Verifying security invariants in *ExpressOS*

Haohui Mai, Edgar Pek, Hui Xue,
Samuel T. King, P. Madhusudan

University of Illinois at Urbana-Champaign

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Mobiles devices are powerful

citibank

Gmail
by Google

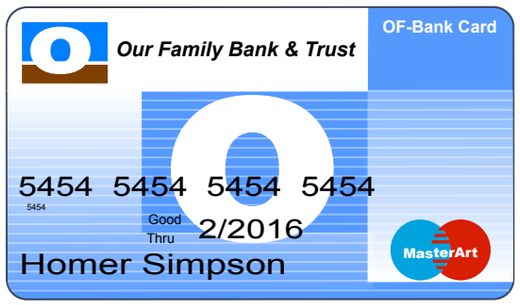
PayPal™

Security of mobile devices is important

- High value targets on mobile devices

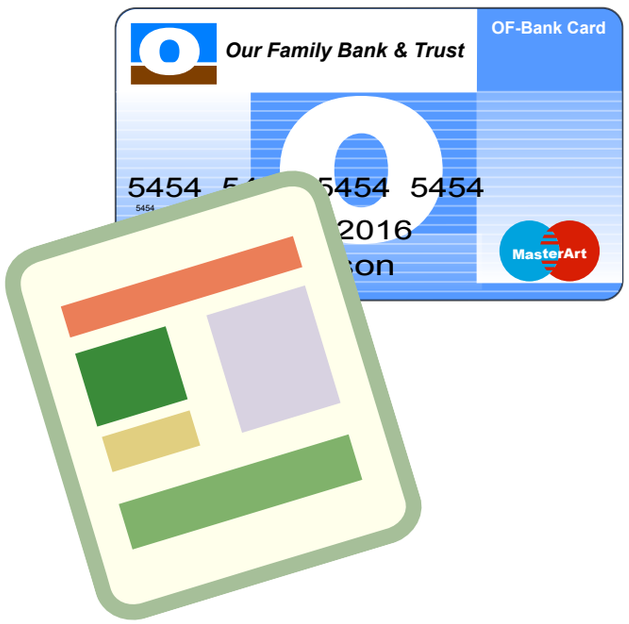
Security of mobile devices is important

- High value targets on mobile devices



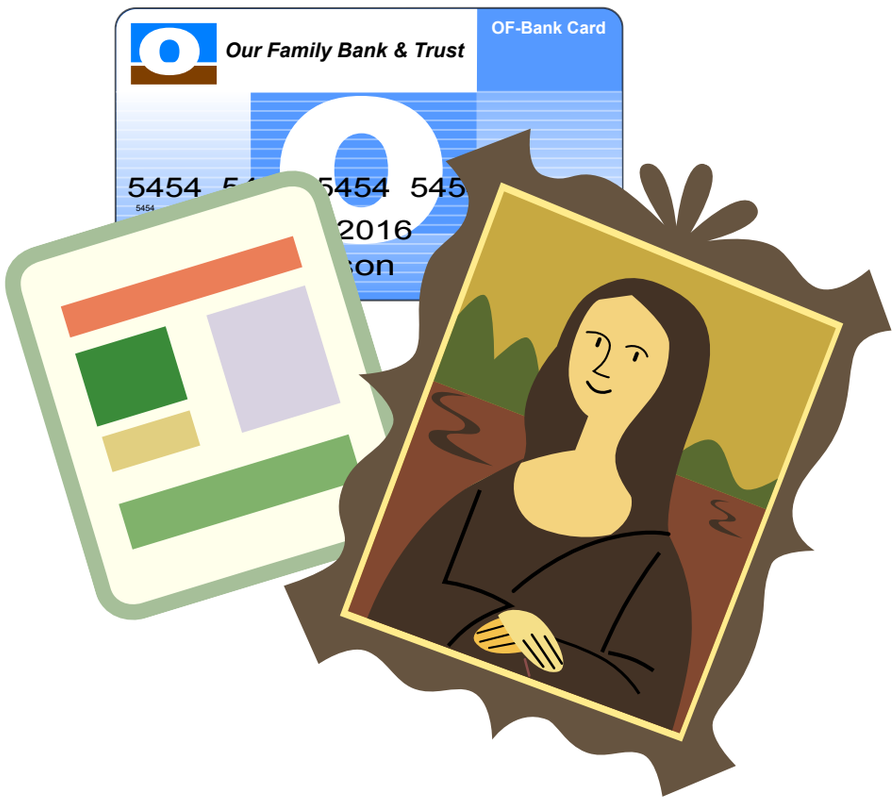
Security of mobile devices is important

- High value targets on mobile devices



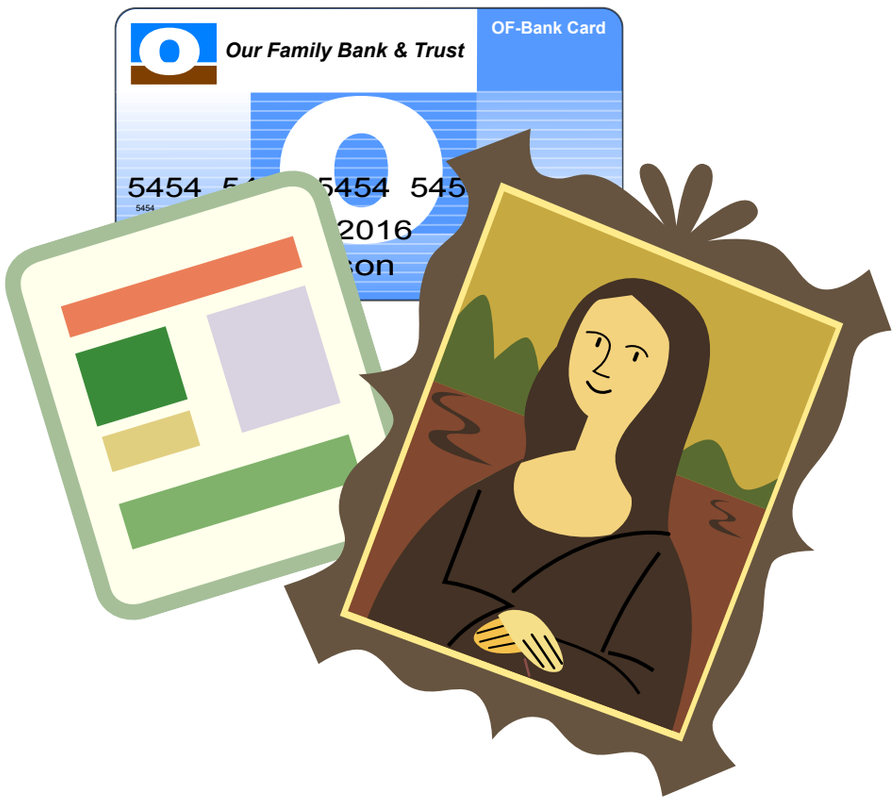
Security of mobile devices is important

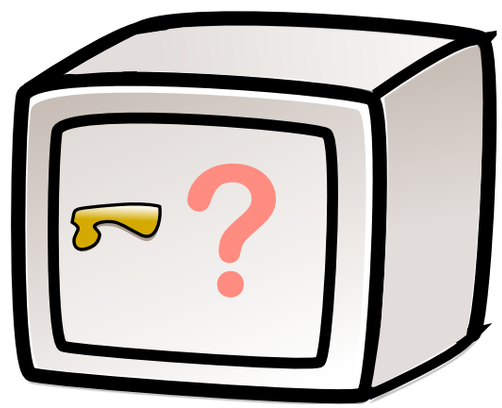
- High value targets on mobile devices



Security of mobile devices is important

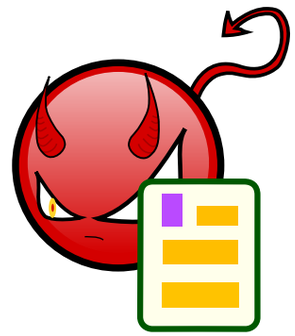
- High value targets on mobile devices





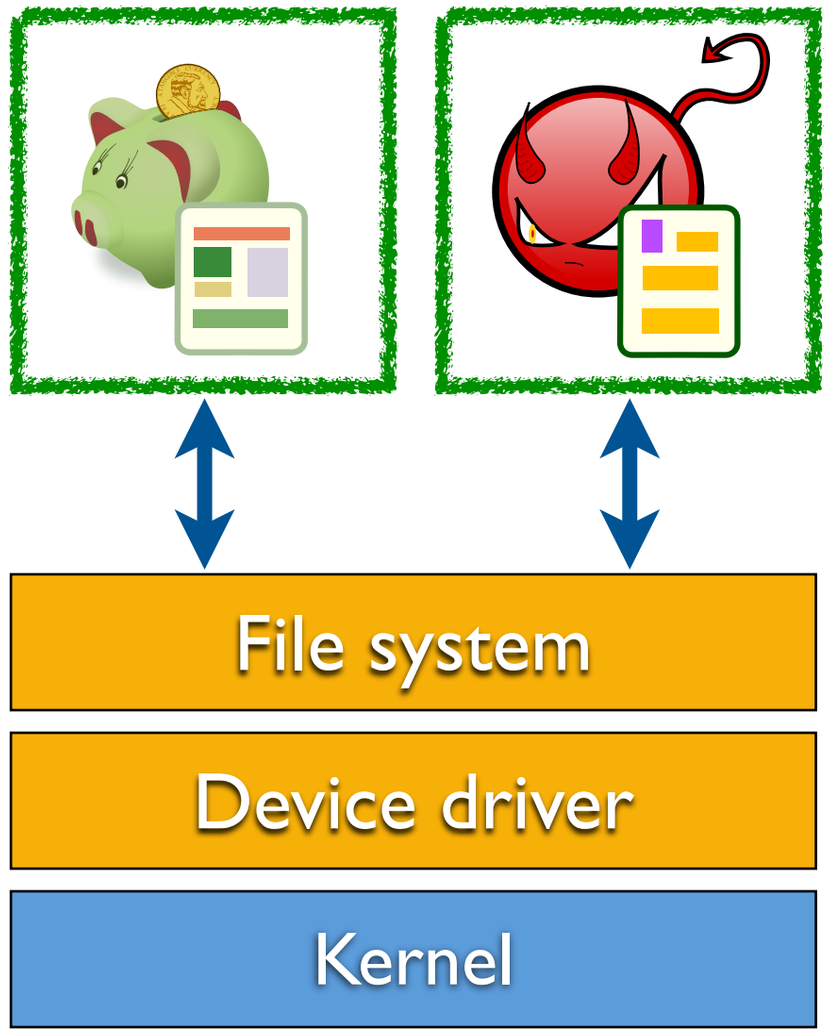
Motivating example

Motivating example



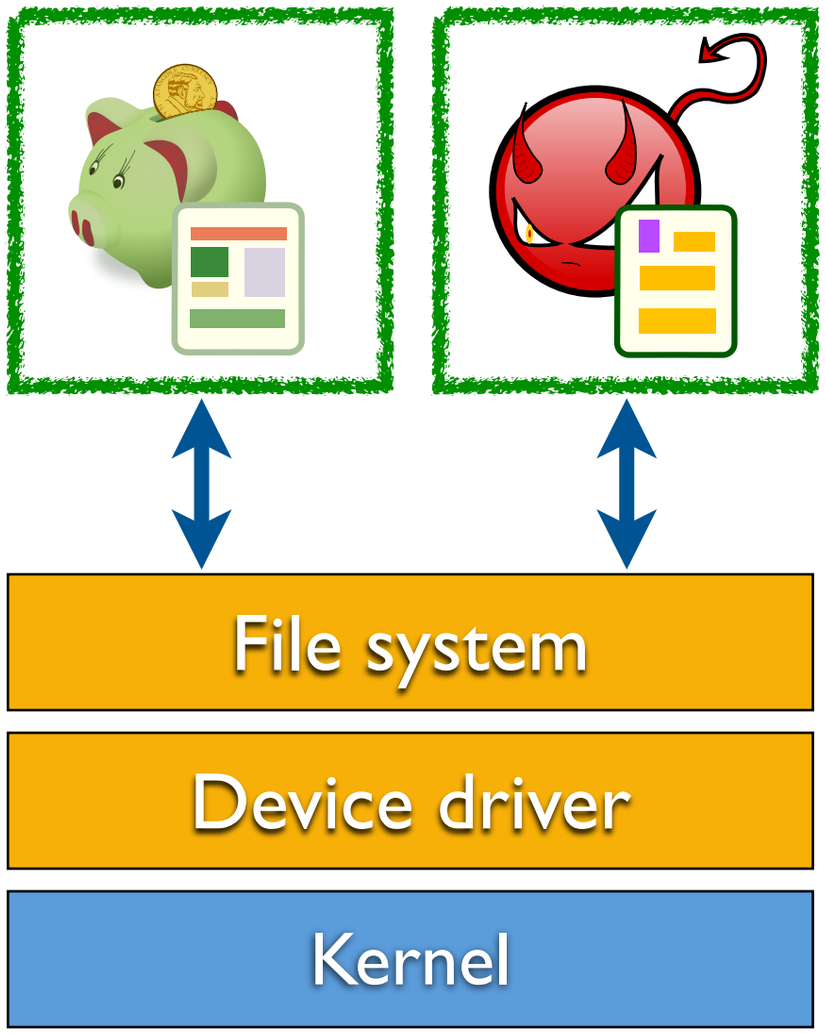
Motivating example

- Isolate the application's persistent storage from
  - other applications
  - components of the system



Motivating example

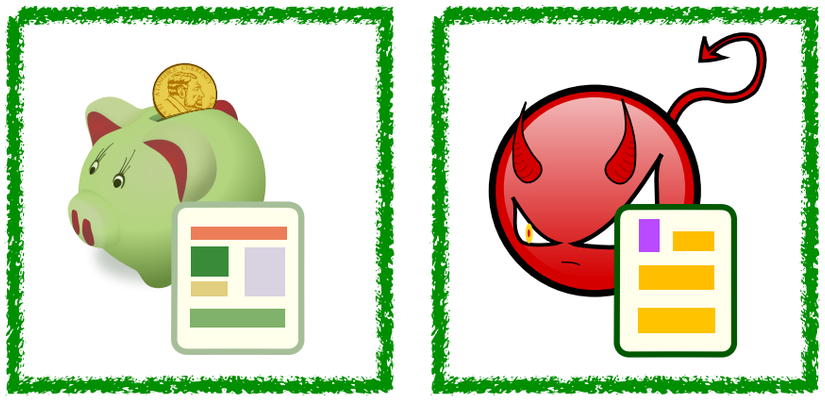
- Isolate the application's persistent storage from
  - other applications
  - components of the system
- Immediately meaningful




ROOTED


09:44
86 %
JailbreakMe by comex (et al.)
Jailbreak to get tweaks and apps Apple won't allow in the App Store.
Free, legal, safe.
You should sync with iTunes before using this tool.
More Info »
slide to jailbreak

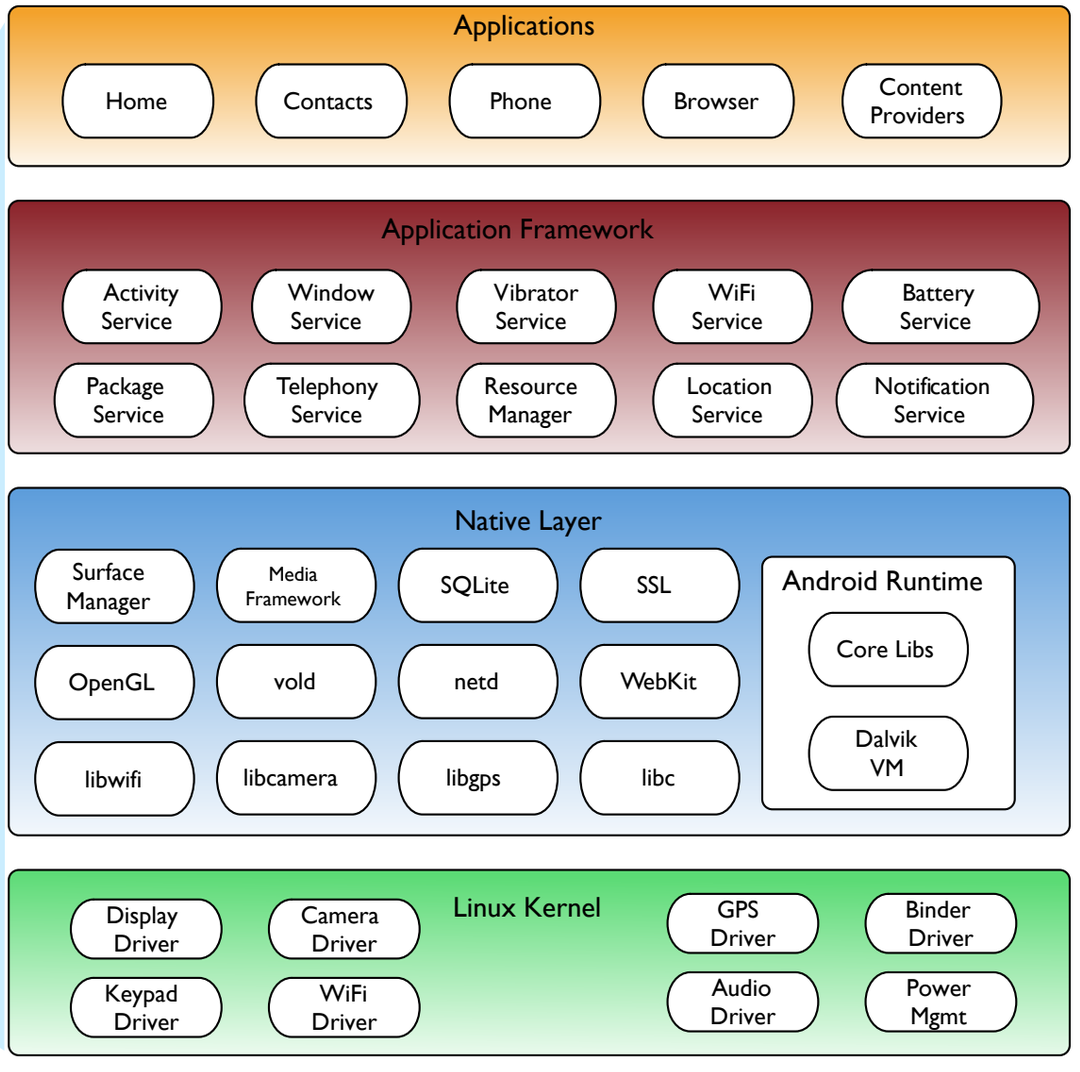
Wide attack surfaces



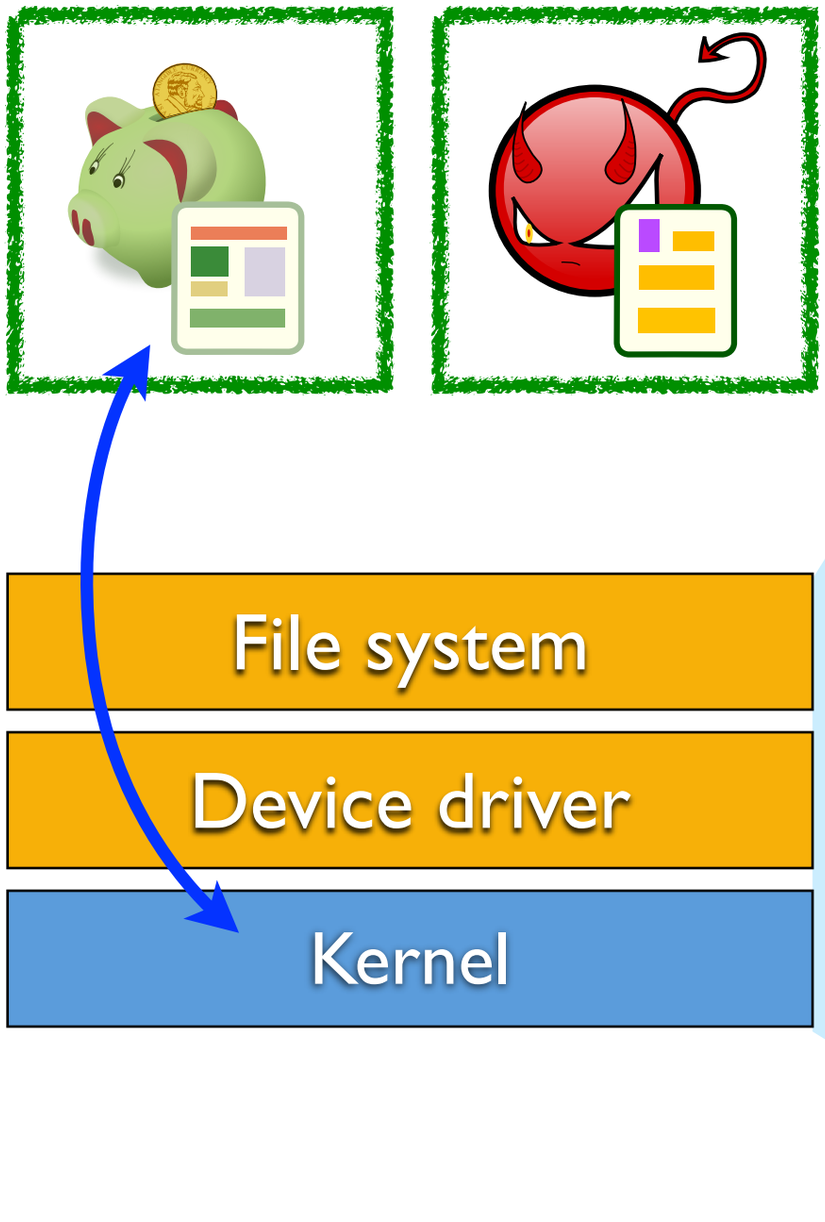
File system

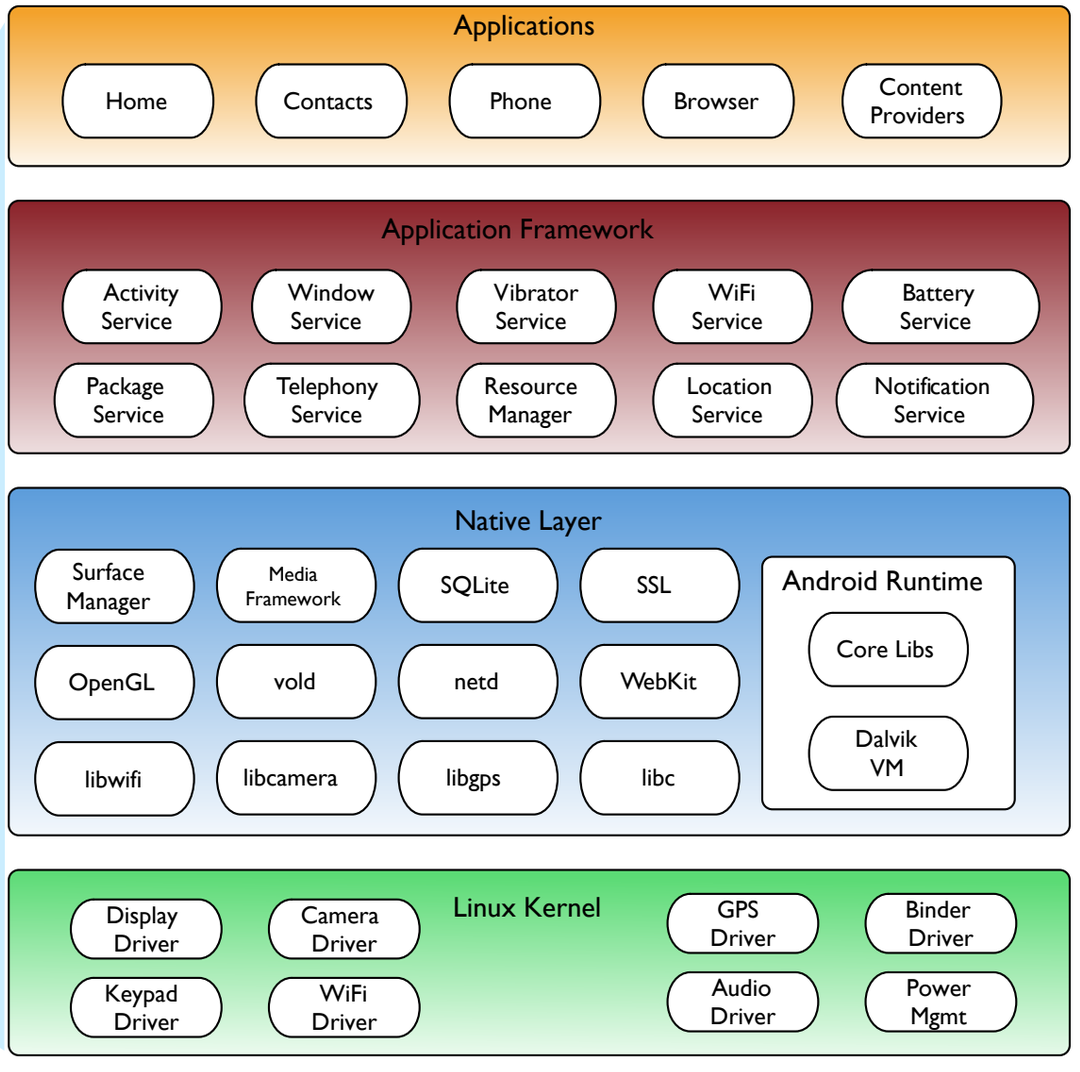
Device driver

Kernel

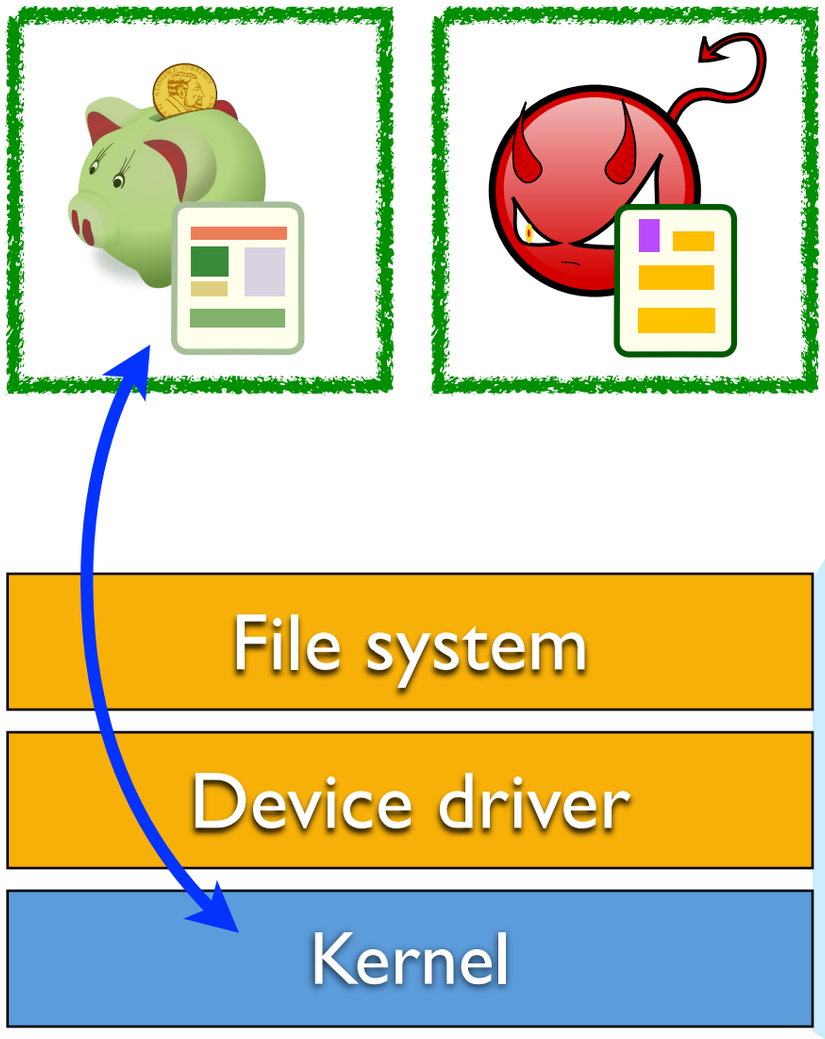


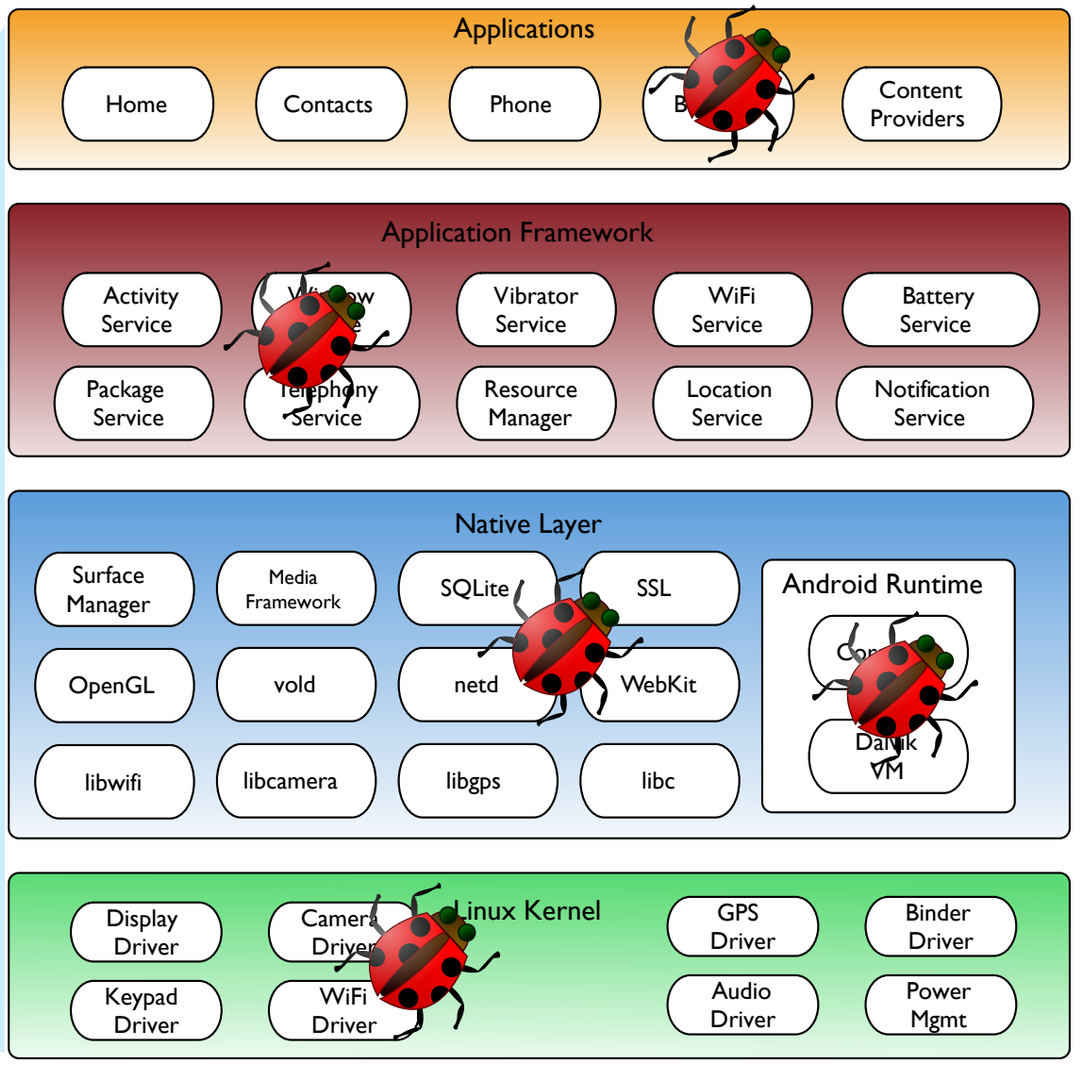
Wide attack surfaces



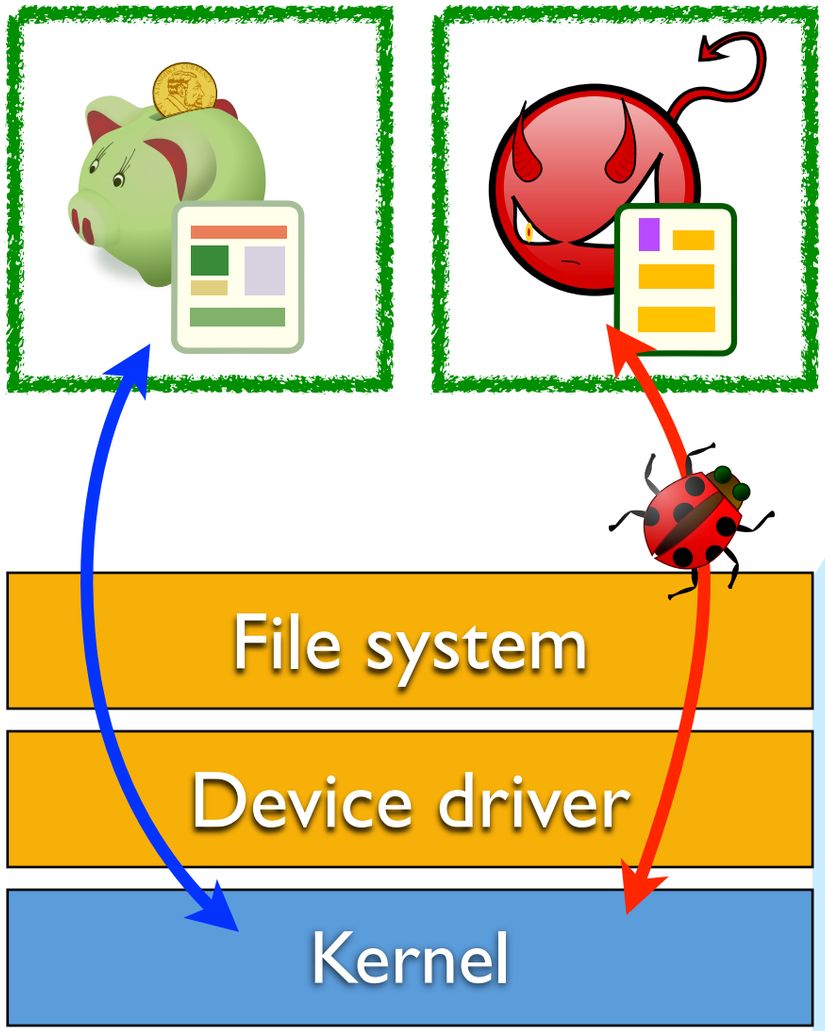


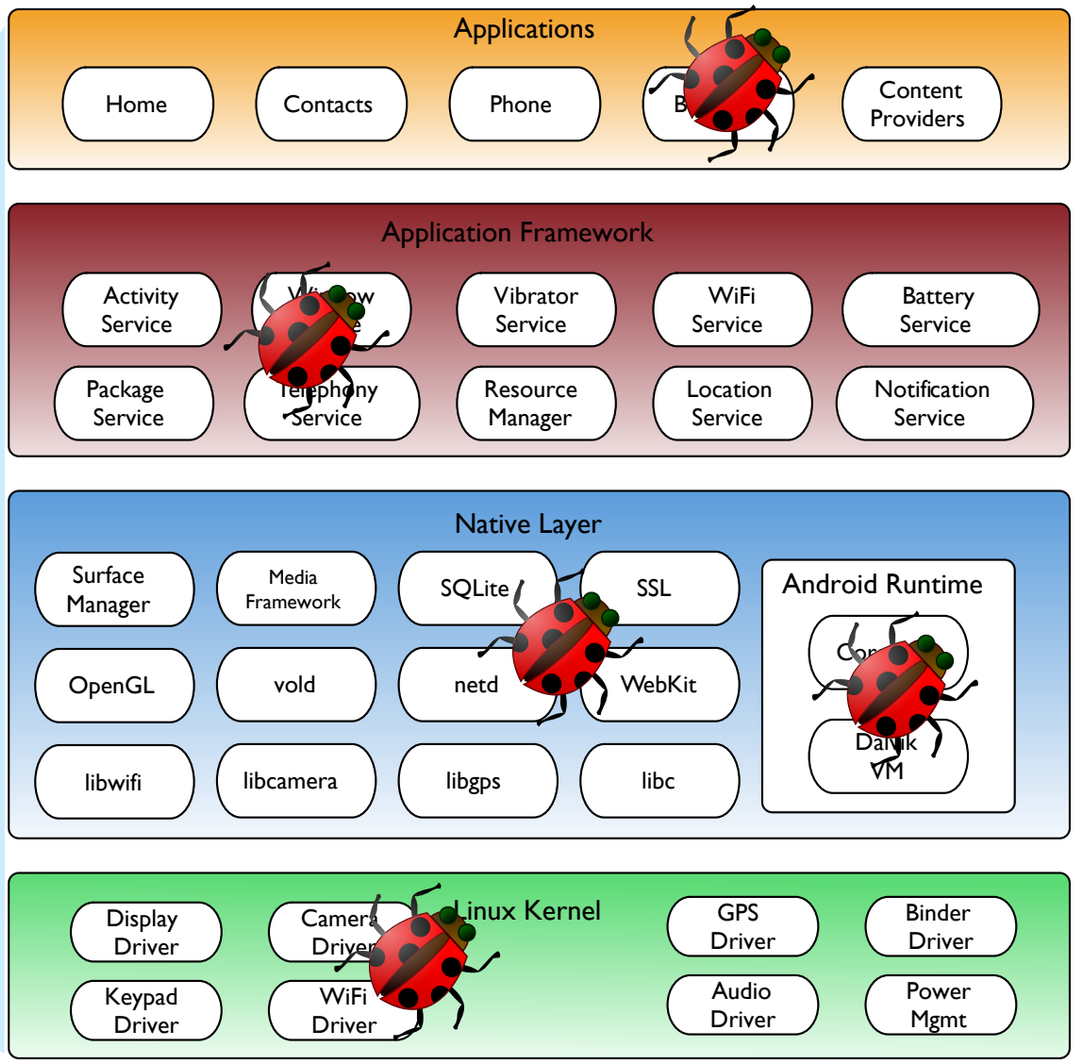
Wide attack surfaces



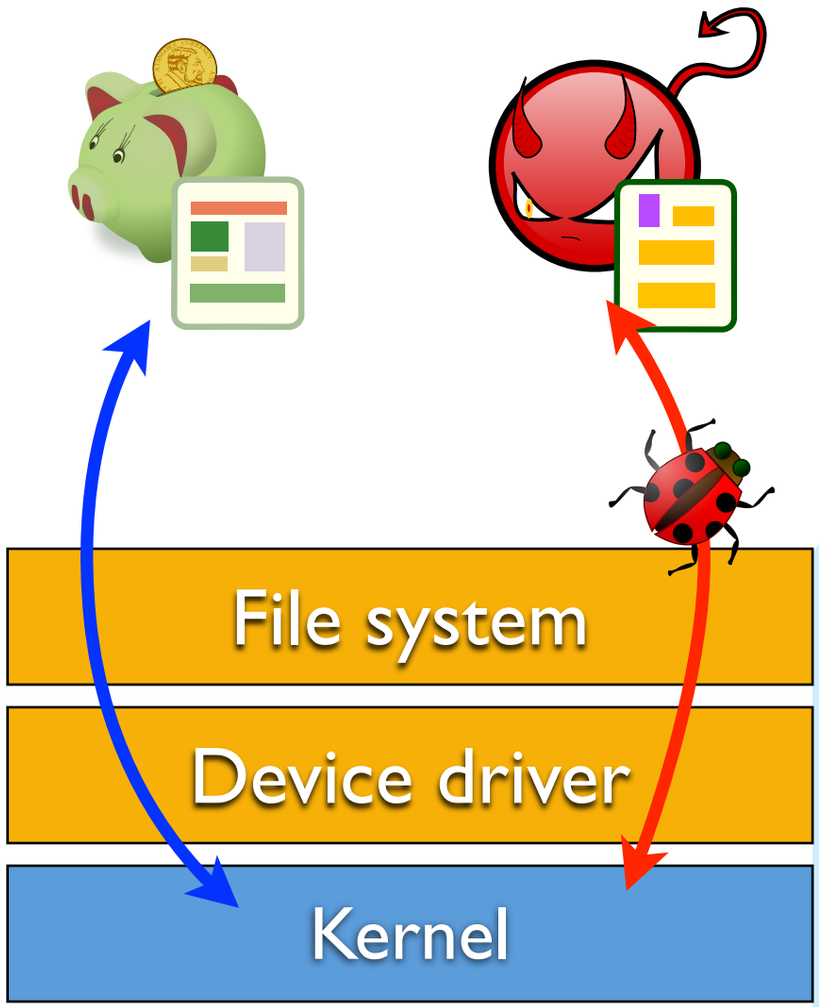


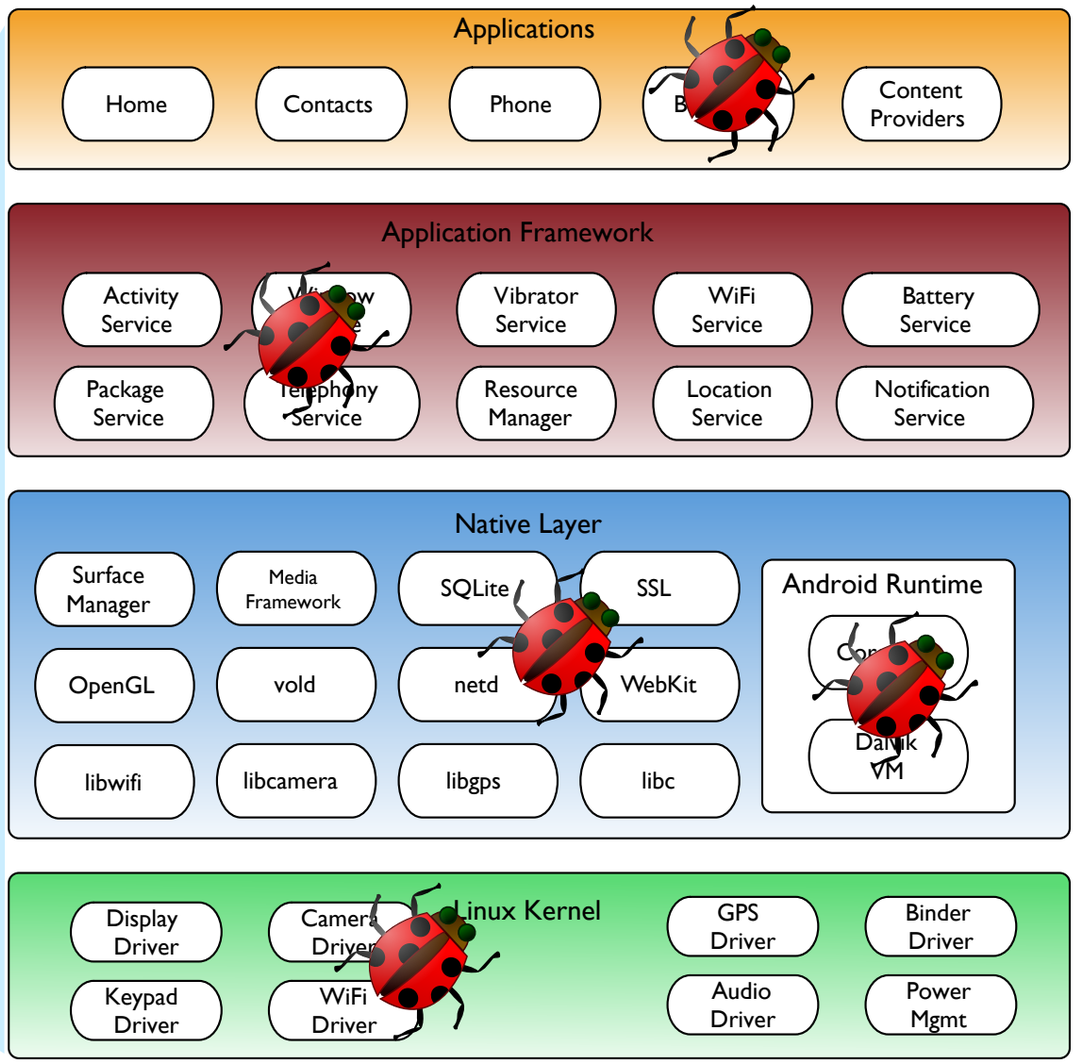
Wide attack surfaces



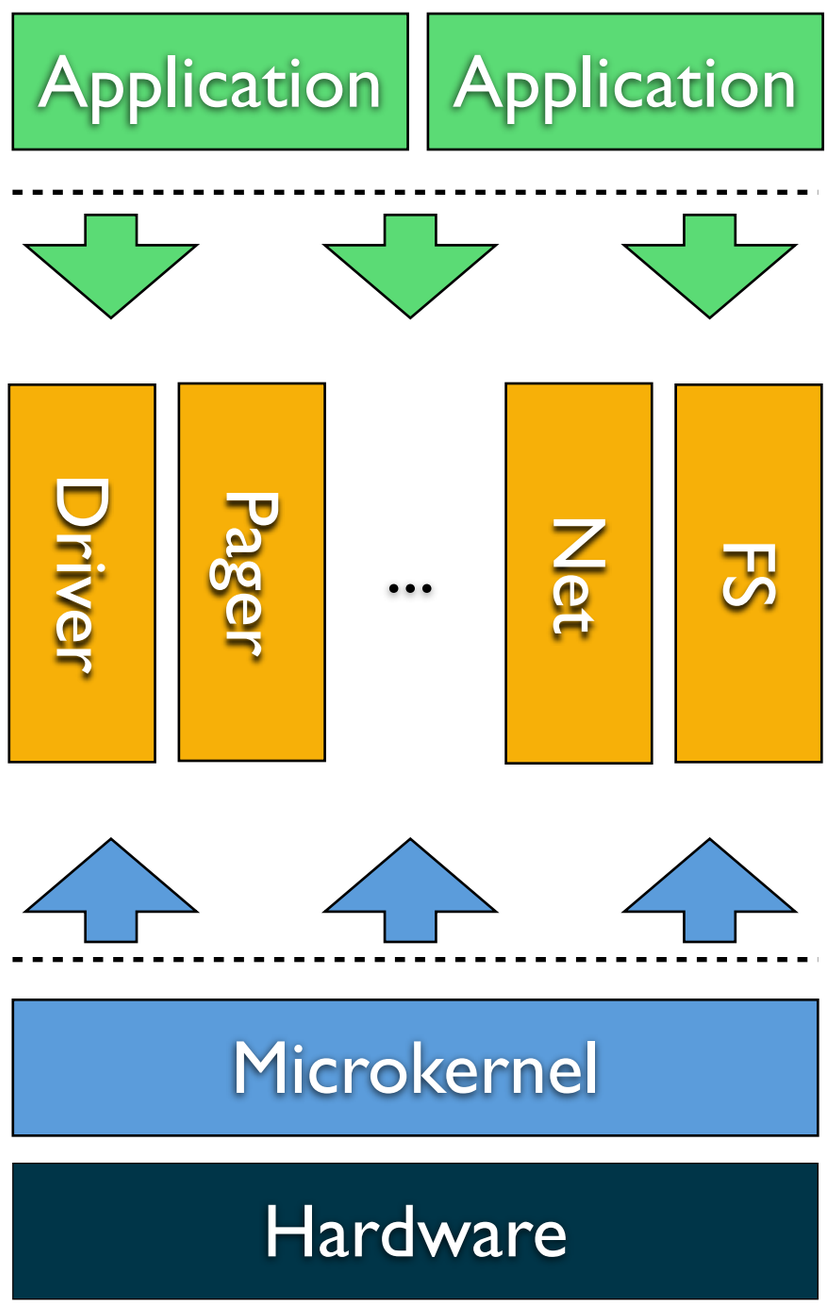


Wide attack surfaces



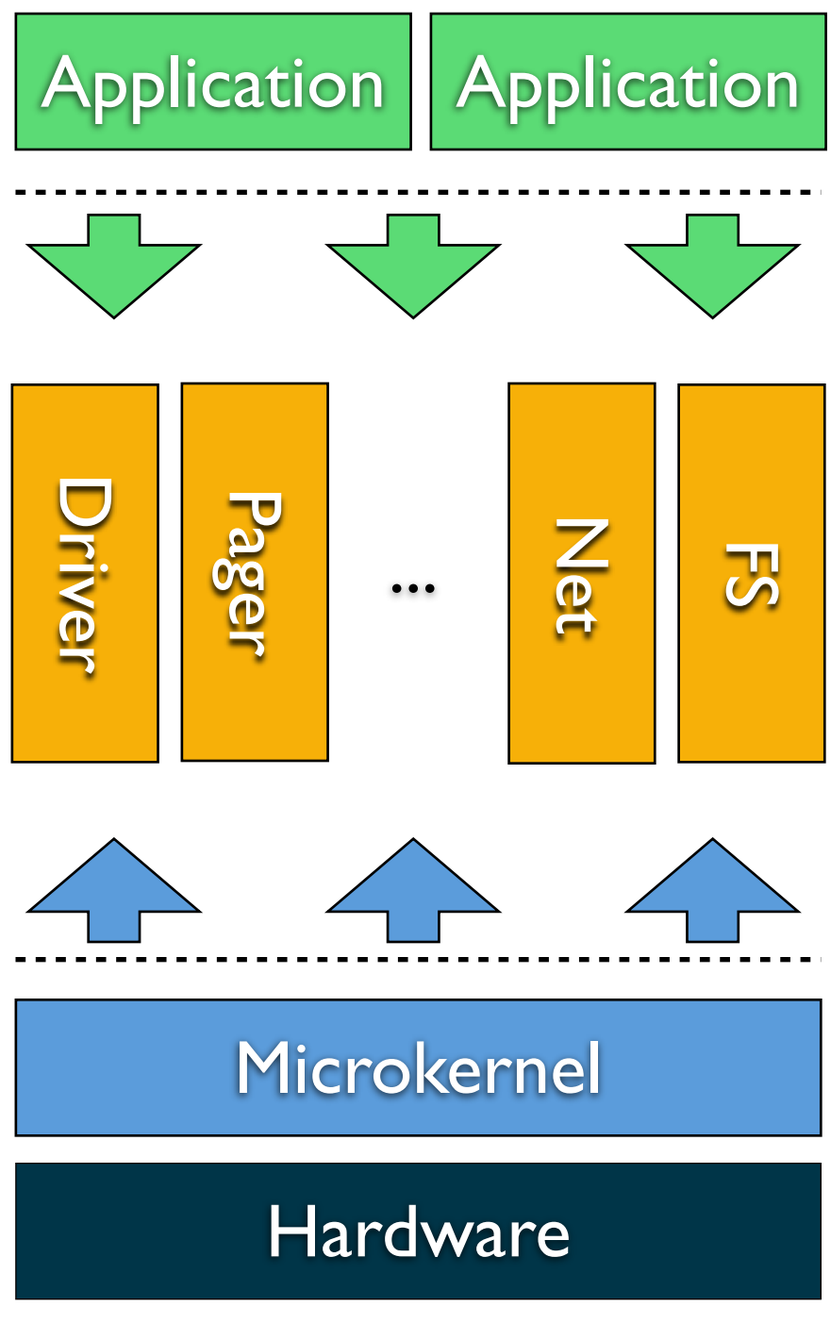


Microkernel is not enough



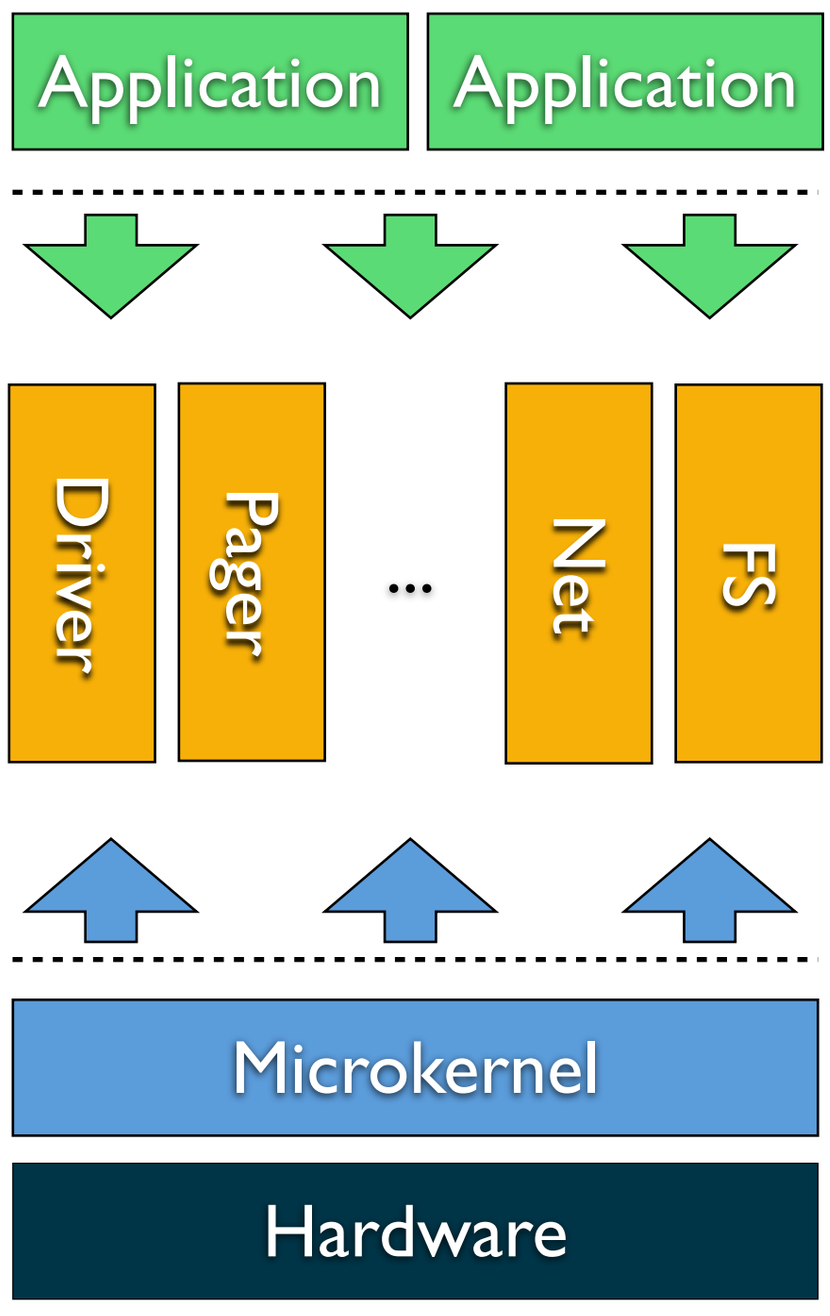
Microkernel is not enough

- Low-level abstractions
- v.s. application semantics



Microkernel is not enough

- Low-level abstractions
  - v.s. application semantics
- Shared services



ExpressOS: a high assurance OS that runs Android apps

ExpressOS: a high assurance OS that runs Android apps

- Capture application-level security requirements as *security invariants*

ExpressOS: a high assurance OS that runs Android apps

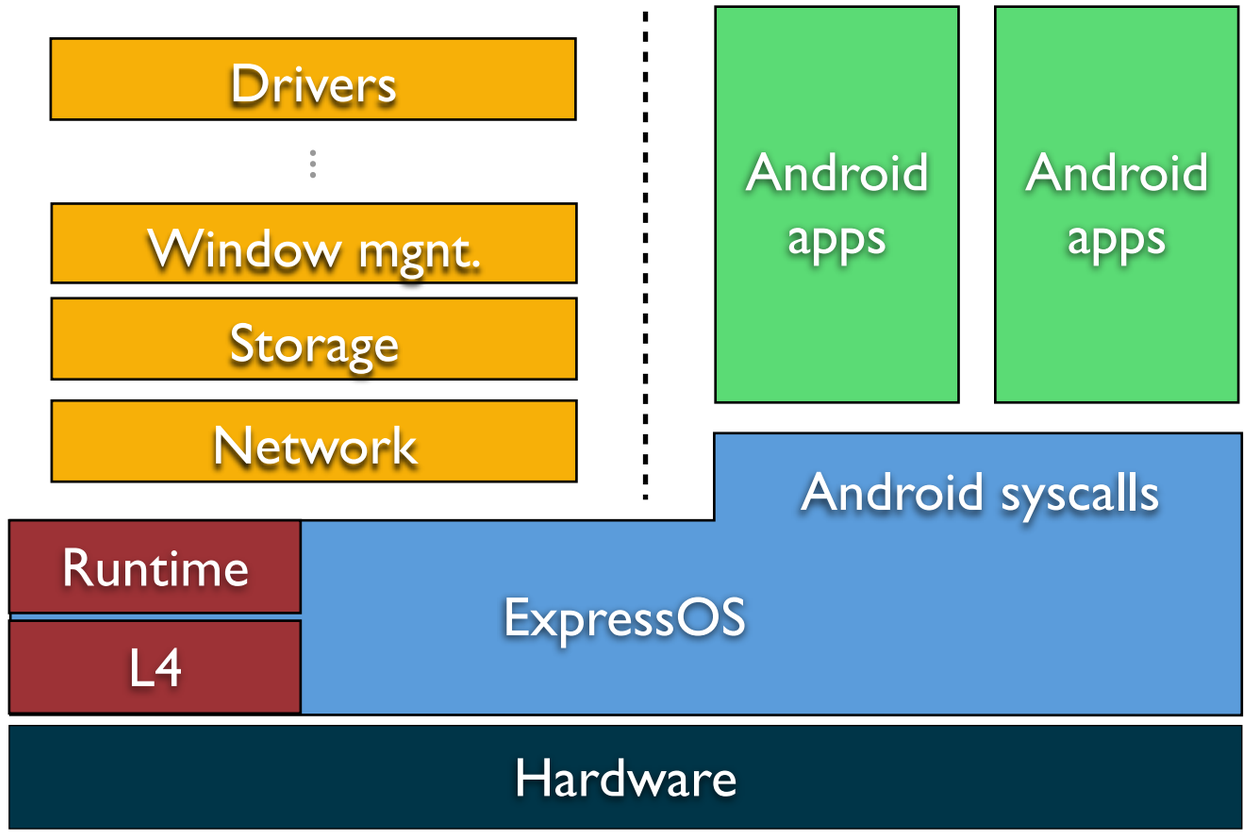
- Capture application-level security requirements as *security invariants*
- Verify security invariants *directly*
  - *NOT* from full functional correctness

ExpressOS: a high assurance OS that runs Android apps

- Capture application-level security requirements as *security invariants*
- Verify security invariants *directly*
  - *NOT* from full functional correctness
- Formally verified security guarantees with *reasonable* verification effort

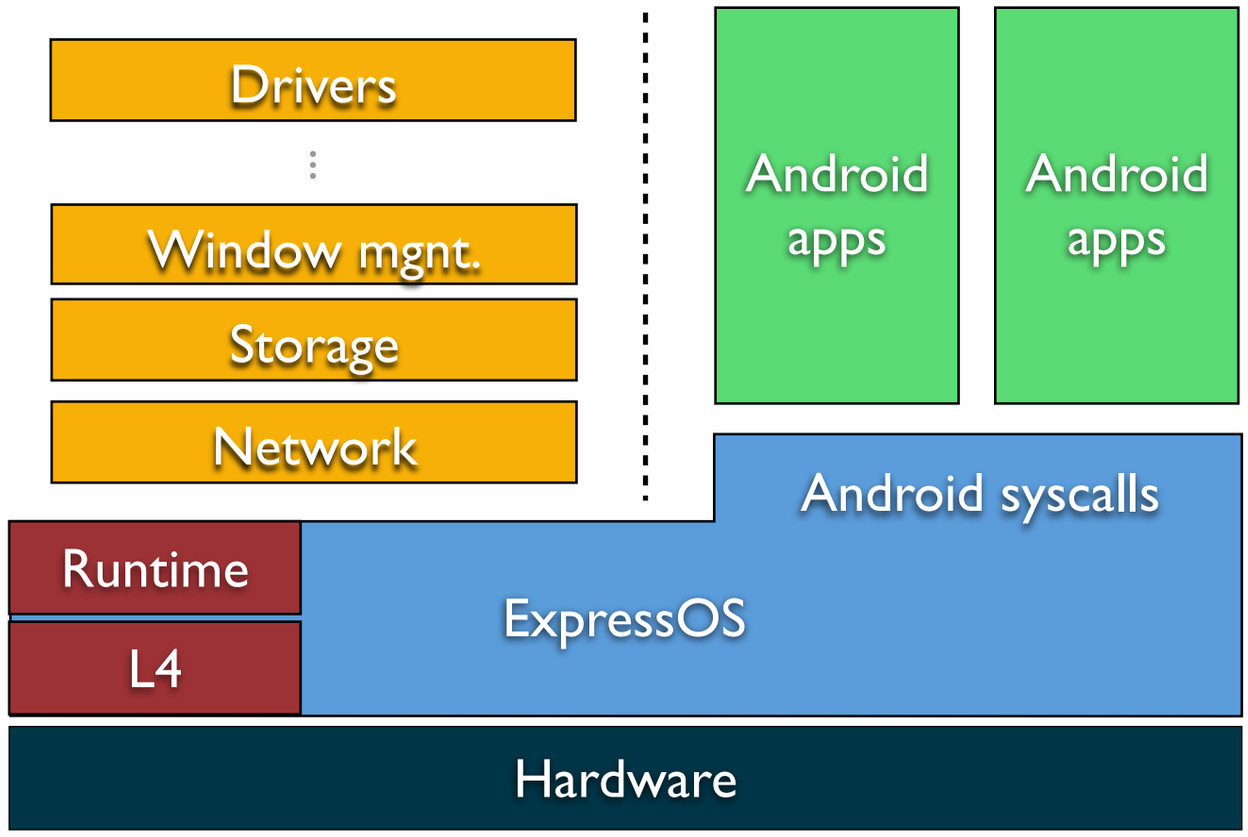
- Introduction
- Verifying security invariants
- Experience
- Conclusion

Design tailored for verification



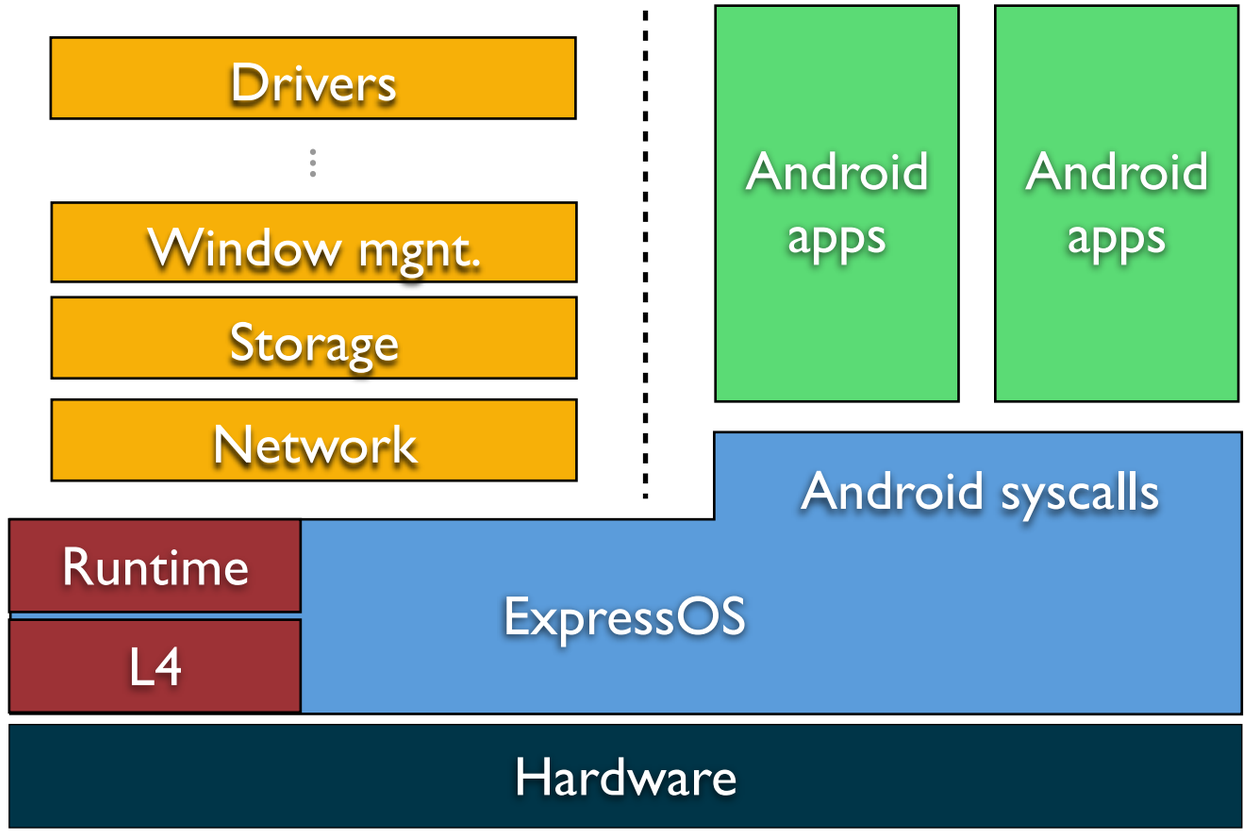
Design tailored for verification

- Understand application semantics
  - Provides Android/Linux-like syscalls

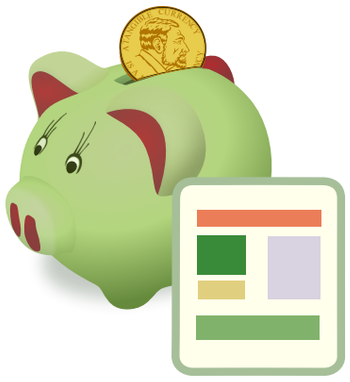


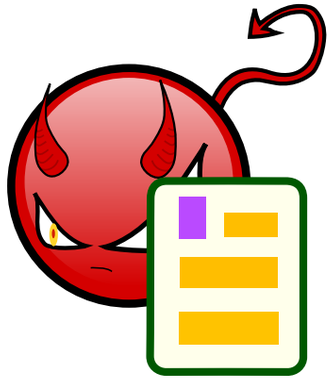
Design tailored for verification

- Understand application semantics
  - Provides Android/Linux-like syscalls
- Bugs in unverified components cannot subvert the security invariants
  - Microkernel
  - Type safety (C# + Dafny)



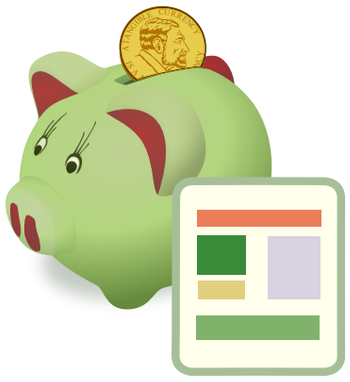
Security invariants

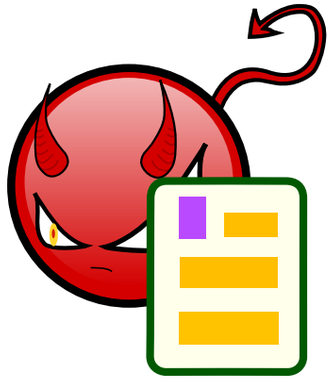




- Secure storage
- Memory isolation
- UI isolation
- Secure IPC

Security invariants

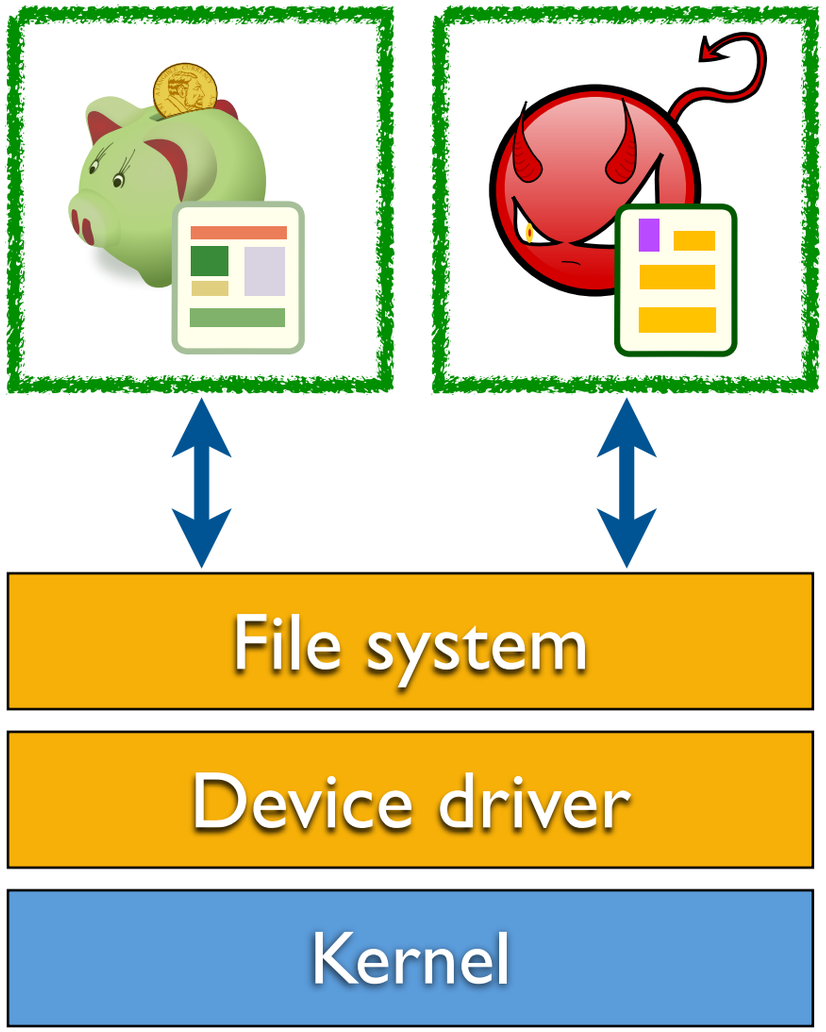




- Secure storage
- Memory isolation
- UI isolation
- Secure IPC

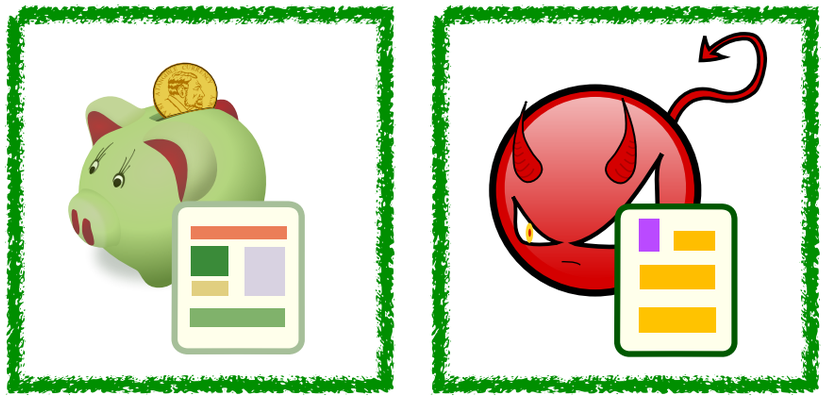
Secure storage

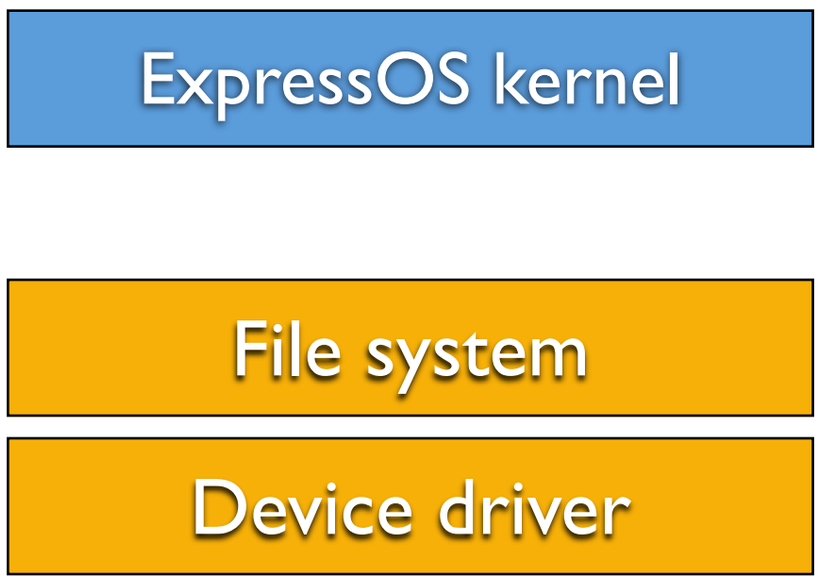
- The persistent storage of an application should be isolated from other applications and the system



Secure storage

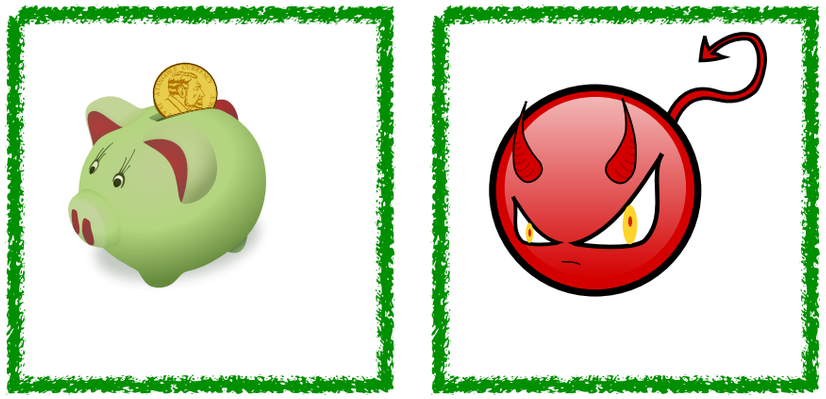
- ExpressOS kernel provides file APIs to applications





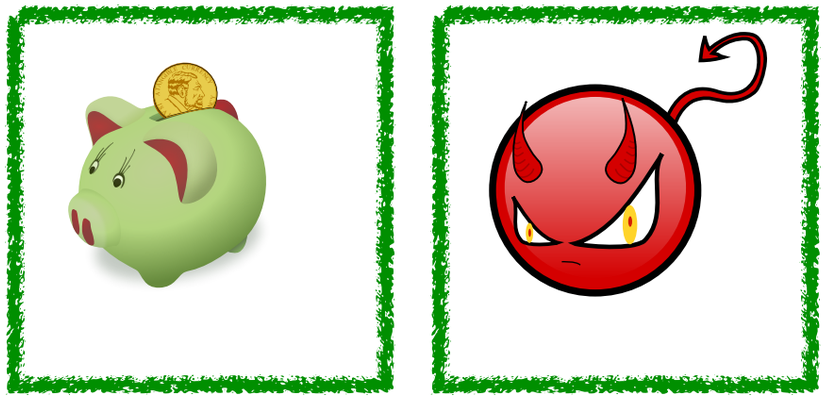
Secure storage

- ExpressOS kernel provides file APIs to applications
- Enforce confidentiality and integrity using HMAC and encryption
- *Verify* these primitives are used correctly



Secure storage

- ExpressOS kernel provides file APIs to applications
- Enforce confidentiality and integrity using HMAC and encryption
- *Verify* these primitives are used correctly
- No trusts on FS / drivers



Assumptions

Assumptions

- The implementation of cryptographic algorithms

Assumptions

- The implementation of cryptographic algorithms
- L4 and the language run-time

Assumptions

- The implementation of cryptographic algorithms
- L4 and the language run-time
- Correct specifications

Assumptions

- The implementation of cryptographic algorithms
- L4 and the language run-time
- Correct specifications

- Do not cover covert channels

Tools for verification

Tools for verification

	Code Contracts	Dafny
Power	*Restricted* (abstract interpretation)	*Expressive* (SMT solvers like Z3)
Annota-tion burden	*Low* (~0.01 lines per LOC)	*High* (5~6 lines per LOC)

Tools for verification

- Dafny only

  ✗ Too much work

	Code Contracts	Dafny
Power	*Restricted* (abstract interpretation)	*Expressive* (SMT solvers like Z3)
Annota-tion burden	*Low* (~0.01 lines per LOC)	*High* (5~6 lines per LOC)

Tools for verification

- Dafny only

  X Too much work

- Code contract(CC) only

  X Incomplete

	Code Contracts	Dafny
Power	*Restricted* (abstract interpretation)	*Expressive* (SMT solvers like Z3)
Annota -tion burden	*Low* (~0.01 lines per LOC)	*High* (5~6 lines per LOC)

Tools for verification

- Dafny only

  ✗ Too much work

- Code contract(CC) only

  ✗ Incomplete

- Dafny + code contracts

  ✓ Practical

	Code Contracts	Dafny
Power	*Restricted* (abstract interpretation)	*Expressive* (SMT solvers like Z3)
Annota-tion burden	*Low* (~0.01 lines per LOC)	*High* (5~6 lines per LOC)

Example of code contracts

```
class Foo {
  int m;
  void Increment() {
    ++m;

  }
}
```

Example of code contracts

```
class Foo {
  int m;
  void Increment() {
    ++m;
  }
}
```

Assertion
Contract.Assert(...);

Example of code contracts

```
class Foo {

   int m;

   void Increment() {

      ++m;

   }

}
```

Pre-condition
Contract.Requires(m > 0);

Assertion
Contract.Assert(...);

Post-condition
Contract.Ensures(m == Contract.Old(m) + 1);

Example of Dafny

```
var Head: Foo;
void IncAll() {
  var p := Head;
  while (p != null) {
    p.Increment(); p := p.Next;
  }
}
```

Example of Dafny

```
var Head: Foo;
void IncAll() {
  var p := Head;
  while (p != null) {
    p.Increment(); p := p.Next;
  }
}
```

assert $\forall x, x \in C \rightarrow x \neq null \wedge x.m == old(x.m) + 1;$

Example of Dafny

```
var Head: Foo;
void IncAll() {
  var p := Head;
  while (p != null) {
    p.Increment(); p := p.Next;
  }
}
```

assert $\forall x, x \in C \rightarrow x \neq null \wedge x.m == old(x.m) + 1;$

Ghost code

code for verification only

Example of Dafny

```
var Head: Foo;

void IncAll() {
  var p := Head;
  while (p != null) {
    p.Increment(); p := p.Next;
  }
}
```

ghost var C :seq<Foo>;

ghost var i := 0;

invariant C[i] == p ∧
p.Next == C[i+1];....

i := i + 1;

assert ∀x, x∈C → x ≠ null ∧
x.m == old(x.m) + 1;

Ghost code

code for verification only

Checking that all data is encrypted using ghost variables

```
class FilePage {

enum State { Empty,
Authentic, Decrypted,
Encrypted }

[Ghost] State S;


void Decrypt(...) {
  Contract.Requires(S ==
State.Authentic);
  Contract.Ensures(S ==
State.Decrypted); ...

}

...
```

Checking that all data is encrypted using ghost variables

- Ghost variable S records the state of the page
- Parts of the specification

```
class FilePage {

enum State { Empty,
Authentic, Decrypted,
Encrypted }

[Ghost] State S;


void Decrypt(...) {
  Contract.Requires(S ==
State.Authentic);
  Contract.Ensures(S ==
State.Decrypted); ...

}

...
```

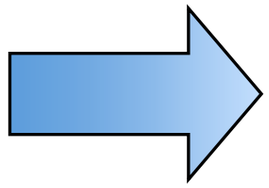
Checking that all data is encrypted using ghost variables

- Ghost variable S records the state of the page
  - Parts of the specification
- Pre-/post-conditions model the state transitions

```
class FilePage {

enum State { Empty,
Authentic, Decrypted,
Encrypted }

[Ghost] State S;


void Decrypt(...) {
  Contract.Requires(S ==
State.Authentic);
  Contract.Ensures(S ==
State.Decrypted); ...

}

...
```

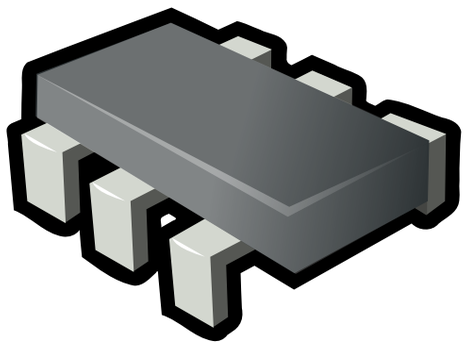
Checking that all data is encrypted using ghost variables

- Encrypt *all* data before sending it to FS / drivers



```
class FilePage {

...

  void Flush(...) {
    Contract.Requires(S ==
State.Encrypted);

    ...

  }

}
```

Memory isolation

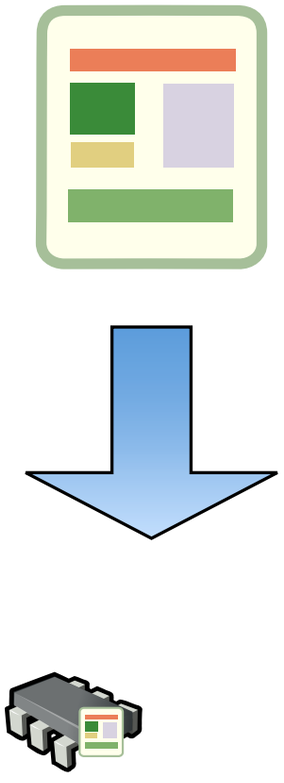


- The pager can read a file and bring it into the application's memory *only if* the application has proper accesses to the file.

Challenge: asynchronous execution

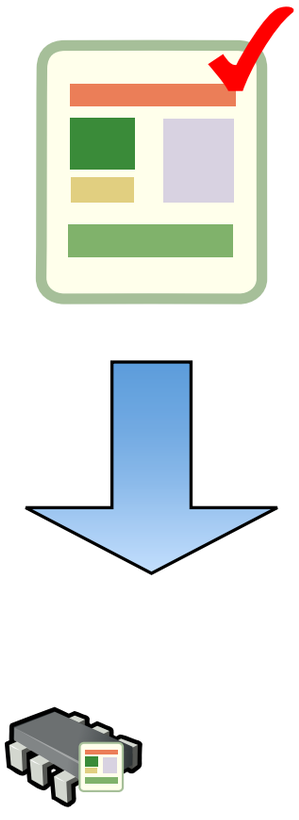
Challenge: asynchronous execution

open/mmap



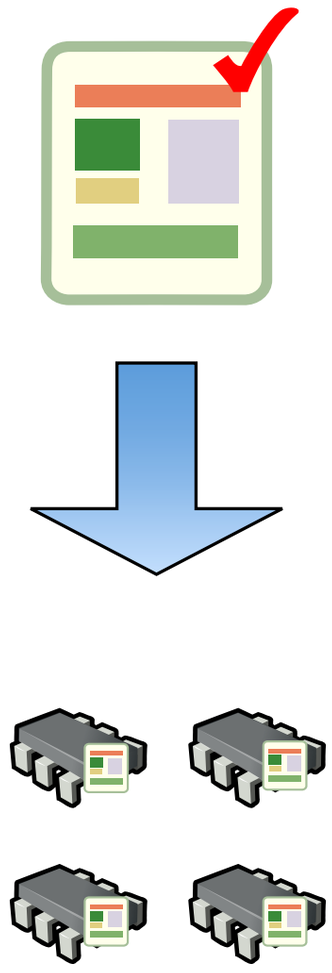
Challenge: asynchronous execution

open/mmap

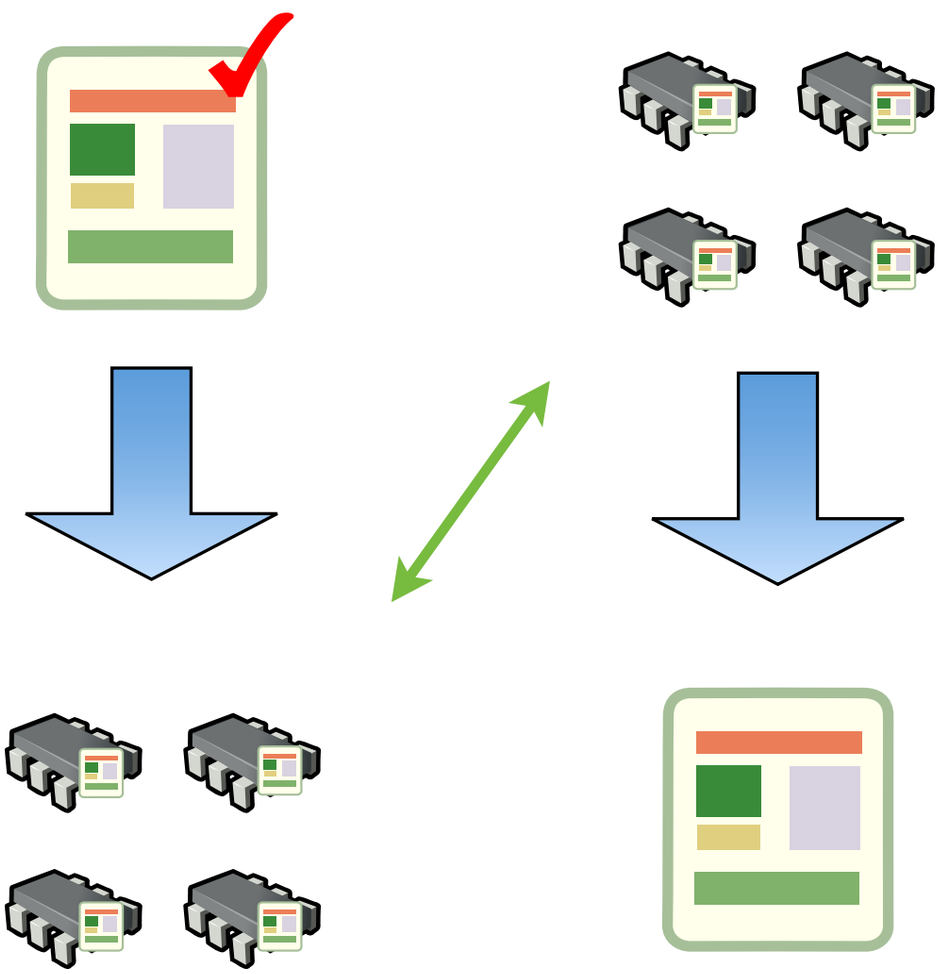


Challenge: asynchronous execution

open/mmap

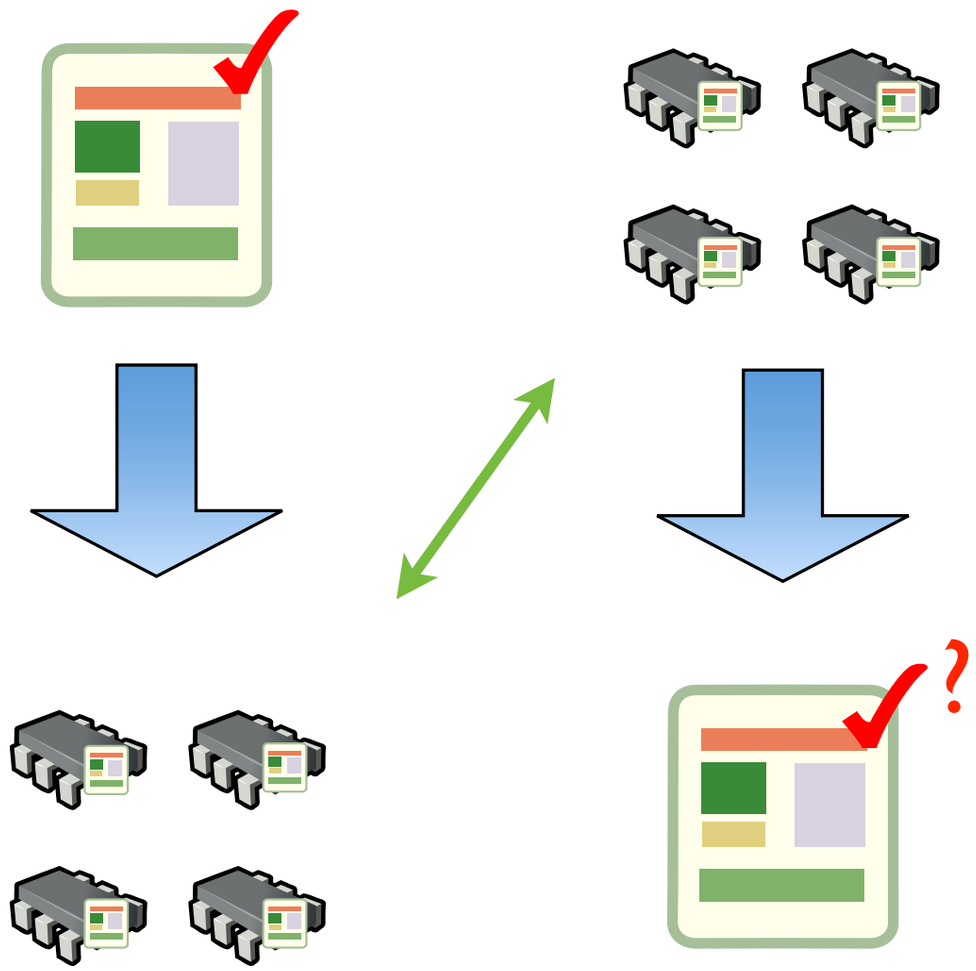


Challenge: asynchronous execution



Challenge: asynchronous execution

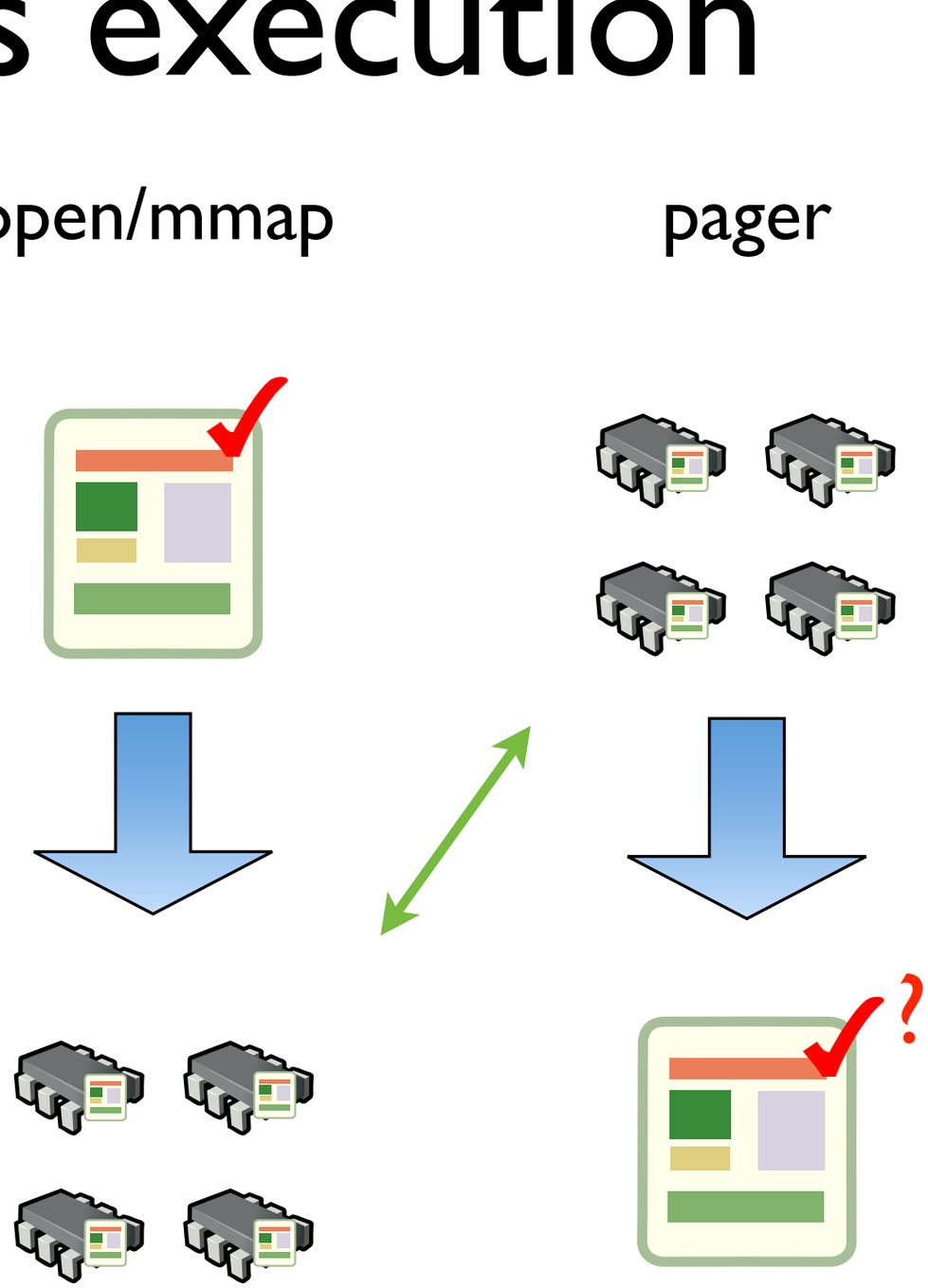
- Insufficient information at the point of assertions



Challenge: asynchronous execution

- Insufficient information at the point of assertions

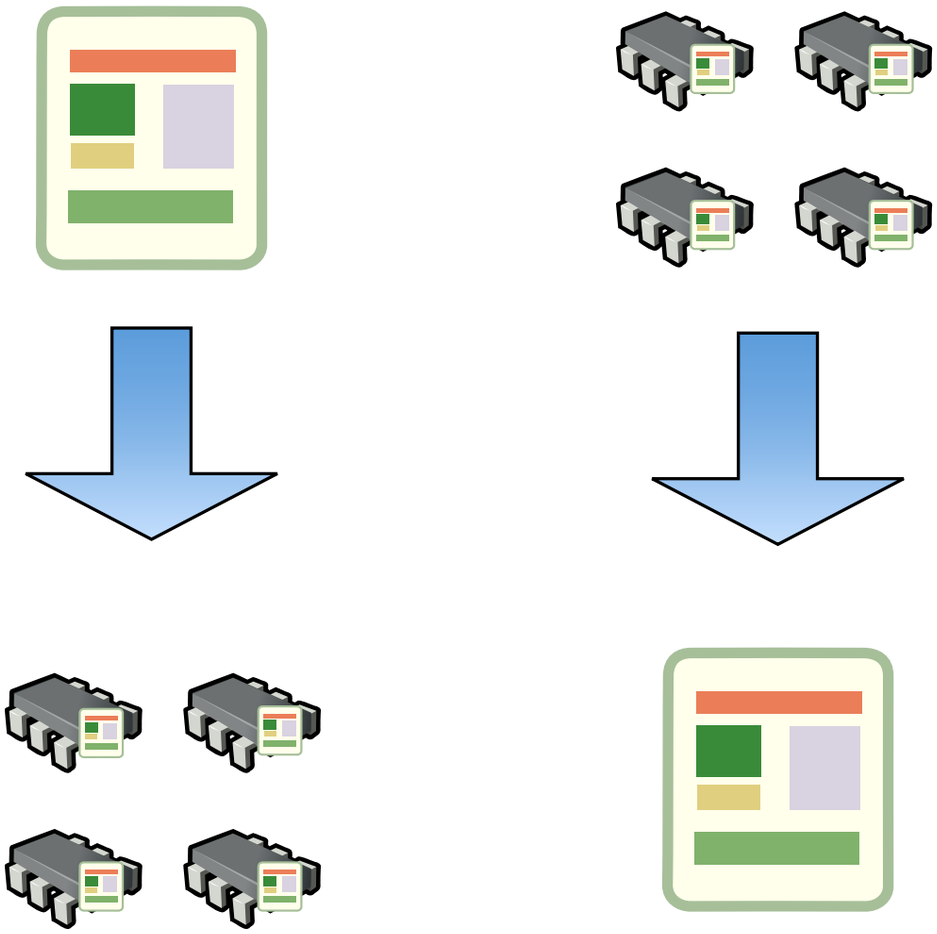
- Permission checks and paging in different execution contexts



Solution: strengthen object invariants

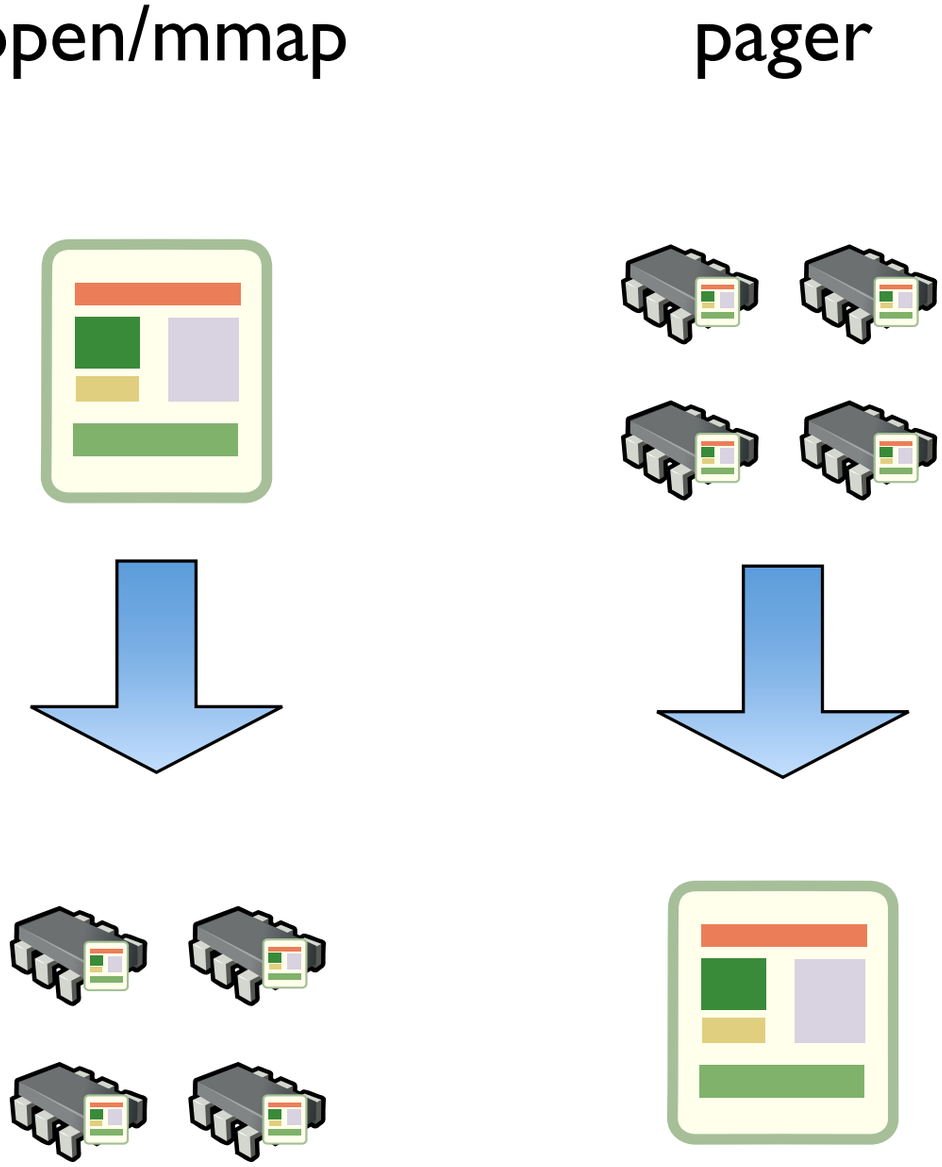
open/mmap

pager



Solution: strengthen object invariants

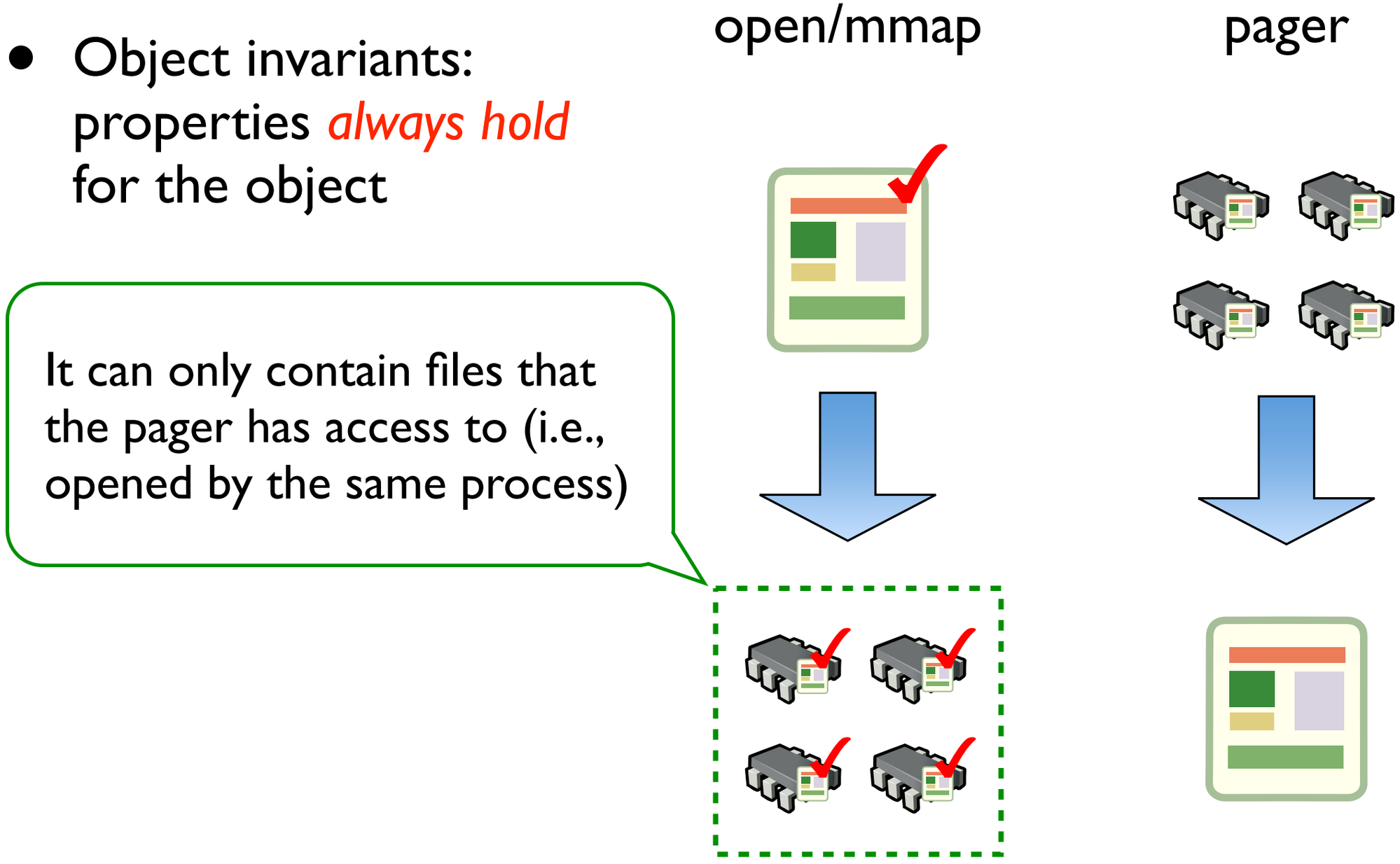
- Object invariants: properties *always hold* for the object



Solution: strengthen object invariants

- Object invariants: properties *always hold* for the object

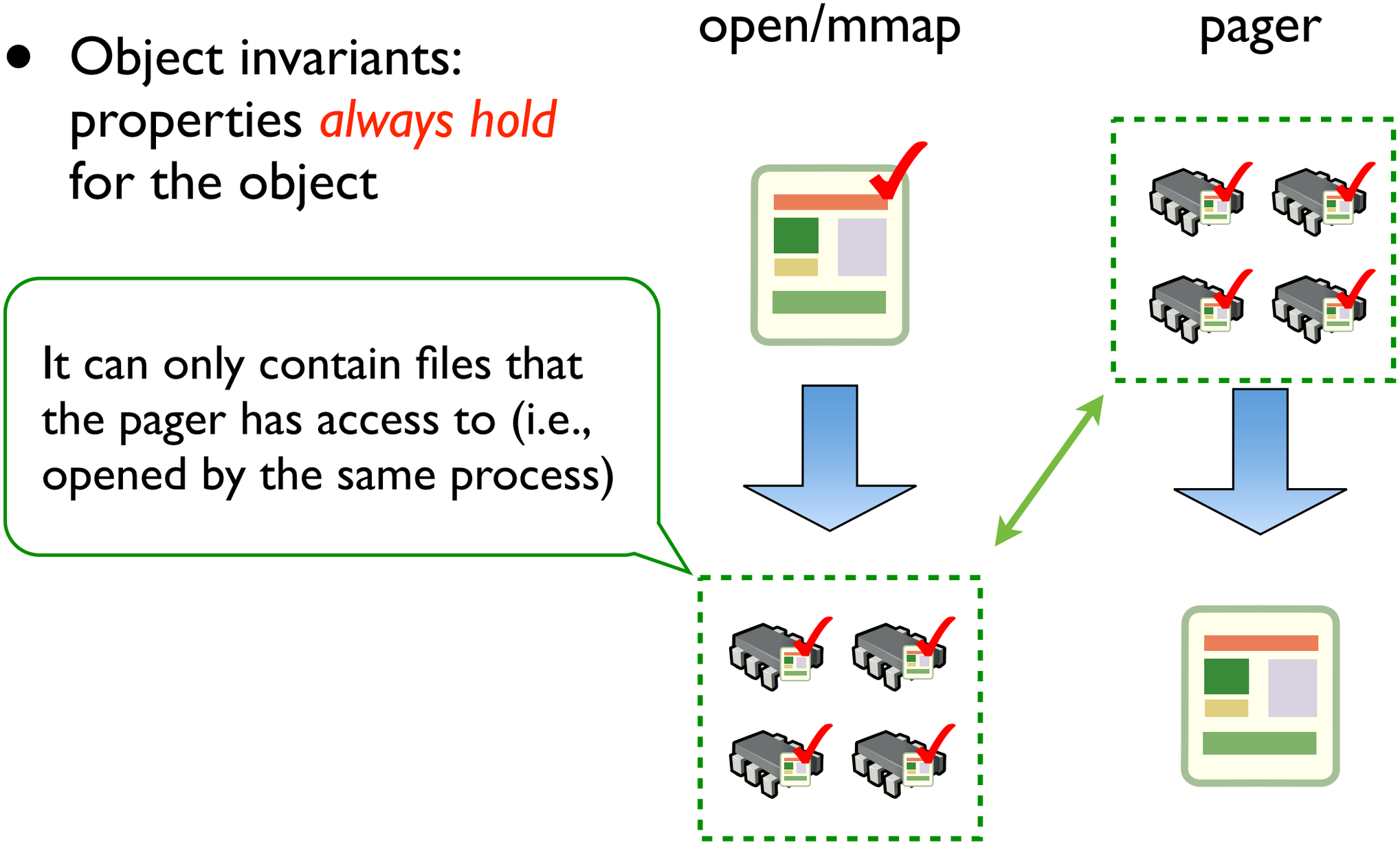
It can only contain files that the pager has access to (i.e., opened by the same process)



Solution: strengthen object invariants

- Object invariants: properties *always hold* for the object

It can only contain files that the pager has access to (i.e., opened by the same process)

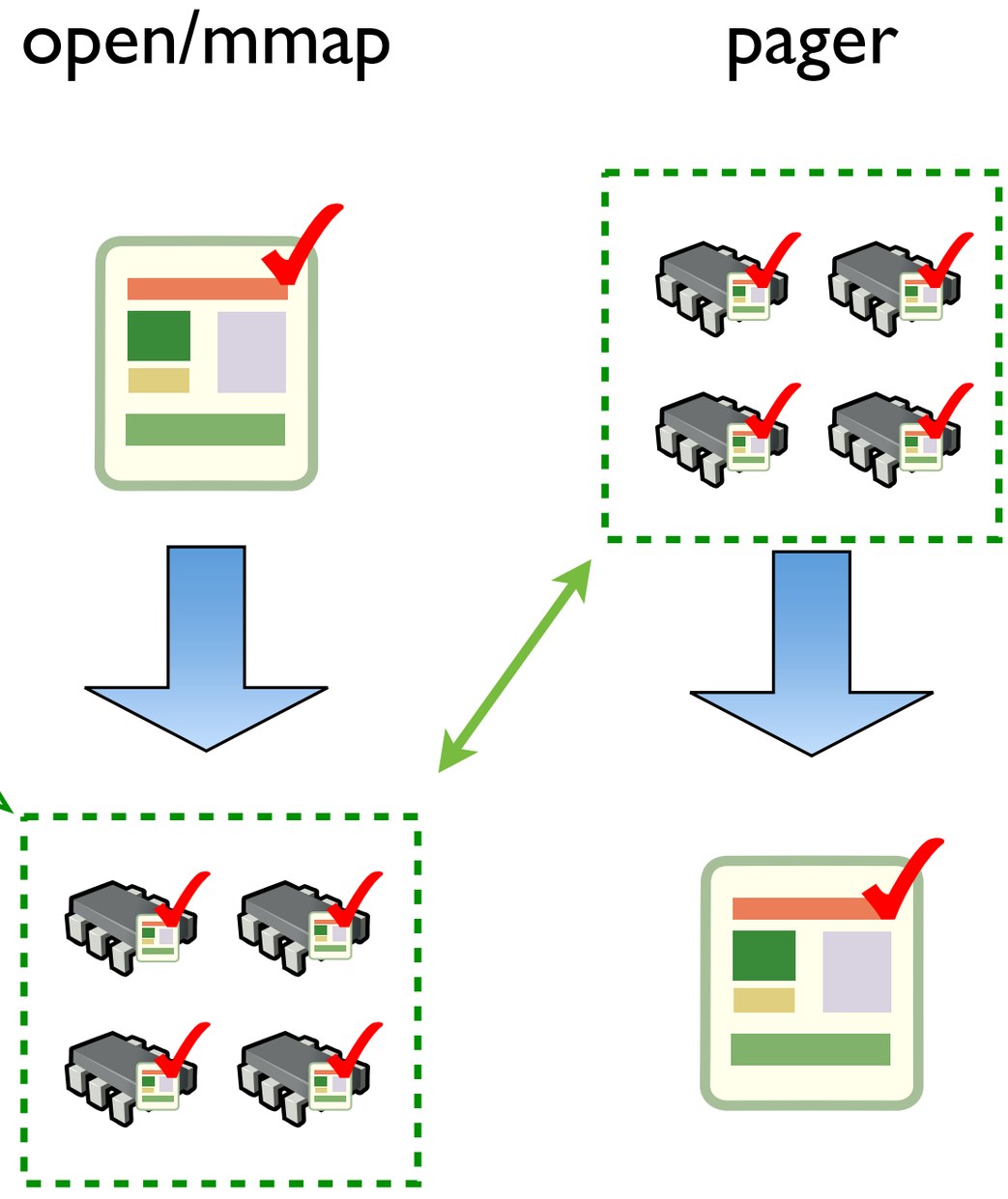


Solution: strengthen object invariants

- Object invariants: properties *always hold* for the object

It can only contain files that the pager has access to (i.e., opened by the same process)

- Can be reasoned about locally



Pager in C#

```
uint HandlePageFault(Process
p, Pointer addr, ...) {

...
var r = p.Space.Find(addr);
...
if (r.File != null) {
  var r = r.File.Read(...);
  ...
}
}
```

Pager in C#

```
uint HandlePageFault(Process
p, Pointer addr, ...) {

...
var r = p.Space.Find(addr);
...
if (r.File != null) {
   var r = r.File.Read(...);
   ...
}
}
```

Security Invariant
Contract.Assert(r.File.
GhostOwner == p);

Pager in C#

```
uint HandlePageFault(Process
p, Pointer addr, ...) {

...
var r = p.Space.Find(addr);

...
if (r.File != null) {
  var r = r.File.Read(...);
  ...
}
}
```

ensures $r \neq null \rightarrow r.ObjInvariant()$
$\wedge$ *r.GhostOwner == GhostOwner;*

Security Invariant
Contract.Assert(r.File.
GhostOwner == p);

Pager in C#

```
uint HandlePageFault(Process
p, Pointer addr, ...) {

...
var r = p.Space.Find(addr);

...
if (r.File != null) {
  var r = r.File.Read(...);
  ...
}
}
```

Space.GhostOwner == this

File≠null→File.GhostOwner == GhostOwner ...

ensures *r≠null→r.ObjInvariant() ∧ r.GhostOwner == GhostOwner;*

r.File.GhostOwner == r.GhostOwner == space.GhostOwner == p

Security Invariant

Contract.Assert(r.File. GhostOwner == p);

AddressSpace in Dafny

```
class AddressSpace {
  var GhostOwner: Process;
  var Head: MemoryRegion;

  ...

  method Find(address:
Pointer)
  returns (ret: MemoryRegion)
  requires ObjInvariant();
  ensures ObjInvariant();

}
```

ensures *r≠null→r.ObjInvariant()*
∧ r.GhostOwner == GhostOwner;

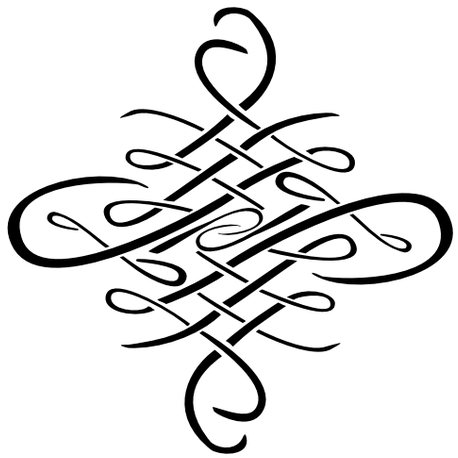
AddressSpace in Dafny

- Heavy verification effort
- 730 lines of code
- ~200 lines of annotations

```
class AddressSpace {
  var GhostOwner: Process;
  var Head: MemoryRegion;

  ...

  method Find(address:
Pointer)
  returns (ret: MemoryRegion)
  requires ObjInvariant();
  ensures ObjInvariant();
}
```

ensures $r \neq null \rightarrow r.ObjInvariant() \wedge r.GhostOwner == GhostOwner;$

Combining Dafny & CC



Dafny

ghost variables

Assume()

```
...
[Ghost] Foo GhostOwner;
Find(...) {
...
Assume(...);
...
}
```

C# + code contracts

Experience

- Focus on security invariants

Experience

- Focus on security invariants
    - Less components (isolation & end-to-end mechanisms)

Experience

- Focus on security invariants
  - Less components (isolation & end-to-end mechanisms)
  - Simpler properties (object invariants vs async contexts)

Experience

- Focus on security invariants
  - Less components (isolation & end-to-end mechanisms)
  - Simpler properties (object invariants vs async contexts)

	Line of code
Full system	13M
Linux kernel	1M
ExpressOS	15,932
Annotation	438

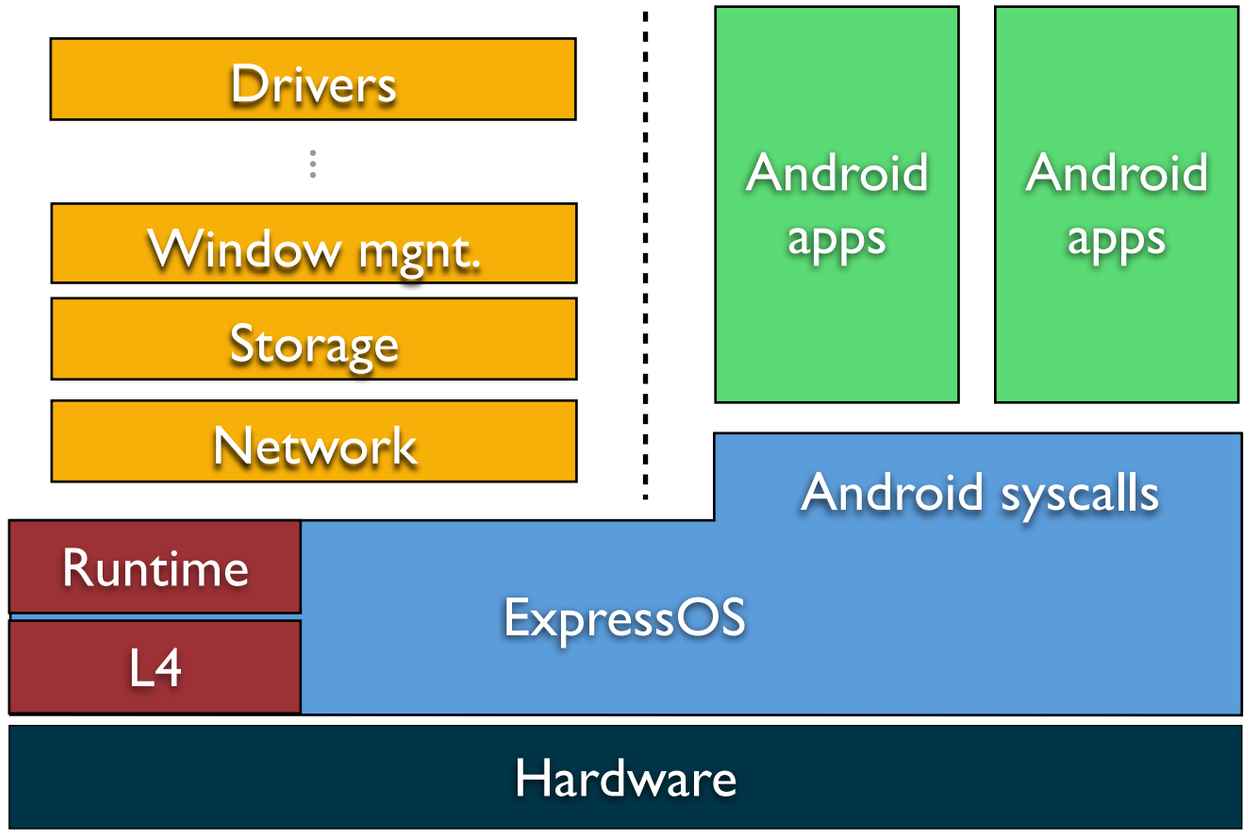
- Code contracts + Dafny: 2.8% annotation overhead

Experience on verification

- Code contracts are *mostly* sufficient
  - Plus ghost variables and the concepts of ownerships
- Dafny for the rest

Implementation

- Build on top of L4::Fiasco
- Use L4Android to implement system services
  - Turning Linux into a microkernel server
- Sufficient to run the Android web browser and this presentation

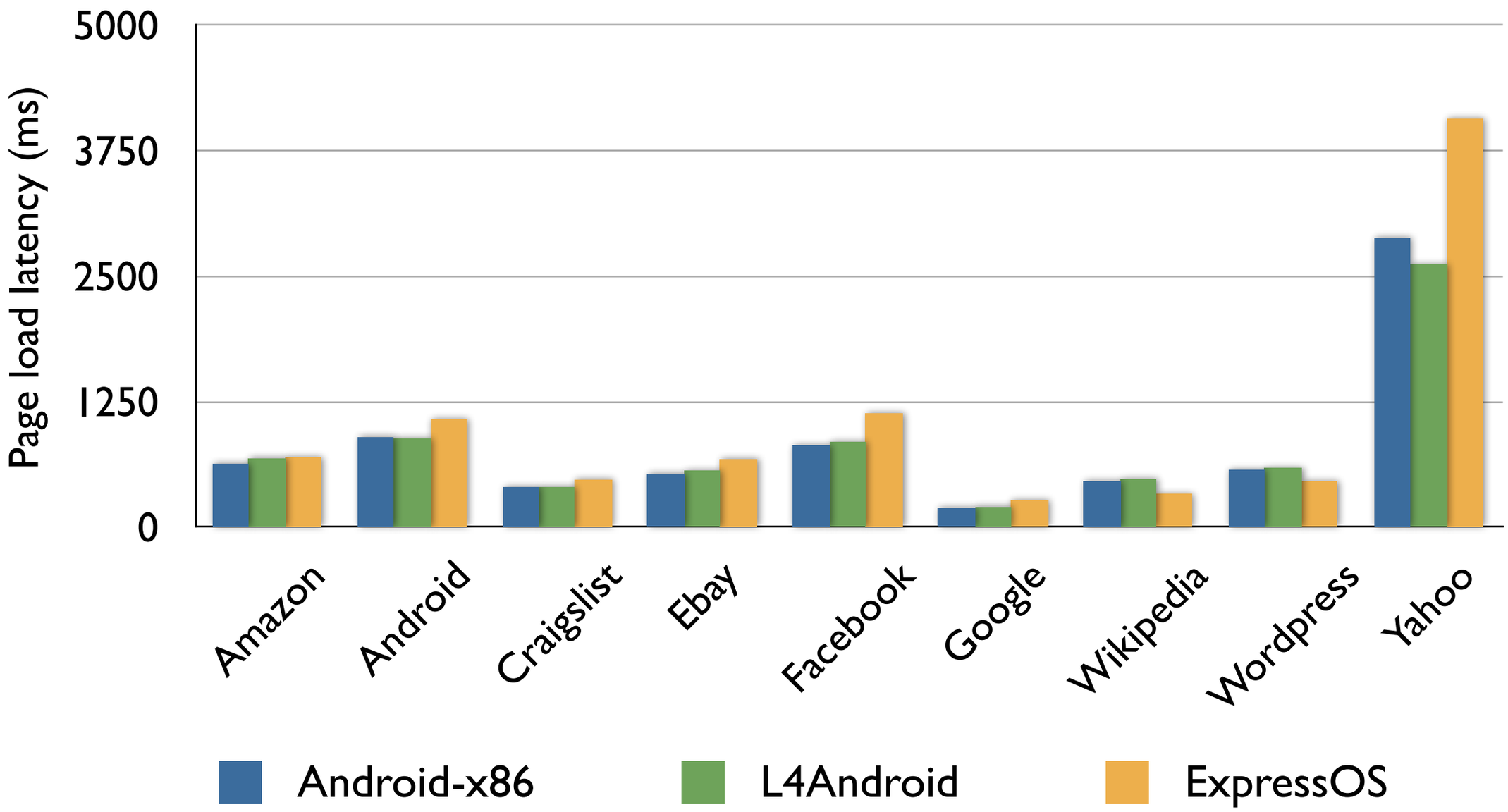


Security analysis

- Studied 742 vulnerabilities from CVE (from Jun, 2011~Jun, 2012)
- 383 of them affect Android
- ExpressOS prevents 364 (95%) of them

	Num	Prevented
Core kernel	9	9
Library of apps	102	102
Services	240	226
Applications	32	27

Page load latency on web browsing



Conclusion

- ExpressOS: *a high assurance OS that runs Android applications*

Conclusion

- ExpressOS: *a high assurance OS that runs Android applications*
  - Define *security invariants*

Conclusion

- ExpressOS: *a high assurance OS that runs Android applications*
  - Define *security invariants*
  - *Isolate* vulnerabilities of components

Conclusion

- ExpressOS: *a high assurance OS that runs Android applications*
  - Define *security invariants*
  - *Isolate* vulnerabilities of components
  - Verify security invariants *directly*

Conclusion

- ExpressOS: *a high assurance OS that runs Android applications*
  - Define *security invariants*
    - *Isolate* vulnerabilities of components
    - Verify security invariants *directly*
- Practical approach to establish high assurance in real-world systems

Thank you!

Source code available at:

https://github.com/ExpressOS/expressos