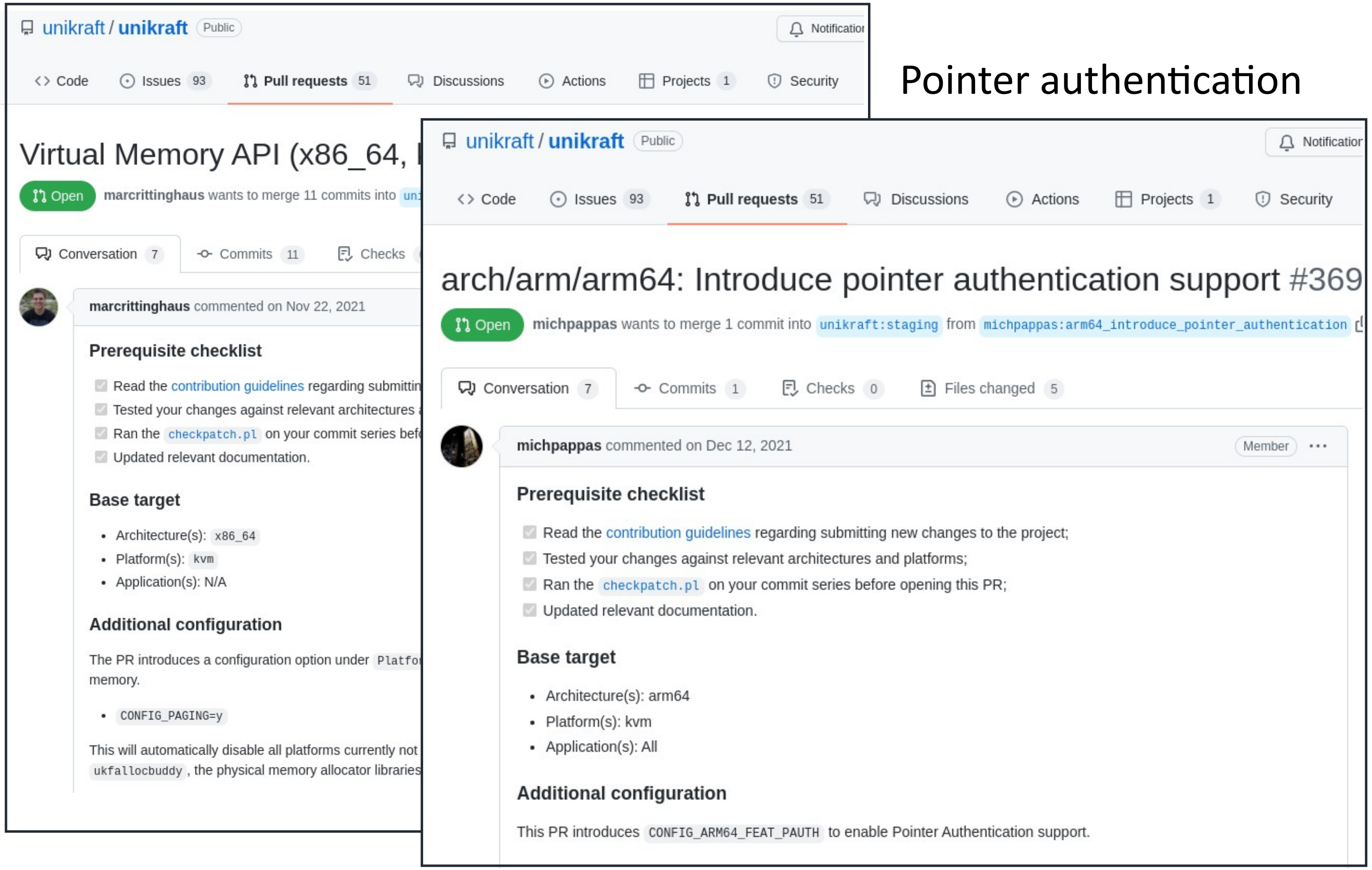
- `CONFIG_PAGING=y`

This will automatically disable all platforms currently not supported (e.g., Xen) and also add a dependency on `ukfalloc` and `ukfallocbuddy`, the physical memory allocator libraries.

Getting in Line with Mainstream OSes

Write or Execute

Pointer authentication



Virtual Memory API (x86_64, I

marcrittinghaus wants to merge 11 commits into un

marcrittinghaus commented on Nov 22, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submittin
- ☑ Tested your changes against relevant architectures a
- ☑ Ran the `checkpatch.pl` on your commit series befo
- ☑ Updated relevant documentation.

Base target

- Architecture(s): `x86_64`
- Platform(s): `kvm`
- Application(s): N/A

Additional configuration

The PR introduces a configuration option under `Platfo` memory.

- `CONFIG_PAGING=y`

This will automatically disable all platforms currently not `ukfallocbuddy`, the physical memory allocator libraries

arch/arm/arm64: Introduce pointer authentication support #369

michpappas wants to merge 1 commit into `unikraft:staging` from `michpappas:arm64_introduce_pointer_authentication`

michpappas commented on Dec 12, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submitting new changes to the project;
- ☑ Tested your changes against relevant architectures and platforms;
- ☑ Ran the `checkpatch.pl` on your commit series before opening this PR;
- ☑ Updated relevant documentation.

Base target

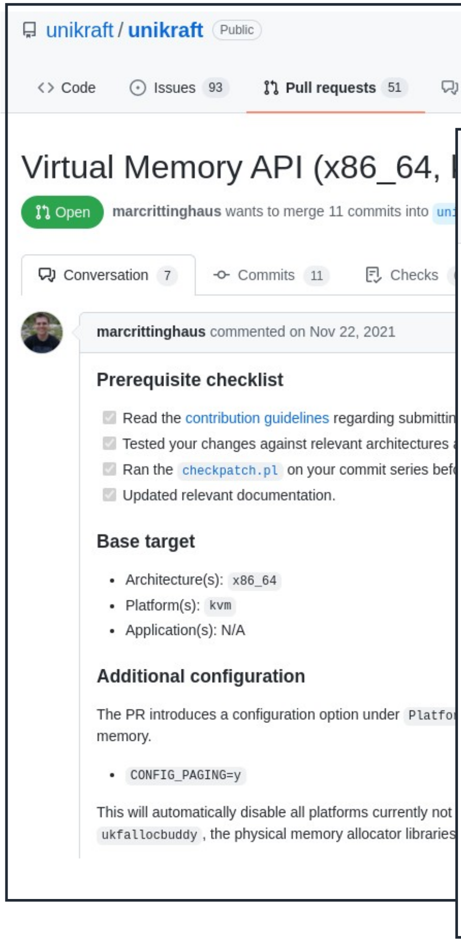
- Architecture(s): arm64
- Platform(s): kvm
- Application(s): All

Additional configuration

This PR introduces `CONFIG_ARM64_FEAT_PAUTH` to enable Pointer Authentication support.

Getting in Line with Mainstream OSes

Write or Execute



Virtual Memory API (x86_64,

Open marcrittinghaus wants to merge 11 commits into un

marcrittinghaus commented on Nov 22, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submittin
- ☑ Tested your changes against relevant architectures
- ☑ Ran the `checkpatch.pl` on your commit series befo
- ☑ Updated relevant documentation.

Base target

- Architecture(s): `x86_64`
- Platform(s): `kvm`
- Application(s): N/A

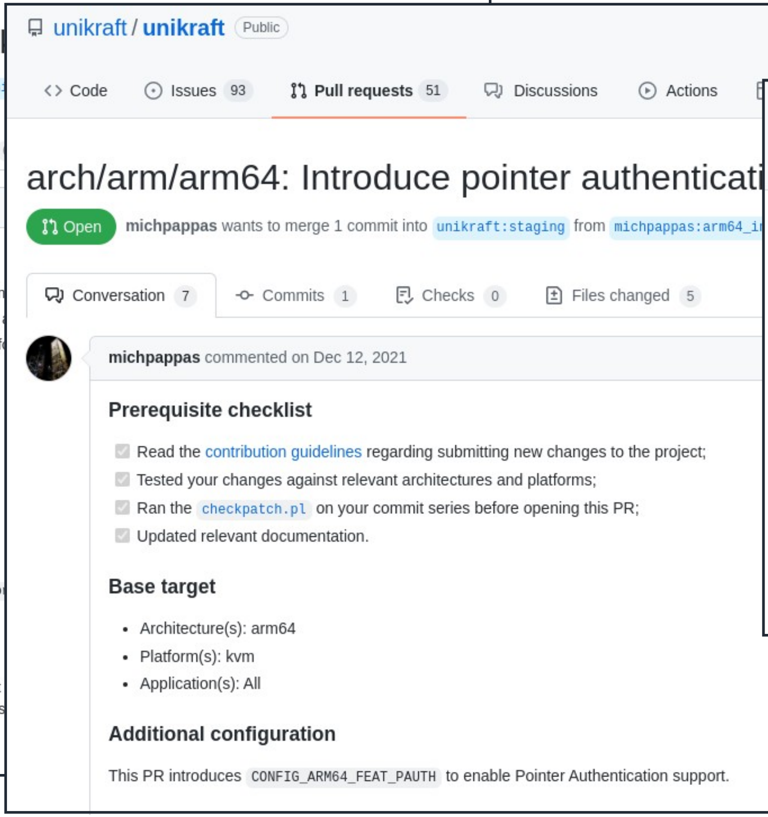
Additional configuration

The PR introduces a configuration option under `Platfo` memory.

- `CONFIG_PAGING=y`

This will automatically disable all platforms currently not `ukfallocbuddy`, the physical memory allocator libraries

Pointer authentication



arch/arm/arm64: Introduce pointer authenticati

Open michpappas wants to merge 1 commit into `unikraft:staging` from `michpappas:arm64_i`

michpappas commented on Dec 12, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submitting new changes to the project;
- ☑ Tested your changes against relevant architectures and platforms;
- ☑ Ran the `checkpatch.pl` on your commit series before opening this PR;
- ☑ Updated relevant documentation.

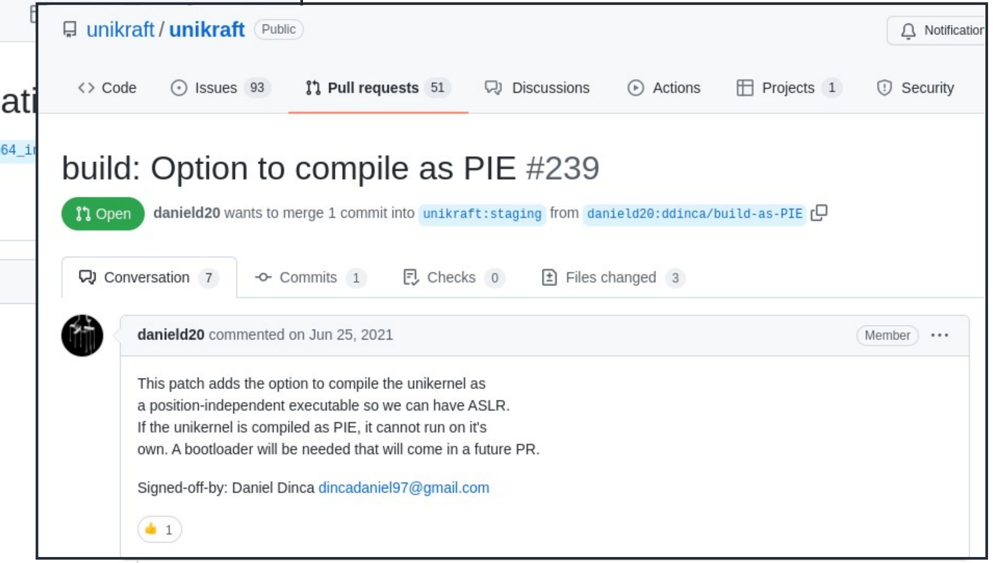
Base target

- Architecture(s): arm64
- Platform(s): kvm
- Application(s): All

Additional configuration

This PR introduces `CONFIG_ARM64_FEAT_PAUTH` to enable Pointer Authentication support.

ASLR



build: Option to compile as PIE #239

Open danield20 wants to merge 1 commit into `unikraft:staging` from `danield20:ddinca/build-as-PIE`

danield20 commented on Jun 25, 2021

This patch adds the option to compile the unikernel as
a position-independent executable so we can have ASLR.
If the unikernel is compiled as PIE, it cannot run on it's
own. A bootloader will be needed that will come in a future PR.

Signed-off-by: Daniel Dinca dincadaniel97@gmail.com

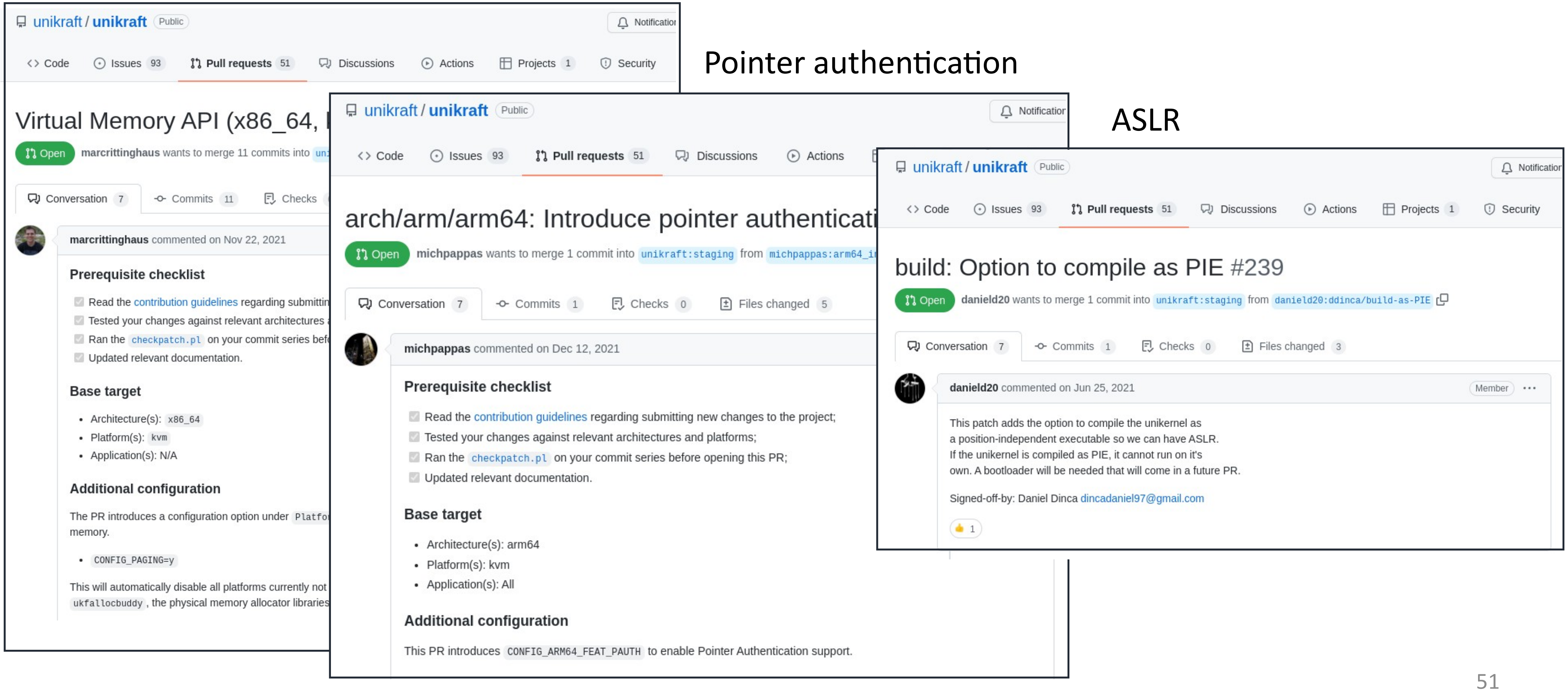
Getting in Line with Mainstream OSes

Write or Execute

... and stack protection, KASan, etc.

Pointer authentication

ASLR



Virtual Memory API (x86_64,

Open marcrittinghaus wants to merge 11 commits into un

Conversation 7 · Commits 11 · Checks

marcrittinghaus commented on Nov 22, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submittin
- ☑ Tested your changes against relevant architectures
- ☑ Ran the `checkpatch.pl` on your commit series bef
- ☑ Updated relevant documentation.

Base target

- Architecture(s): `x86_64`
- Platform(s): `kvm`
- Application(s): N/A

Additional configuration

The PR introduces a configuration option under `Platfo` memory.

- `CONFIG_PAGING=y`

This will automatically disable all platforms currently not `ukfallocbuddy`, the physical memory allocator libraries

arch/arm/arm64: Introduce pointer authenticati

Open michpappas wants to merge 1 commit into `unikraft:staging` from `michpappas:arm64_i`

Conversation 7 · Commits 1 · Checks 0 · Files changed 5

michpappas commented on Dec 12, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submitting new changes to the project;
- ☑ Tested your changes against relevant architectures and platforms;
- ☑ Ran the `checkpatch.pl` on your commit series before opening this PR;
- ☑ Updated relevant documentation.

Base target

- Architecture(s): arm64
- Platform(s): kvm
- Application(s): All

Additional configuration

This PR introduces `CONFIG_ARM64_FEAT_PAUTH` to enable Pointer Authentication support.

build: Option to compile as PIE #239

Open danield20 wants to merge 1 commit into `unikraft:staging` from `danield20:ddinca/build-as-PIE`

Conversation 7 · Commits 1 · Checks 0 · Files changed 3

danield20 commented on Jun 25, 2021 Member

This patch adds the option to compile the unikernel as
a position-independent executable so we can have ASLR.
If the unikernel is compiled as PIE, it cannot run on it's
own. A bootloader will be needed that will come in a future PR.

Signed-off-by: Daniel Dinca dincadaniel97@gmail.com

👍 1

Getting in Line with Mainstream OSes

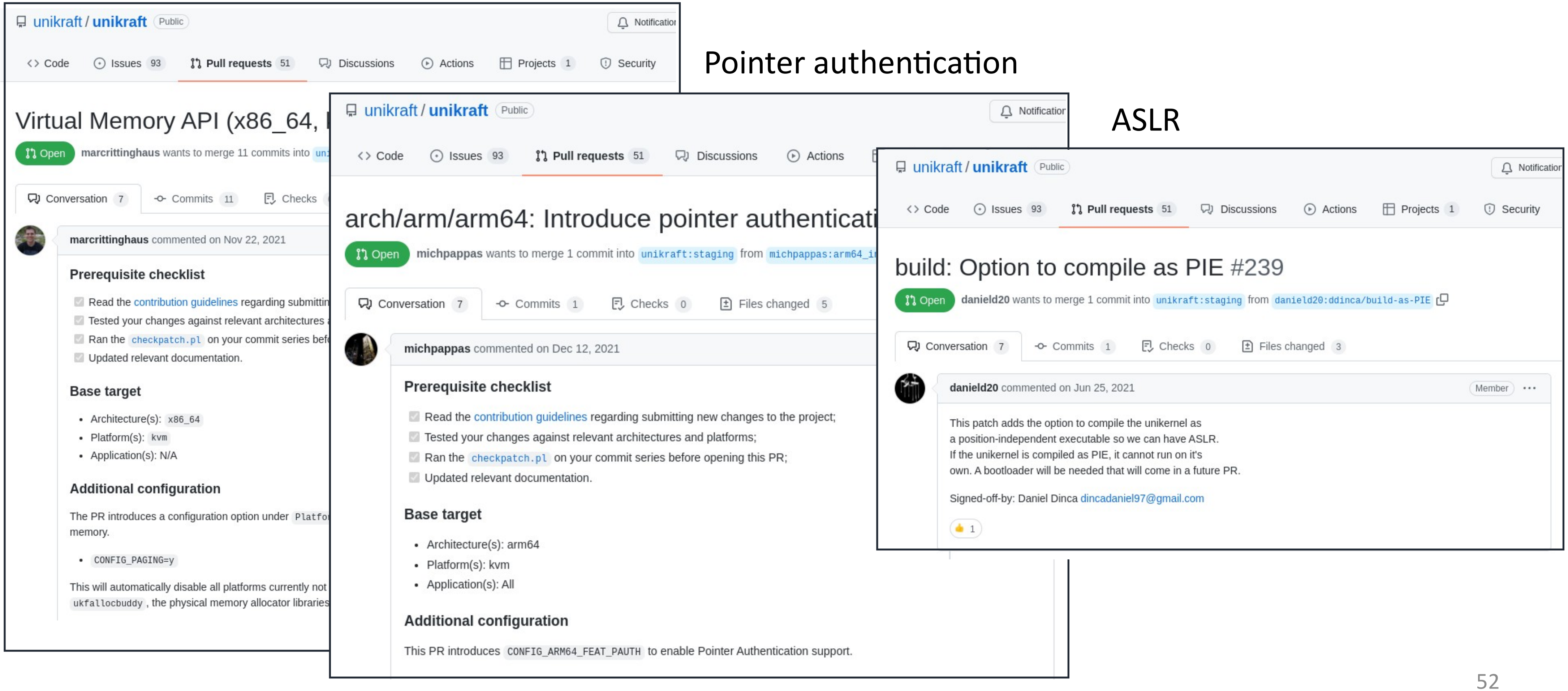
Good Progress

Write or Execute

Pointer authentication

ASLR

... and stack protection, KASan, etc.



Virtual Memory API (x86_64, I

Open marcrittinghaus wants to merge 11 commits into un

marcrittinghaus commented on Nov 22, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submittin
- ☑ Tested your changes against relevant architectures a
- ☑ Ran the `checkpatch.pl` on your commit series bef
- ☑ Updated relevant documentation.

Base target

- Architecture(s): `x86_64`
- Platform(s): `kvm`
- Application(s): N/A

Additional configuration

The PR introduces a configuration option under `Platfo`
memory.

- `CONFIG_PAGING=y`

This will automatically disable all platforms currently not
`ukfallocbuddy`, the physical memory allocator libraries

arch/arm/arm64: Introduce pointer authenticati

Open michpappas wants to merge 1 commit into `unikraft:staging` from `michpappas:arm64_i`

michpappas commented on Dec 12, 2021

Prerequisite checklist

- ☑ Read the contribution guidelines regarding submitting new changes to the project;
- ☑ Tested your changes against relevant architectures and platforms;
- ☑ Ran the `checkpatch.pl` on your commit series before opening this PR;
- ☑ Updated relevant documentation.

Base target

- Architecture(s): arm64
- Platform(s): kvm
- Application(s): All

Additional configuration

This PR introduces `CONFIG_ARM64_FEAT_PAUTH` to enable Pointer Authentication support.

build: Option to compile as PIE #239

Open danield20 wants to merge 1 commit into `unikraft:staging` from `danield20:ddinca/build-as-PIE`

danield20 commented on Jun 25, 2021 Member

This patch adds the option to compile the unikernel as
a position-independent executable so we can have ASLR.
If the unikernel is compiled as PIE, it cannot run on it's
own. A bootloader will be needed that will come in a future PR.

Signed-off-by: Daniel Dinca dincadaniel97@gmail.com

👍 1

In the Works: Testing, Fuzz Testing

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

In the Works: Testing, Fuzz Testing

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

Production-grade testing =

- Continuous testing (CI/CD, test suite)
- Destructive testing (Fuzzing)

In the Works: Testing, Fuzz Testing

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

Production-grade testing =

- Continuous testing (CI/CD, test suite)
- Destructive testing (Fuzzing)

Significant efforts on continuous testing:

In the Works: Testing, Fuzz Testing

Good Progress

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

Production-grade testing =

- Continuous testing (CI/CD, test suite)
- Destructive testing (Fuzzing)

Concourse

Significant efforts on continuous testing:

- CI/CD pipeline tests patches systematically (Concourse)
- Application-level tests but also kernel unit-tests (`uktest`)

A. Jung @ FOSDEM'22
https://fosdem.org/2022/schedule/event/massive_unikernel_matrices_with_unikraft_concourse_and_more/

In the Works: Testing, Fuzz Testing

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

Ongoing efforts on fuzzing as well

In the Works: Testing, Fuzz Testing

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

Ongoing efforts on fuzzing as well

- Not entirely trivial, as most OS fuzzers are **tailored for Linux**

In the Works: Testing, Fuzz Testing

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

Ongoing efforts on fuzzing as well

- Not entirely trivial, as most OS fuzzers are **tailored for Linux**

Not "just" a matter of porting Syzkaller to Unikraft

In the Works: Testing, Fuzz Testing

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

Ongoing efforts on fuzzing as well

- Not entirely trivial, as most OS fuzzers are **tailored for Linux**
  - Coverage measurement (no Kcov, porting to gcov not without changes)
  - Not every system call is fully implemented

Not "just" a matter of porting Syzkaller to Unikraft

In the Works: Testing, Fuzz Testing

To go mainstream, unikernels need not only hardening that's in line with mainstream OSes, but also **production-grade testing**

Ongoing efforts on fuzzing as well

- Not entirely trivial, as most OS fuzzers are **tailored for Linux**
  - Coverage measurement (no Kcov, porting to gcov not without changes)
  - Not every system call is fully implemented
- How does unikernel fuzzing impact the architecture of fuzzers?
- How to design a fuzzer that's **ready to "plug and play" in any POSIX OS**?

Not "just" a matter of porting Syzkaller to Unikraft

In the Works: Automatic Specialization

Another end of the "testing" topic: how do you determine **how good a configuration really is**?

A. Jung et al. @ APSys'21 https://www.youtube.com/watch?v=YLf86gcHW4E

In the Works: Automatic Specialization

Another end of the "testing" topic: how do you determine **how good a configuration really is**?

Unikernels pitch specialization:

- Your best Nginx configuration is not your best SQLite configuration
- ...and probably not your best Redis configuration either 🙂

A. Jung et al. @ APSys'21 https://www.youtube.com/watch?v=YLf86gcHW4E

In the Works: Automatic Specialization

Another end of the "testing" topic: how do you determine **how good a configuration really is**?

Unikernels pitch specialization:

- Your best Nginx configuration is not your best SQLite configuration
- ...and probably not your best Redis configuration either 🙂

The number of possible configurations: astronomical scale

A. Jung et al. @ APSys'21 https://www.youtube.com/watch?v=YLf86gcHW4E

In the Works: Automatic Specialization

Another end of the "testing" topic: how do you determine **how good a configuration really is**?

Unikernels pitch specialization:

- Your best Nginx configuration is not your best SQLite configuration
- ...and probably not your best Redis configuration either 🙂

The number of possible configurations: astronomical scale

- Small subset of configuration options (Nginx) ~ 10^{13}
- How do you explore this? Can you use optimization algorithms? ML?

A. Jung et al. @ APSys'21 https://www.youtube.com/watch?v=YLf86gcHW4E

In the Works: Compartmentalization

In the Works: Compartmentalization

Traditional understanding of unikernels:

Unikernel = one single trust domain (kernel + application)

(Used to) make sense.

In the Works: Compartmentalization

Traditional understanding of unikernels:

Unikernel = one single trust domain (kernel + application)

(Used to) make sense.

Certain applications are large, with heterogeneous components: trust, safety, properties, requirements...

In the Works: Compartmentalization

Traditional understanding of unikernels:

Unikernel = one single trust domain (kernel + application)

(Used to) make sense.

Certain applications are large, with heterogeneous components: trust, safety, properties, requirements...

And at the same time we see (re-)appearing a lot of lightweight isolation mechanisms (protection keys, HW capabilities, SFI, etc.)

In the Works: Compartmentalization

There is an opportunity to use these mechanisms to make unikernels even safer without yielding their benefits!

In the Works: Compartmentalization

There is an opportunity to use these mechanisms to make unikernels even safer without yielding their benefits!

This is what initially motivated our work FlexOS: can we reconcile unikernels/libOSes with isolation to obtain a new OS model that offers not only specialization **towards performance, but also towards safety**?

H. Lefeuvre et al. @ ASPLOS'22, come to our talk Thursday morning!
(also @ FOSDEM'22 https://fosdem.org/2022/schedule/event/tee_flexos/)

In the Works: Compartmentalization

There is an opportunity to use these mechanisms to make unikernels even safer without yielding their benefits!

This is what initially motivated our work FlexOS: can we reconcile unikernels/libOSes with isolation to obtain a new OS model that offers not only specialization **towards performance, but also towards safety**?

Other groups explored this direction: CubicleOS (also ASPLOS, 2021). Explore intra-unikernel isolation with Intel MPK.

H. Lefeuvre et al. @ ASPLOS'22, come to our talk Thursday morning!
(also @ FOSDEM'22 https://fosdem.org/2022/schedule/event/tee_flexos/)

How about Compatibility?

Compatibility: a Solved Problem?

Historical unikernels: hand-written for an application (ClickOS).

Compatibility: a Solved Problem?

Historical unikernels: hand-written for an application (ClickOS).

Terrible Compatibility

Compatibility: a Solved Problem?

Historical unikernels: hand-written for an application (ClickOS).

Terrible Compatibility

But we're not in the 2010s anymore, Unikraft aims at Linux/POSIX compliance.

Compatibility: a Solved Problem?

Historical unikernels: hand-written for an application (ClickOS).

Terrible Compatibility

But we're not in the 2010s anymore, Unikraft aims at Linux/POSIX compliance.

Growing number of supported system calls: now 170+

As a point of comparison, Graphene (Gramine) also supports about ~170

Graphene/Gramine: https://gramineproject.io/

Compatibility: a Solved Problem?

Historical unikernels: hand-written for an application (ClickOS).

Terrible Compatibility

But we're not in the 2010s anymore, Unikraft aims at Linux/POSIX compliance.

Growing number of supported system calls: now 170+
As a point of comparison, Graphene (Gramine) also supports about ~170

To be clear: Unikraft is neither fully Linux compatible, nor fully POSIX compliant!

- The good old `fork()` problem
- Not all system calls are fully implemented

But **do you really need to be fully compatible** to be useful?

Graphene/Gramine: https://gramineproject.io/

Compatibility: a Solved Problem?

But **do you really need to be fully compatible** to be useful?

Many applications that use unsupported features tend to be bad candidates for unikernelization anyways (system admin tools, heavily multiprocess apps that cannot use threads).

H. Lefeuvre et al. @ USENIX ;login
https://www.usenix.org/publications/loginonline/unikraft-and-coming-age-unikernels

Compatibility: a Solved Problem?

But **do you really need to be fully compatible** to be useful?

Many applications that use unsupported features tend to be bad candidates for unikernelization anyways (system admin tools, heavily multiprocess apps that cannot use threads).

For the rest, partial compatibility is just fine if porting is a reasonable task.

And in 2022, it is.

H. Lefeuvre et al. @ USENIX ;login
https://www.usenix.org/publications/loginonline/unikraft-and-coming-age-unikernels

Compatibility: a Solved Problem?

But **do you really need to be fully compatible** to be useful?

Many applications that use unsupported features tend to be bad candidates for unikernelization anyways (system admin tools, heavily multiprocess apps that cannot use threads).

For the rest, partial compatibility is just fine if porting is a reasonable task.

And in 2022, it is.

Given the benefits of Unikraft, a week of porting is a minor annoyance.
All you need is a good application test-suite (but you have one, right? 🙂)

H. Lefeuvre et al. @ USENIX ;login
https://www.usenix.org/publications/loginonline/unikraft-and-coming-age-unikernels

Conclusion

In a Nutshell

In the last decade we've seen a lot of unikernels come and go, but **none that managed to**:

In a Nutshell

In the last decade we've seen a lot of unikernels come and go, but **none that managed to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows

In a Nutshell

In the last decade we've seen a lot of unikernels come and go, but **none that managed to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows
- take security seriously
- meet production-grade testing standards
- good debuggability

In a Nutshell

In the last decade we've seen a lot of unikernels come and go, but **none that managed to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows
- take security seriously
- meet production-grade testing standards
- good debuggability

And that's fine, because they were just **research projects**

In a Nutshell

In the last decade we've seen a lot of unikernels come and go, but **none that managed to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows
- take security seriously
- meet production-grade testing standards
- good debuggability

And that's fine, because they were just **research projects**

Now, if we want to see unikernels in production one day, one project must manage it. **We want this project to be Unikraft.**

In a Nutshell

In the last decade we've seen a lot of unikernels come and go, but **none that managed to**:

- provide good compatibility
- excellent performance
- transparent integration in major deployment workflows
- take security seriously
- meet production-grade testing standards
- good debuggability

And that's fine, because they were just **research projects**

Now, if we want to see unikernels in production one day, one project must manage it. **We want this project to be Unikraft.**

Unikraft is not there yet. But it's progressing, and we hope to see it reaching full maturity in the coming year

In a Nutshell

Over time, doing all this engineering proved **fruitful on the research side**

In a Nutshell

Over time, doing all this engineering proved **fruitful on the research side**

Unikraft turns out to be an **excellent substrate for top-tier publications**

It's open, small, clean, modular, fast to experiment with, such
that undergrads and grad students quickly get to understand it

In a Nutshell

Over time, doing all this engineering proved **fruitful on the research side**

Unikraft turns out to be an **excellent substrate for top-tier publications**

It's open, small, clean, modular, fast to experiment with, such
that undergrads and grad students quickly get to understand it

Probably one of the good examples where starting clean-slate pays out in the long run

In a Nutshell

Over time, doing all this engineering proved **fruitful on the research side**

Unikraft turns out to be an **excellent substrate for top-tier publications**

It's open, small, clean, modular, fast to experiment with, such
that undergrads and grad students quickly get to understand it

Probably one of the good examples where starting clean-slate pays out in the long run

What will the broader systems community build with Unikraft?

Pushing Unikernels to Production!

Unikraft Community: https://unikraft.org/
Unikraft Cloud: https://unikraft.io/
Code: https://github.com/unikraft