A Case Study in Language-Based Security: Building an I/O Library for Wyvern

Jonathan Aldrich
aldrich@cs.cmu.edu
http://www.cs.cmu.edu/~aldrich/

Jennifer Fish, Darya Melicher, and Jonathan Aldrich.
A Case Study in Language-Based Security: Building an I/O Library for Wyvern.
To appear in Onward! 2020.

CyLab Partners Conference
September 2020

institute for SOFTWARE RESEARCH
Carnegie Mellon University Computer Science Department
CyLab Security and Privacy Institute

Emerging languages build in security features

Java: static types, stack inspection

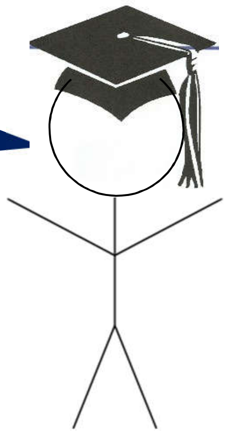
Jif: information flow

E: capability safety

Rust: safe memory management

Why should I use language based security?

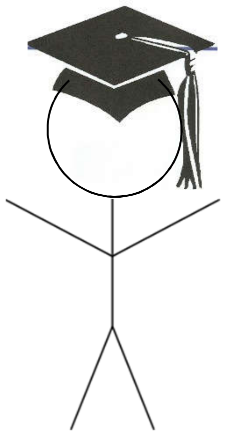
Because it can eliminate whole classes of vulnerabilities



Not all features are useful, or usable

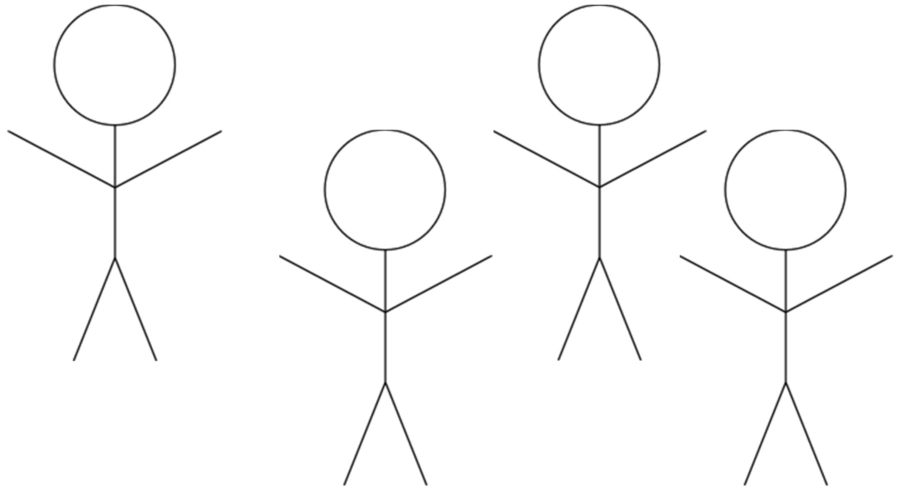
Example class experience with an information flow type system

Successful case:



Ph.D. student in programming languages

Unsuccessful case:



Everybody else

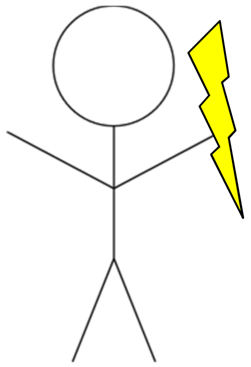
What PL-based security features are ready…

…for language designers to integrate?

…for early adopters to use?

The Wyvern PL has novel security features

Wyvern was designed from the ground up to be secure. It has static types, plus:

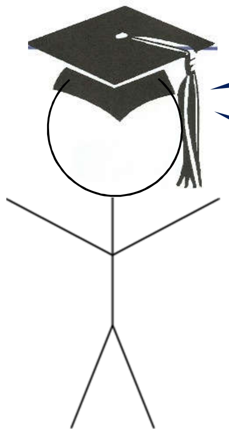


Object capabilities

```
Wyvern
  + RegEx
  + SQL
```

Language extensions

Effect system



RQ 1: Can we write programs with these features?

RQ 2: Do these features aid security in practice?

Building an I/O library is a useful case study

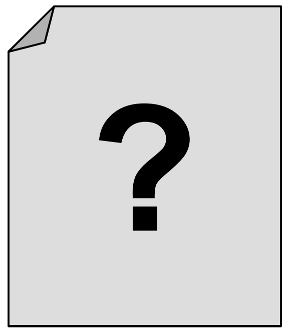
- Everything dangerous involves input or output

read_password()

launch_missiles()

- We built a small standard I/O library for Wyvern, and investigated…
  - For what **purposes** could we use Wyvern's security features?
  - How did these features **enhance system security**?
  - Were there any **barriers** to using these features?

Want to limit privileges of untrusted code



Untrusted.java

Past sandboxing solutions (e.g. Java)

- Complex → error-prone
- Many past vulnerabilities

I want to use this module, but is it safe?

Capabilities give you the ability to do things

Zeus

With my lightning bolt, I can destroy anything!
I can't do anything bad…unless I get that bolt!

Luke Castellan

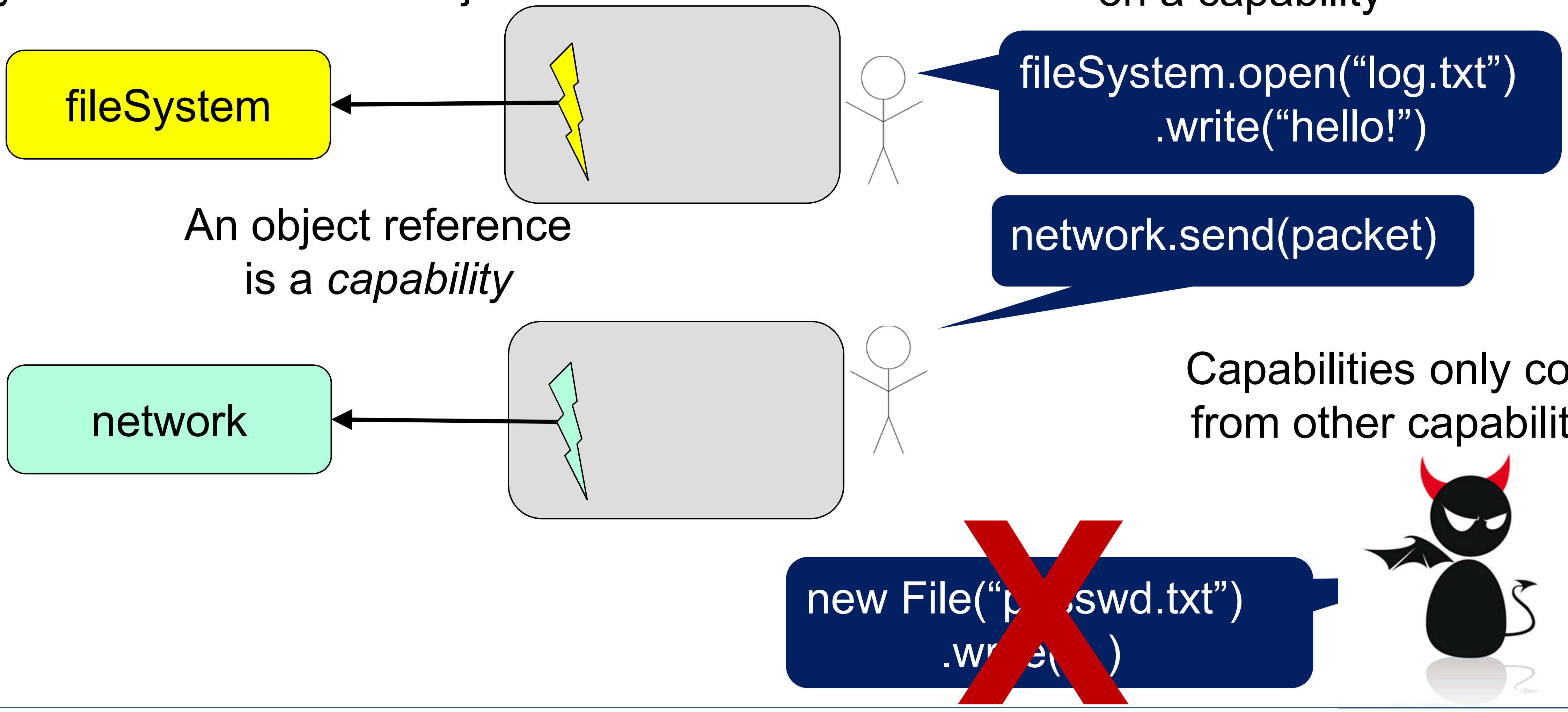
Wyvern’s object capabilities control privileges

System resources are objects

All actions are methods on a capability

An object reference is a *capability*

Capabilities only come from other capabilities



```
fileSystem.open(“log.txt”)
    .write(“hello!”)
```

```
network.send(packet)
```

```
new File(“p  swd.txt”)
    .wr  e(   )
```

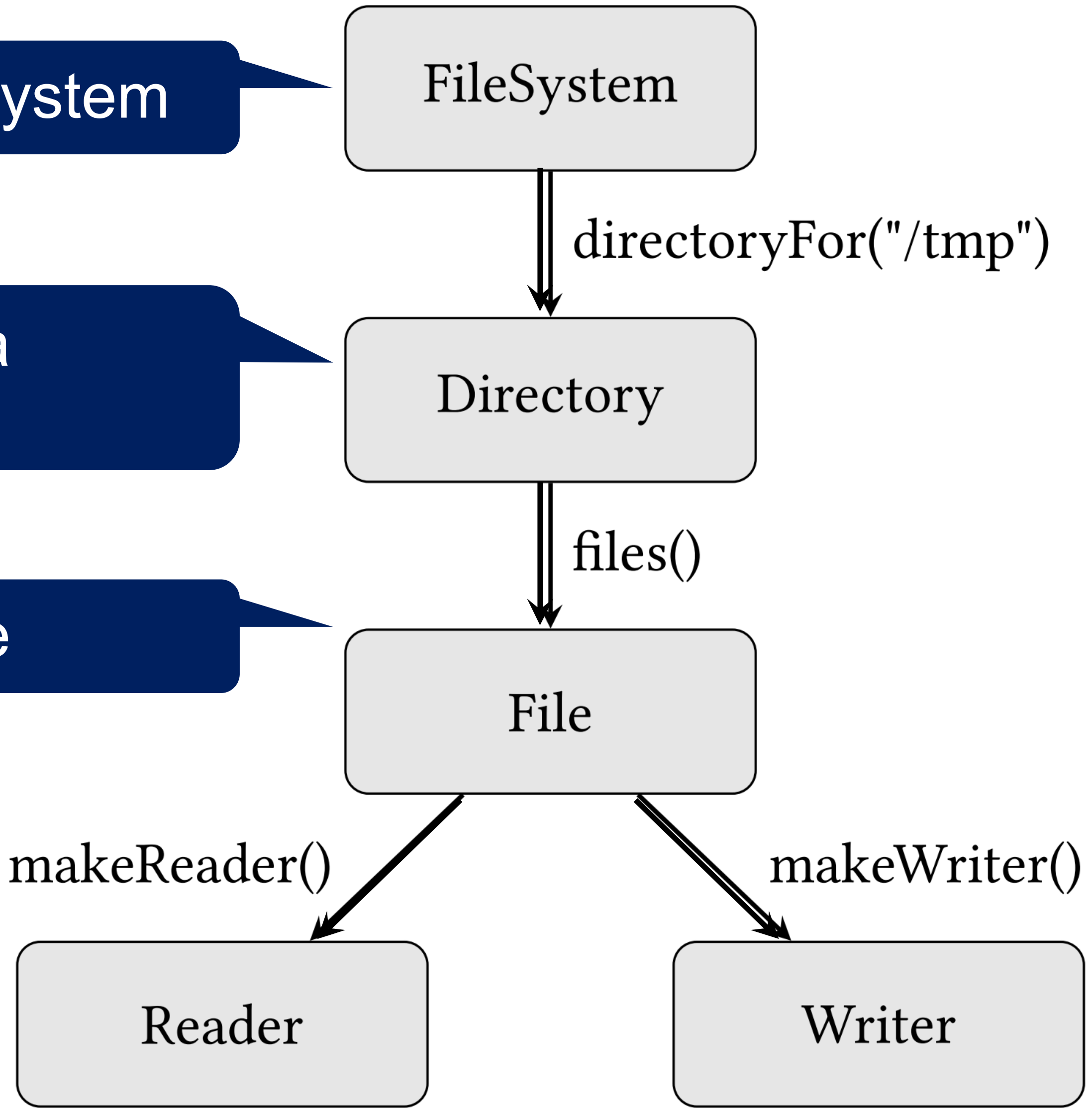
Capability granularities in file system design

Capability to entire file system

Capability to all files in a directory

Capability to a single file

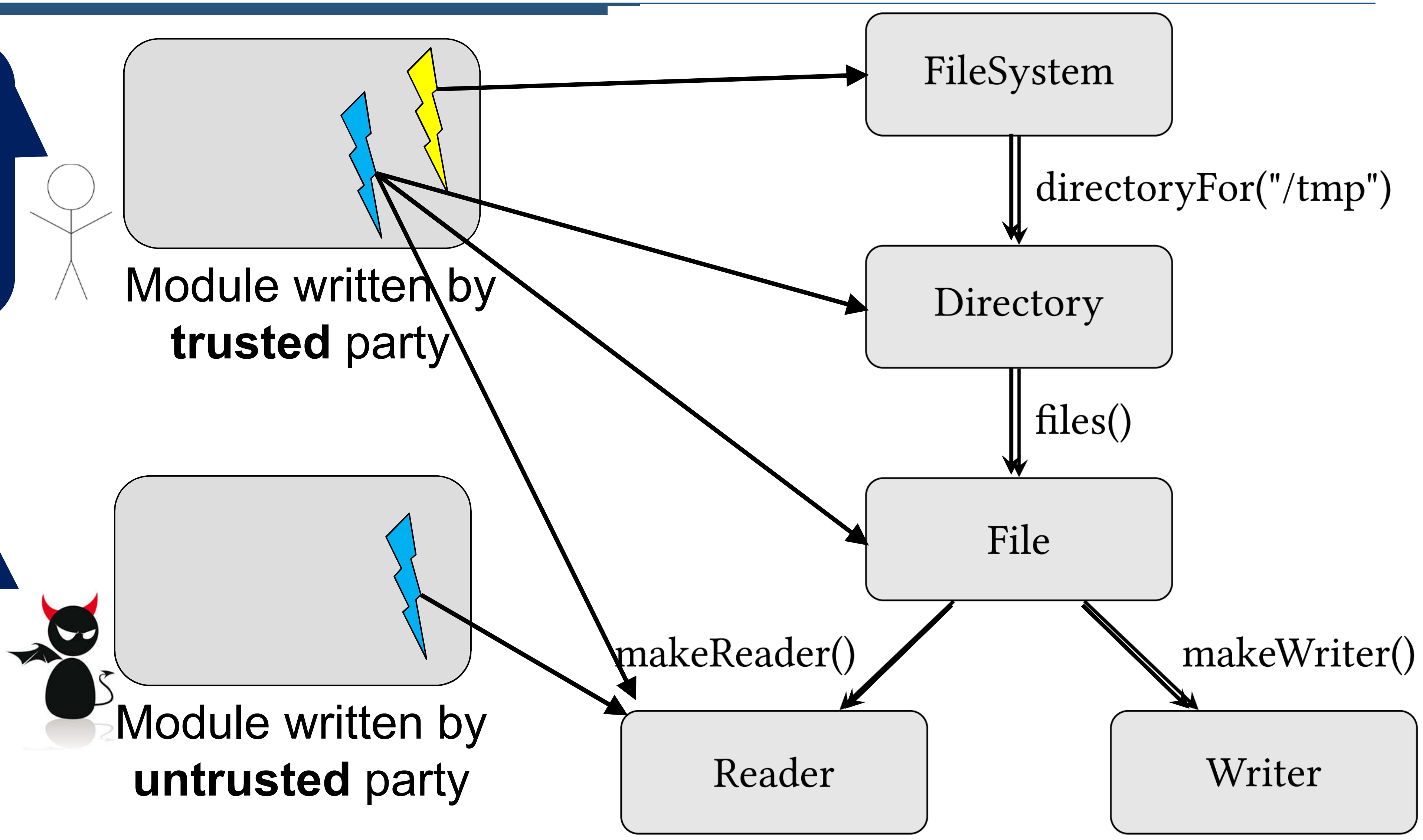
Read-only access to a file



Restricting untrusted module privileges

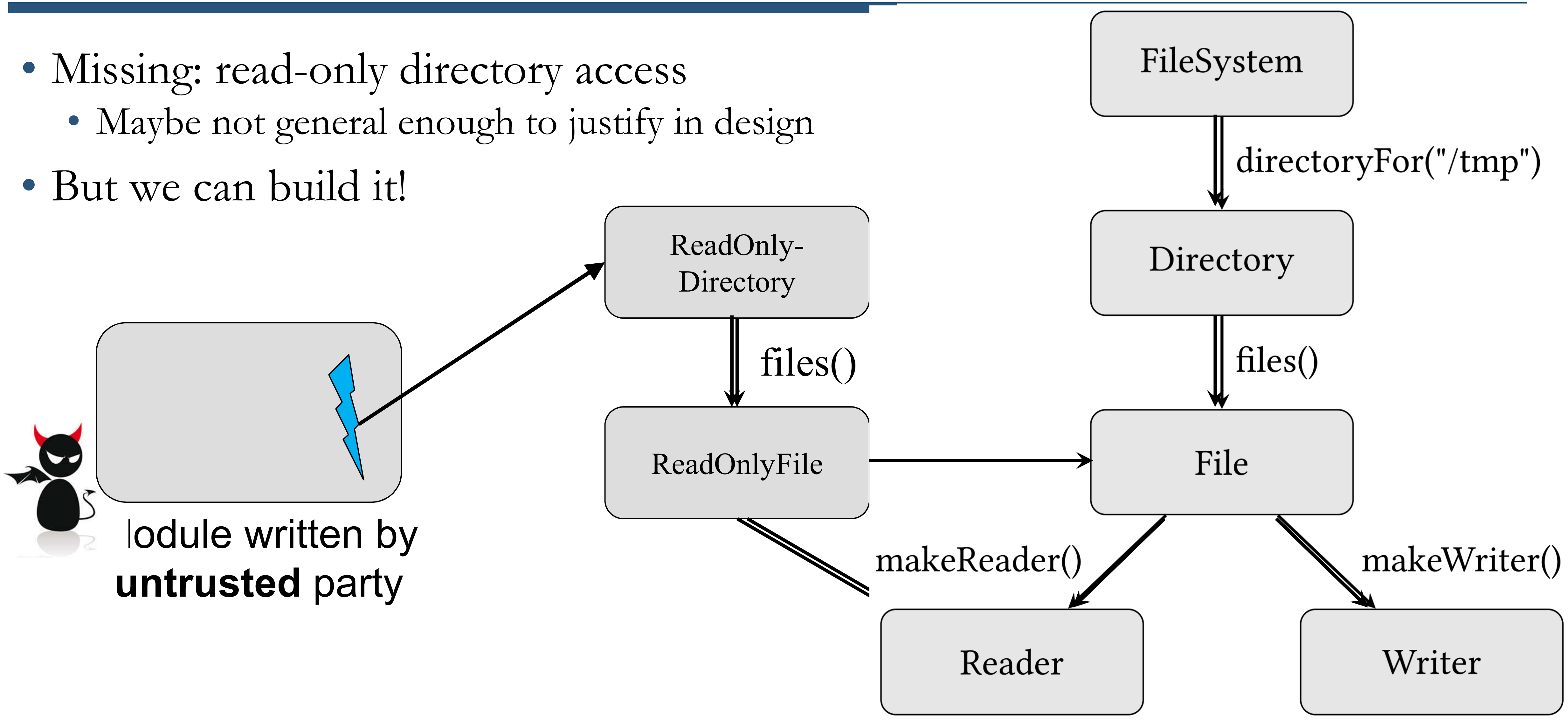
I want to use a third-party module, but don't fully trust it

Shucks! All I can do is read one lousy file!



Other permissions supported by wrapping

- Missing: read-only directory access
  - Maybe not general enough to justify in design
- But we can build it!



Capabilities were effective; tweaks can help

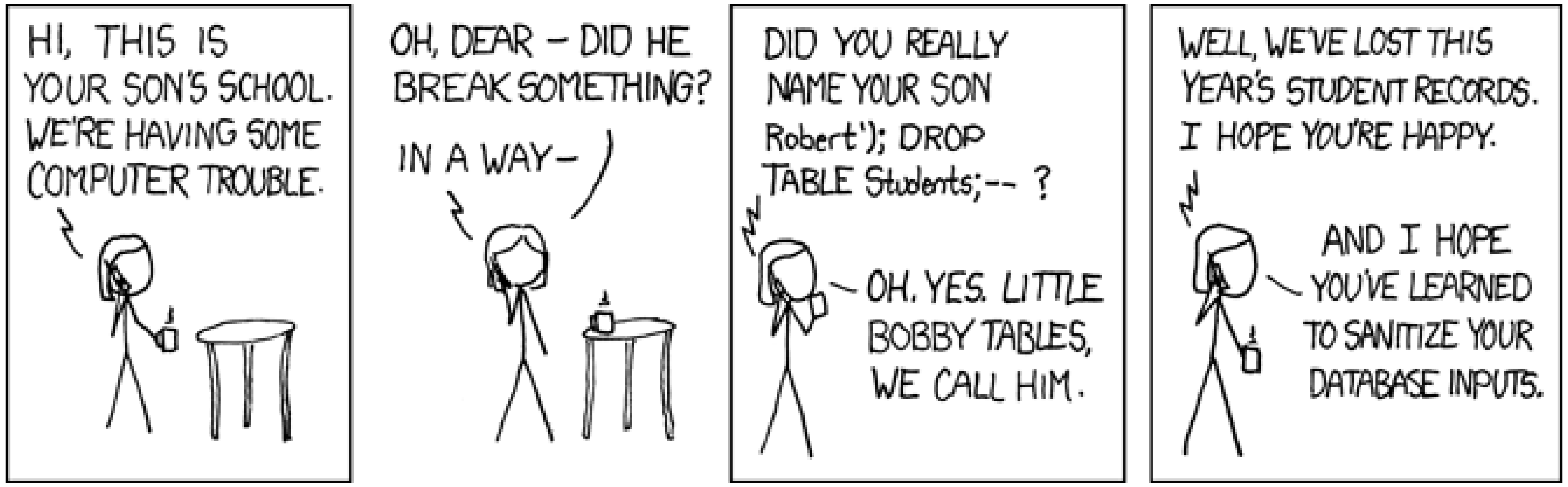
- Design provides useful granularity variations
- Supports user-defined abstractions
- Extensions to Wyvern’s type system could make wrapping more efficient

Effects: static reasoning about I/O actions

```
resource type Writer
   effect Write
   def write(s: String) : {this.Write} Unit
```

- Interesting tradeoffs between global and file-specific effects
- Study motivated new effect-checking features

PL extensions mitigate command injection


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

credit XKCD: HTTP://XKCD.COM/327/

- Command injection can't happen if programmers write literal commands
  - Instead of embedding them in strings
- Case study: simple but expressive library for string formatting (printf, etc.)

Wyvern I/O Library Case Study Takeaways

- **Capabilities** support the **principle of least privilege**
- **Effects** support **static reasoning about I/O actions**
- **Language extensions** mitigate **command injection**
- Identified language design improvements
  - Follow-up research working on this now!

- **Read more:** Jennifer Fish, Darya Melicher, and Jonathan Aldrich. *A Case Study in Language-Based Security: Building an I/O Library for Wyvern*. To appear in Onward! 2020.