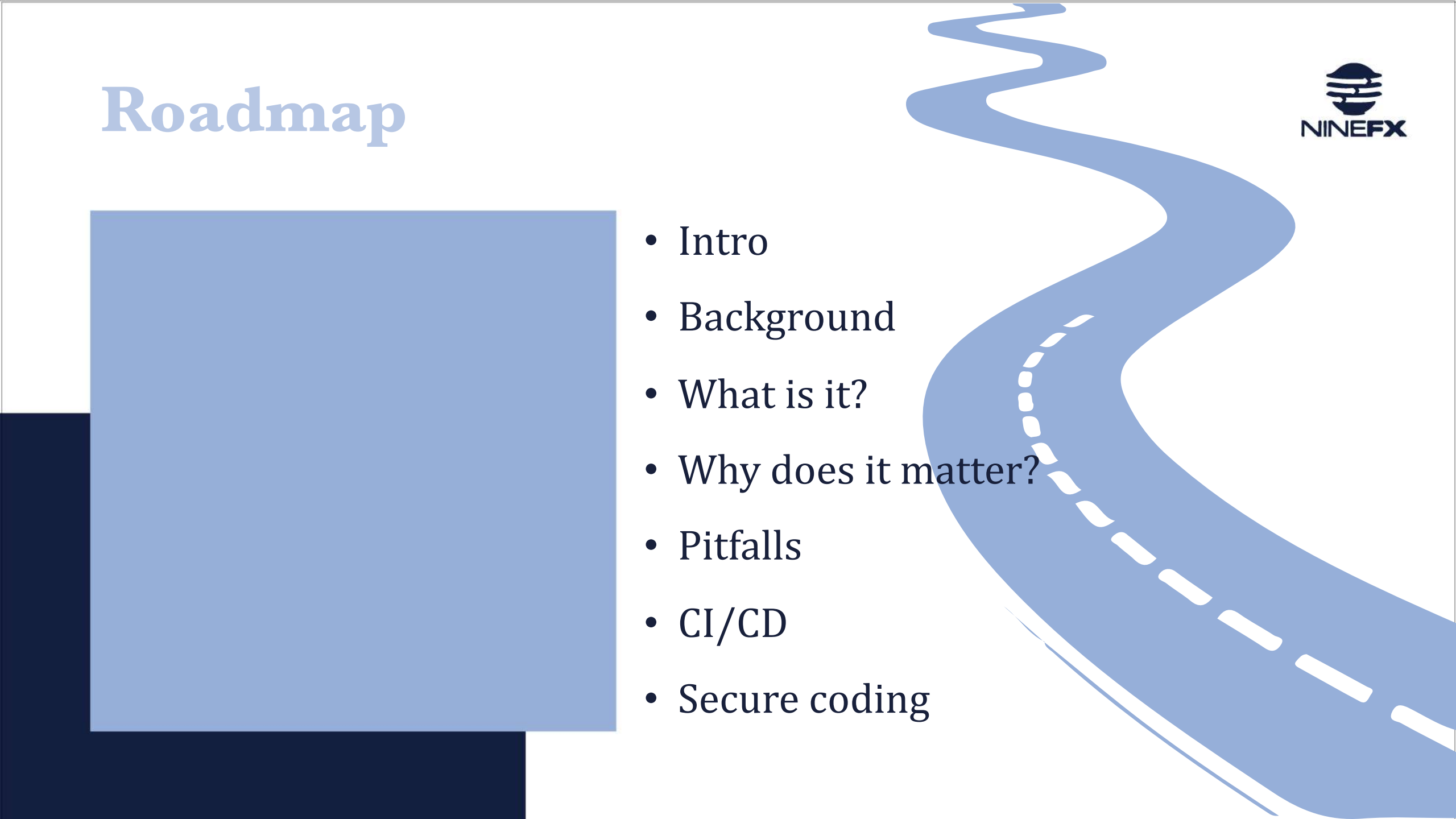
FIPS 140-2 in BEAM Apps

Roadmap



- Intro
- Background
- What is it?
- Why does it matter?
- Pitfalls
- CI/CD
- Secure coding

Intro

NINEFX

- Ground Rules
  - No advocacy
  - IANAL
- Background
  - BEAM languages
  - Federal experience

Who are you?

- Supporting a BEAM application in the Federal space
- Developing a BEAM application for the Federal space
- Considering a move to the Federal space

What is FIPS 140-2?

"Security Requirements for Cryptographic Modules"

- **Administered by NIST Cryptographic Module Validation Program**
- **Cryptographic and Security Testing (CST) Laboratories perform conformance testing of cryptographic modules**
- **Covers hardware and software security modules**

What is FIPS 140-2?

- **Specifies approved security functions and Deterministic Random Number Generators**
- **Validation Certificate**
- **Certification Types**
  - FIPS Validated
  - FIPS Inside

What FIPS 140-2 isn't

- **Does not restrict hashing outside of cryptographic context**
  - `erlang:md5/1`
  - `erlang:phash/2`
  - `erlang:phash2/1`
- **FIPS isn't just the crypto module**
- **It is not forgiving in some implementations**

Should I enable FIPS mode?

No! Unless you have to…

Why does it matter?

NINEFX

Phase 1	Phase 2	Phase 3

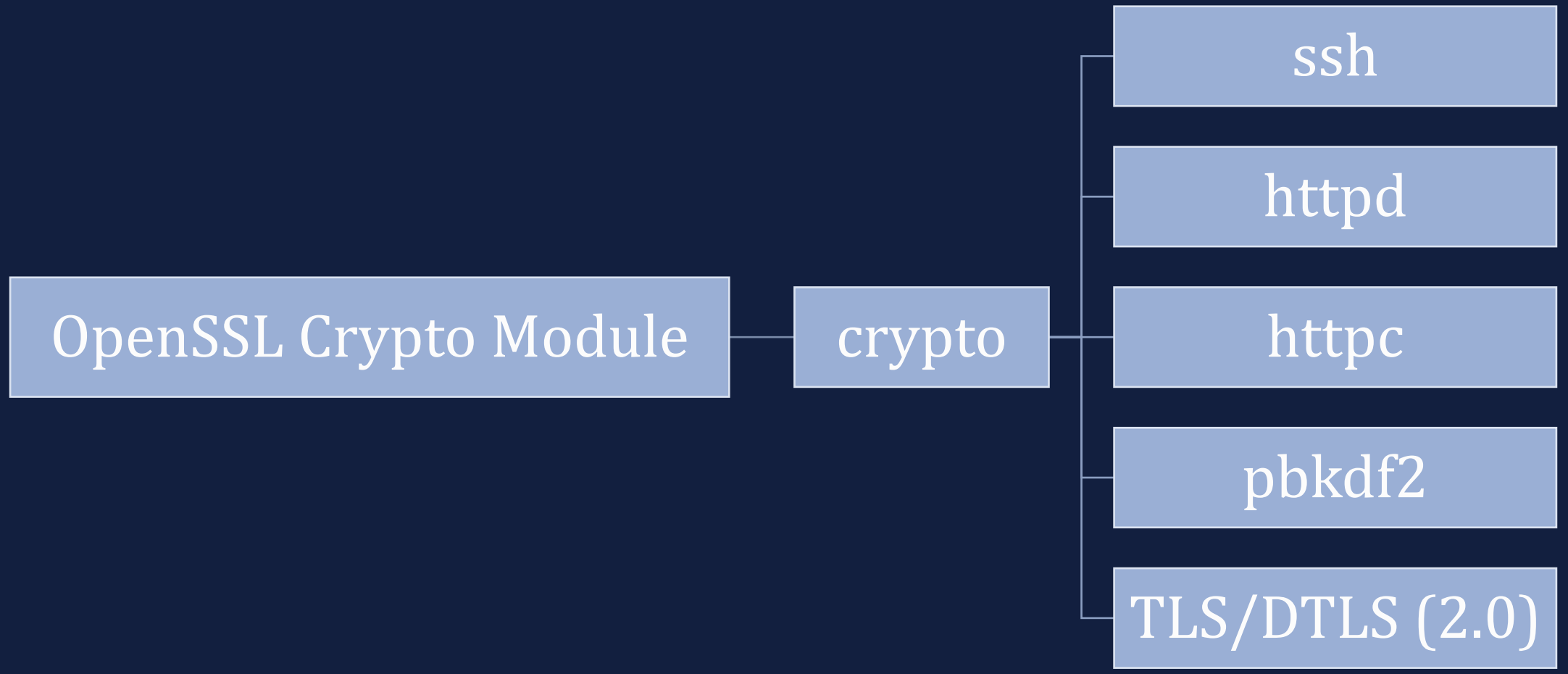
`enable_fips_mode(true)`

?

Profit

https://southpark.fandom.com/wiki/Underpants_Gnomes

Why does it matter?



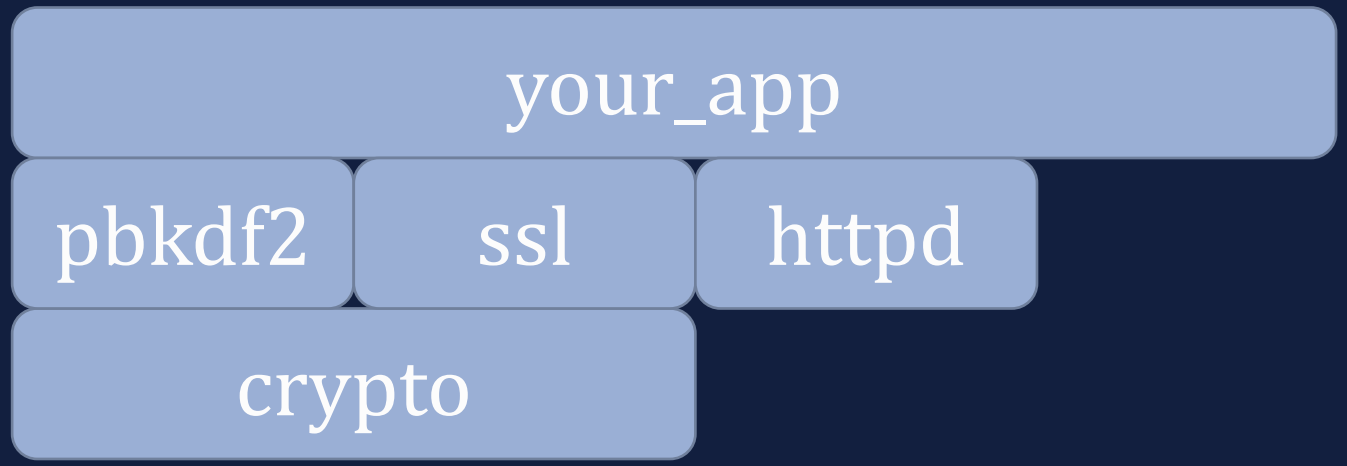
Pitfalls

Cryptographic Boundary

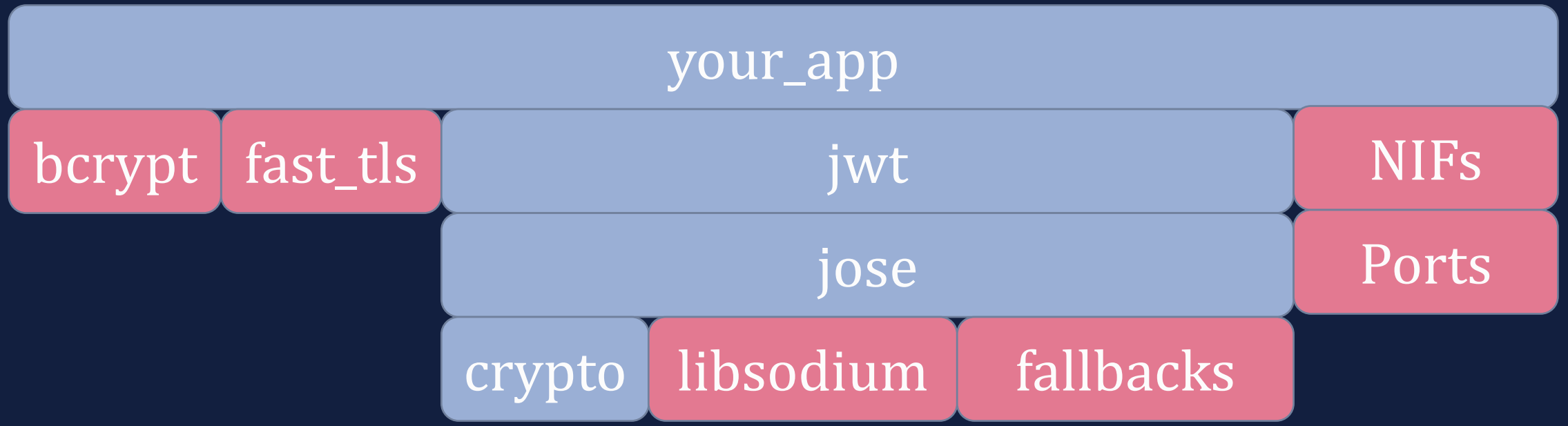
An explicitly defined continuous perimeter that establishes the physical bounds of a cryptographic module and contains all the hardware, software, and/or firmware components of a cryptographic module.

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.140-2.pdf

Compliant Application



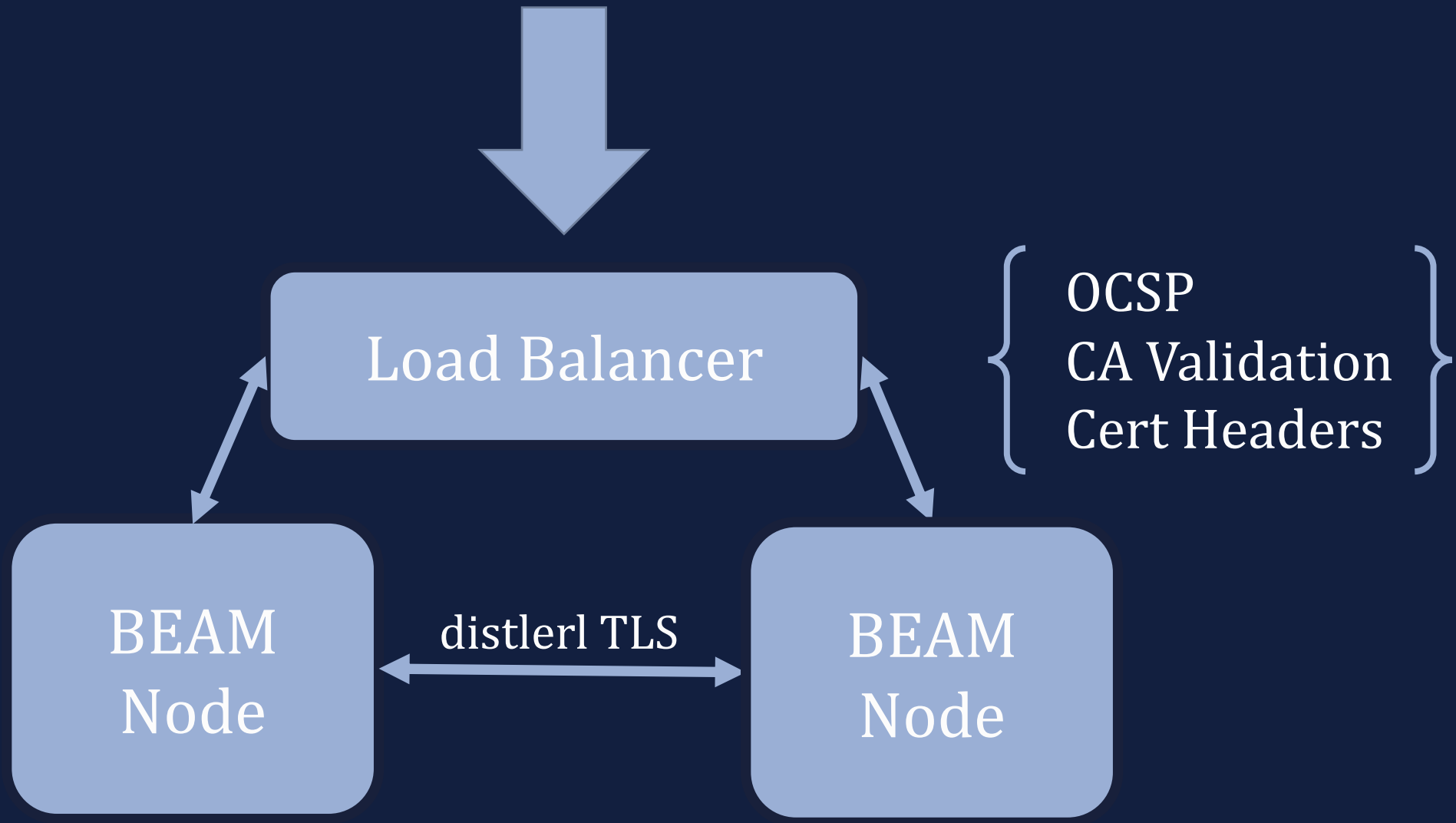
Non-compliant Application



Dependencies risk expanding the cryptographic boundary to include uncertified algorithms

Inter-node Encryption

NINEFX

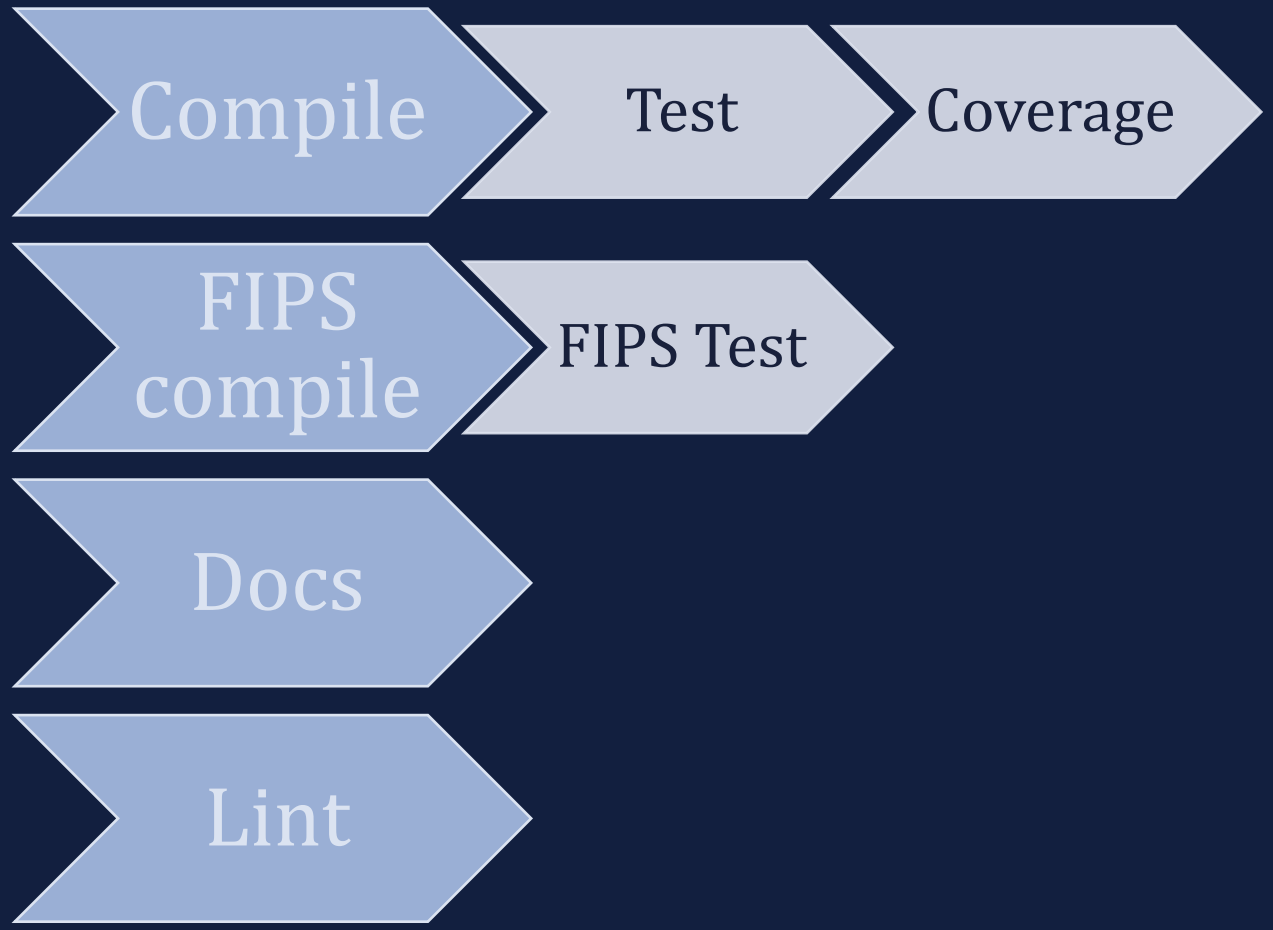


MongooseIM

NINEFX

FIPS 140-2 CI/CD

Concurrent FIPS Testing



Secure Coding

Risk Areas

NIFs

Resource Exhaustion

Atoms

ETS

Traditional Vulnerabilities

Timing

Injection

Shell

SQL

Native Implemented Functions

NINEFX

Bleeding-edge performance

- dynamically linked into the emulator
- a crash in a NIF brings the emulator down

Atom Exhaustion Risks

Danger	Safe
list_to_atom(L)	list_to_existing_atom(L)
binary_to_atom(B, utf8)	binary_to_existing_atom(B, utf8)
binary_to_term(B)	binary_to_term(B, [safe])
http_uri:parse(URI)	http_uri:parse(URI, [{scheme_validation_fun, fun foo/1}])
xmerl_scan:*	Another parser

Time: Not on your side


```
-module(codebeam).

-export([validate_username/1]).

validate_username(Username) ->
  ExpectedUsername = fetch_username(),
  case Username of
    ExpectedUsername -> ok;
    _ -> access_denied
  end.

fetch_username() -> <<"foo">>.
```

Time: On your side

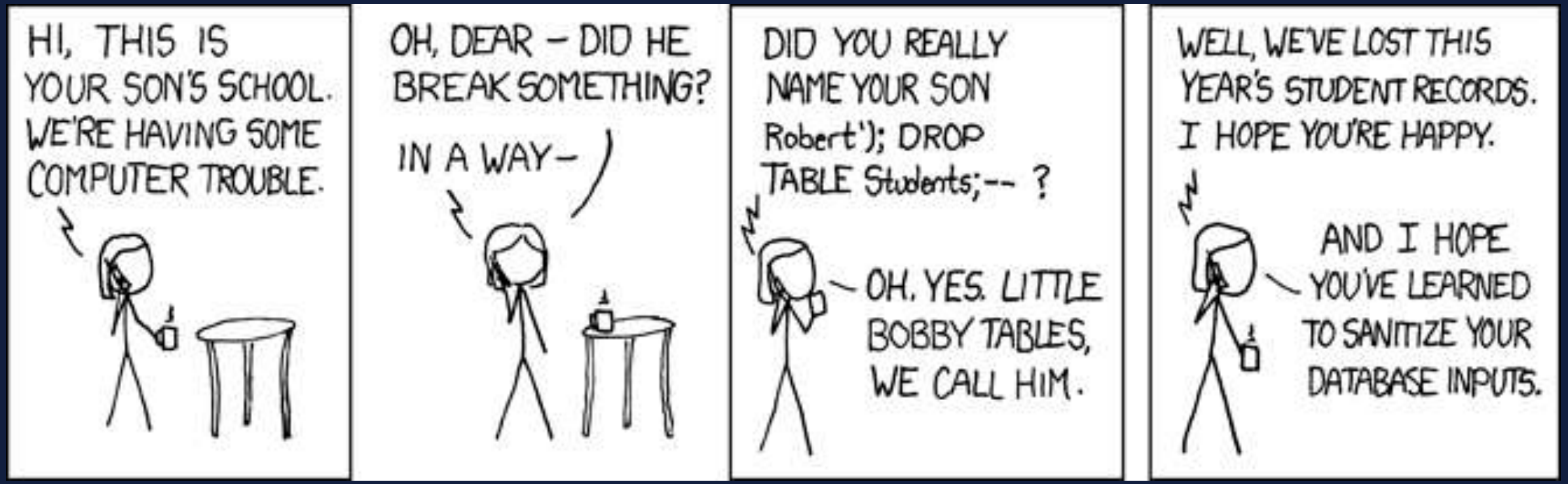

```
-module(codebeam).

-export([validate_username/1]).

validate_username(Username) ->
  Expected = fetch_username(),
  case pbkdf2:compare_secure(Expected, Username) of
    true -> ok;
    false -> access_denied
  end.

fetch_username() -> <<"foo">>.
```

Little Bobby Tables



https://xkcd.com/327/

Injections

SQL Injection

```
"SELECT * FROM dogs WHERE id = "  ++ Id
```

Shell Injection

```
"convert temp.gif -resize 128x128\> " ++ Out ++ ".gif"
```

Property-based Testing

As a security person, you need to repeat this mantra:

"security problems are just bugs"

and you need to _internalize_ it, instead of scoff at it.

- Linus Torvalds

http://lkml.iu.edu/hypermail/linux/kernel/1711.2/01701.html

References

Name	URL
OpenSSL FIPS Documentation	https://www.openssl.org/docs/fips.html
CircleCI Blog FIPS & BEAM	https://circleci.com/blog/workflow-testing-for-fips-140-2-compatibility/
NineFX Containers	https://hub.docker.com/u/ninefx/
Primitive Erlang Security Tool	https://github.com/okeuday/pest
Timing Attacks	https://codahale.com/a-lesson-in-timing-attacks/

Thanks!

drew.varner@ninefx.com

https://www.ninefx.com

@ varnerac