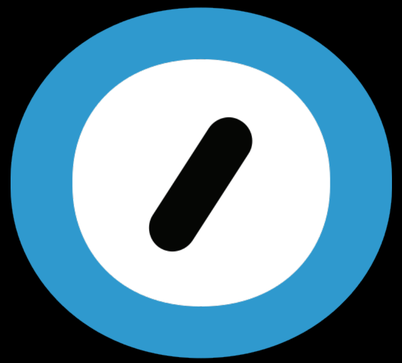
is your app secure?

revelations a Jedi wouldn’t tell you

Kabir Oberai

$ whoami

University of Waterloo



Automattic Messaging
Texts + Beeper

Sun Tzu:

if you know neither the enemy nor yourself, you will succumb in every battle.

if you know yourself but not the enemy, for every victory gained you will also suffer a defeat.

if you know the enemy and know yourself, you need not fear the result of a hundred battles.

⇒ **threat model**

assets

I talk to a server:

- 🔑 api keys
- 👥 other users' data
- 🤐 company secrets
- 💯 uptime

I'm local-first:

- 🔒 file access
- 💪 data integrity
- 📵 device availability
- 💵 purchases

I… exist:

- 📱 users' devices
- 📣 your reputation
- 😶 users' reputation

assets

I talk to a server:

- 🔑 api keys
- 👥 other users' data
- 🤐 company secrets
- 💯 uptime

I'm local-first:

- 🔒 file access
- 💪 data integrity
- 📵 device availability
- 💵 purchases

I… exist:

- 📱 users' devices
- 📣 your reputation
- 😶 users' reputation

assets

confidentiality

- 🔑 api keys
- 👥 other users' data
- 🤐 company secrets
- 🔒 file access

integrity

- 💪 data integrity
- 💵 purchases
- 📣 your reputation
- 😶 users' reputation

availability

- 📱 users' devices
- 💯 uptime
- 📵 device availability

attack surface

all input is untrusted.
all output should be controlled.

} trust boundaries

attack surface

…but Swift!

Swift is a **general-purpose** programming language that's **approachable** for newcomers and **powerful** for experts.

It is **fast**, **modern**, **safe**, and a **joy** to write.

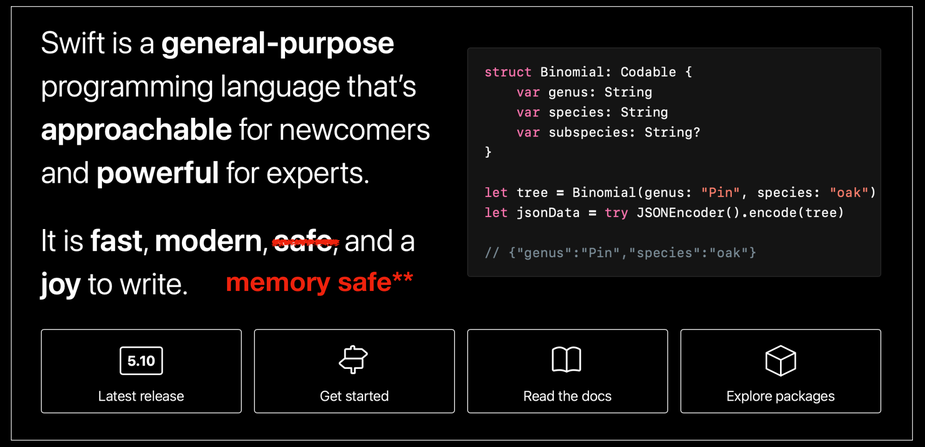
```
struct Binomial: Codable {
    var genus: String
    var species: String
    var subspecies: String?
}

let tree = Binomial(genus: "Pin", species: "oak")
let jsonData = try JSONEncoder().encode(tree)

// {"genus":"Pin","species":"oak"}
```

5.10 Latest release | Get started | Read the docs | Explore packages

attack surface

…but Swift!

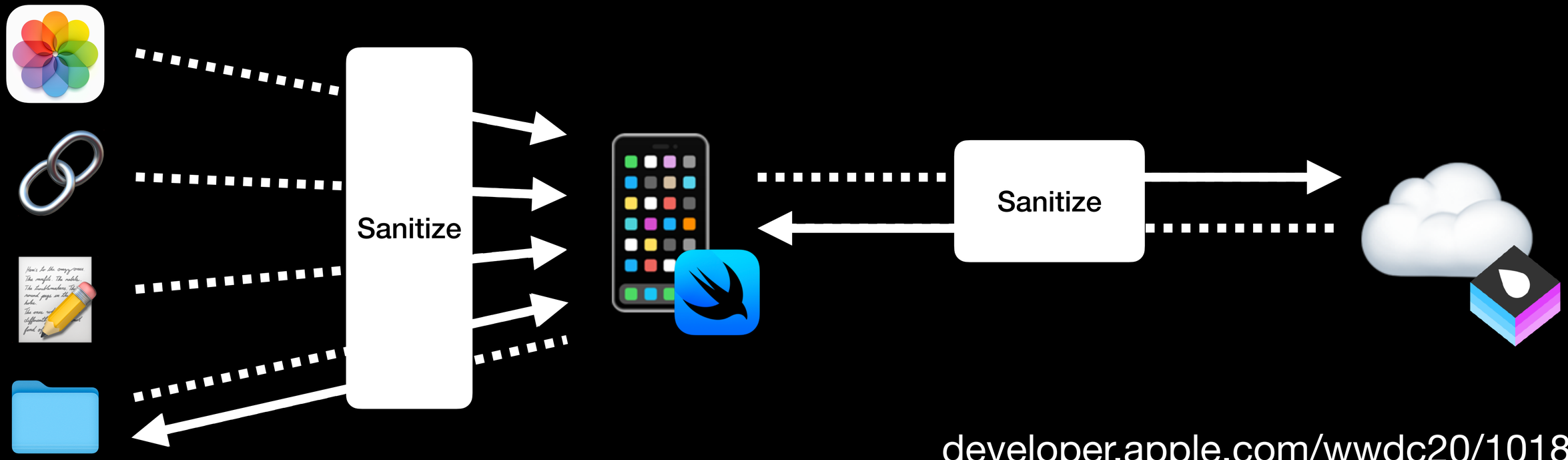
- memory safety != total safety
- not swift all the way down



C image decoder?

filename overwrites something?

attack surface



developer.apple.com/wwdc20/10189

interlude

Yelp

interlude

Yelpepperoni*

*we forgot to hire a legal team

interlude

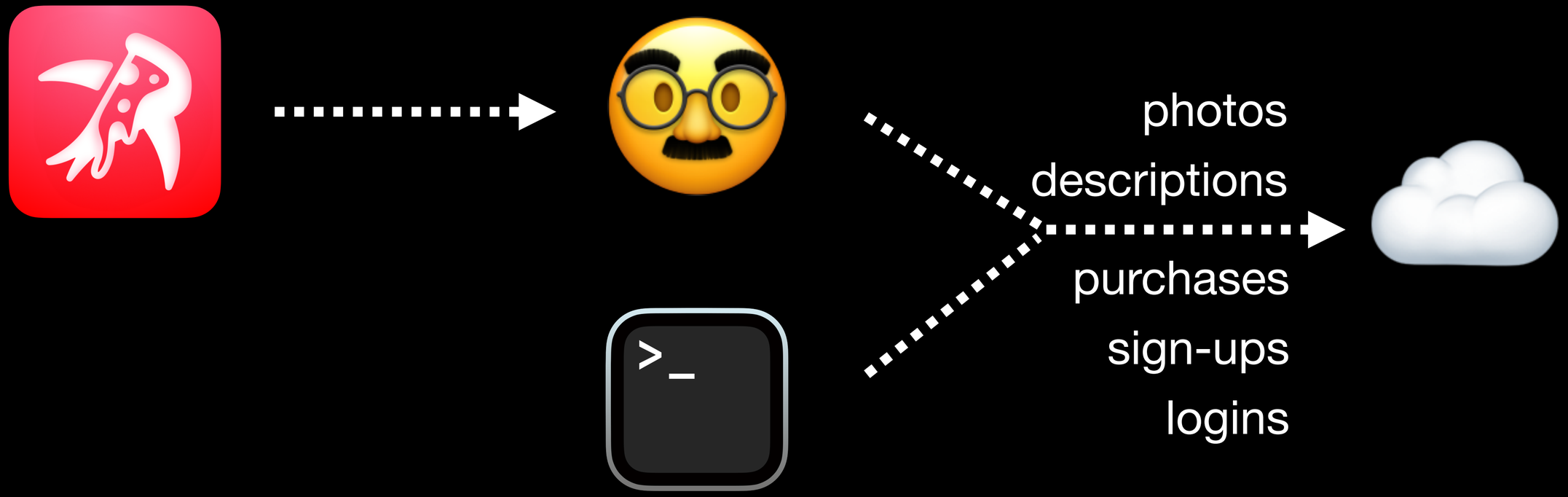
Yelpepperoni

an app for pizzeria discovery!

- pictures and details of pizzerias
- manage pizzerias you own
- classify pizza pictures with cutting-edge pizza detection AI
- buy **YelpeppePROni** for amazing discounts

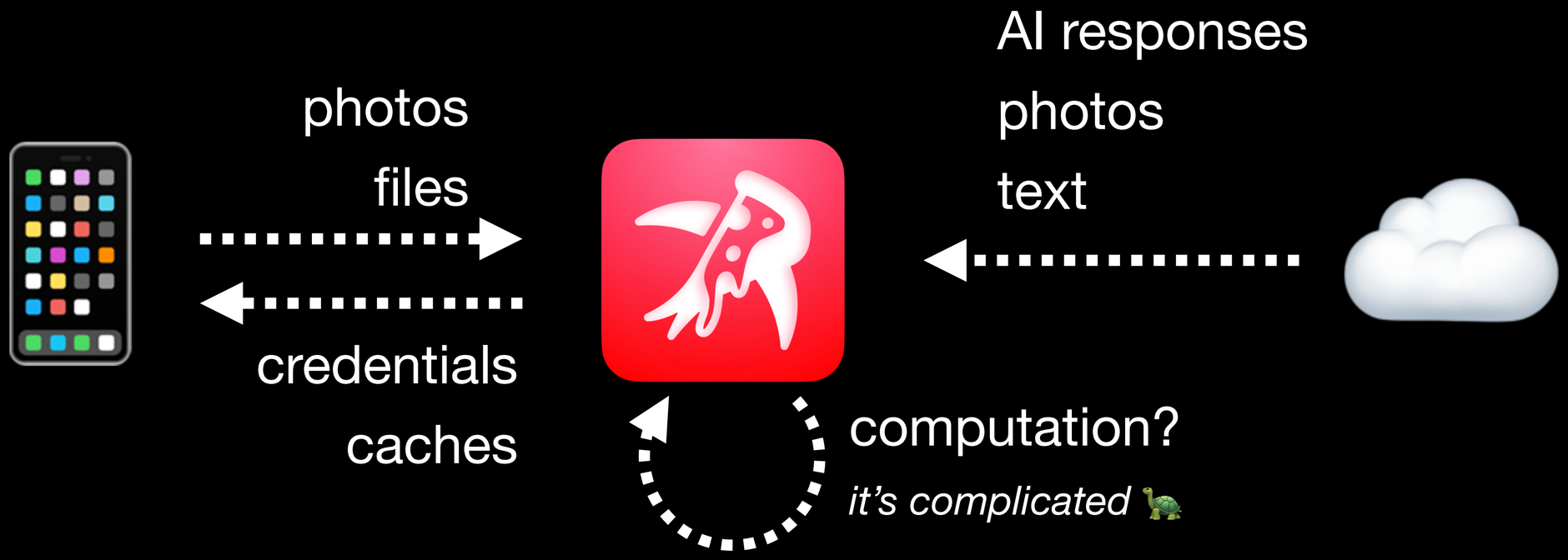
attack surface (server)

attack surface (server)



reminder: input is untrusted! *don't trust the client.*

attack surface (client)



securing integrity

scenario: attacker is the user. attempting to compromise in-app purchases.

securing integrity

```
transaction != nil
```

trust assumptions:

📱 **user hasn’t modified the system**

jailbreaking is game over for the client. but it’s getting harder, so let’s set this aside.

🔧 **user hasn’t modified your app’s code**

can the user modify your app without a jailbreak?

securing integrity

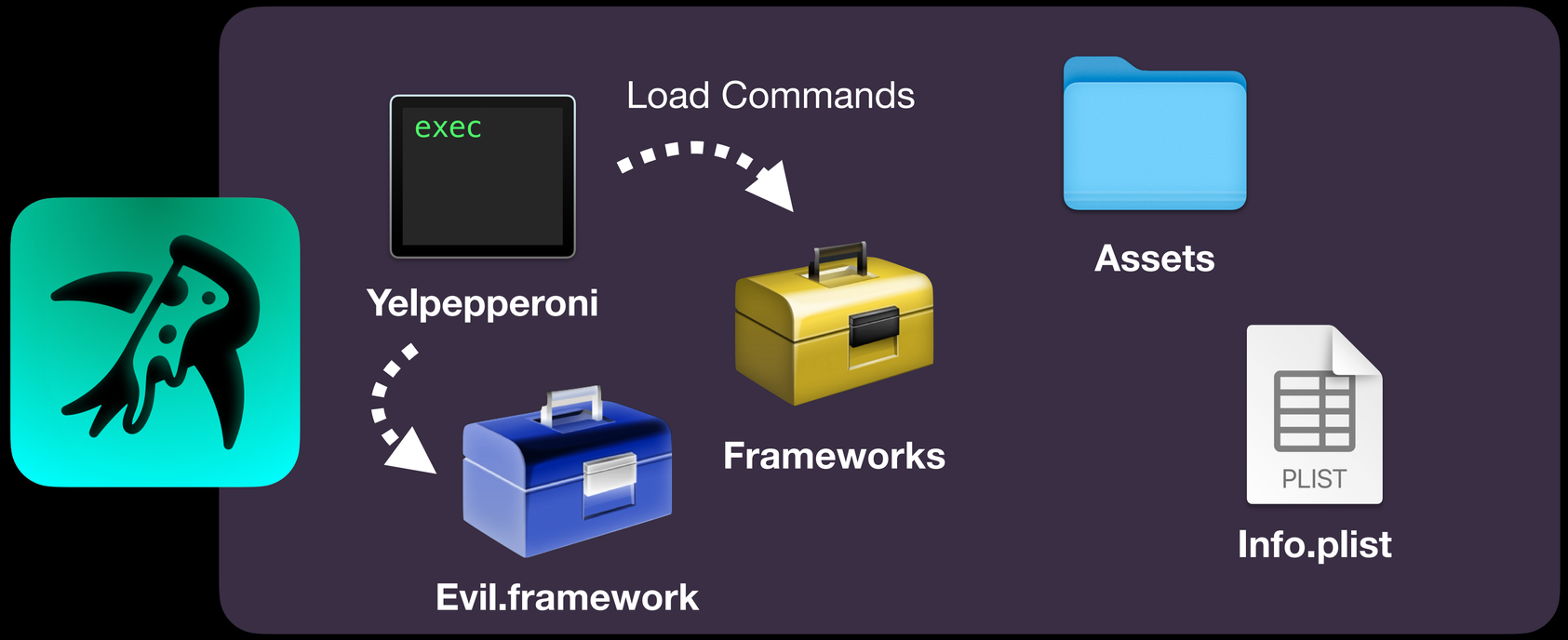
can the user modify your app without a jailbreak?

step 1. extract Yelpepperoni.ipa

step 2. inject

step 3. sideload

securing integrity



step 2. inject

securing integrity

```
+[Transaction all] { ... }
```

ObjC runtime swizzling /
bespoke Swift hackery

Yelpepperoni

```
return [Transaction(purchased: true)]
```

Evil.framework

securing integrity

```
+[Transaction all] { return [Transaction(purchased: true)] }
```

Yelpepperoni

Evil.framework

securing integrity

securing integrity

see also: developer.apple.com/documentation/appstoreservernotifications

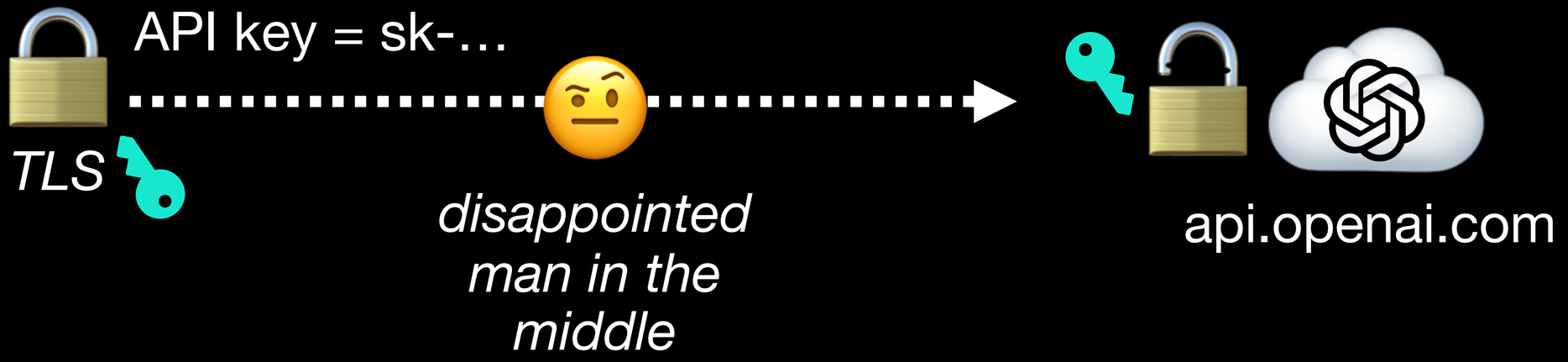
demo

receipt validation

securing confidentiality

scenario: attacker wants to piggyback off your OpenAI access

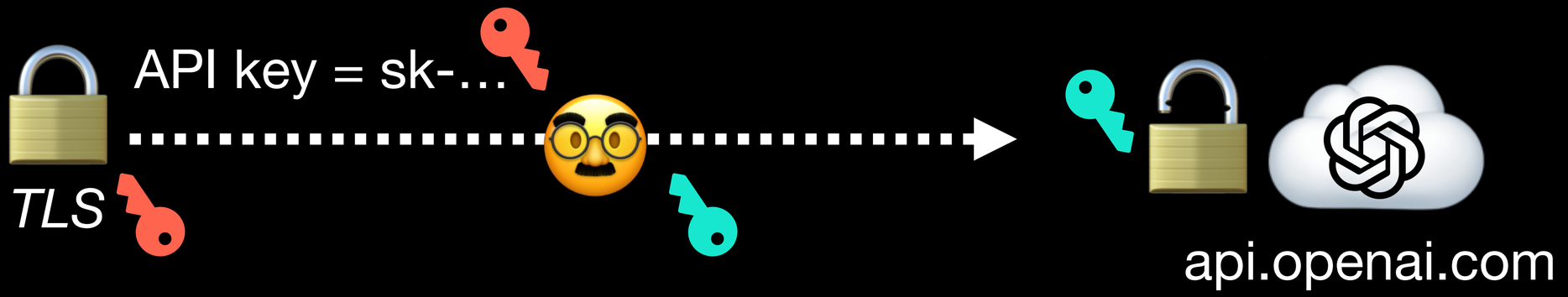
securing confidentiality



securing confidentiality

how do we secure against this?

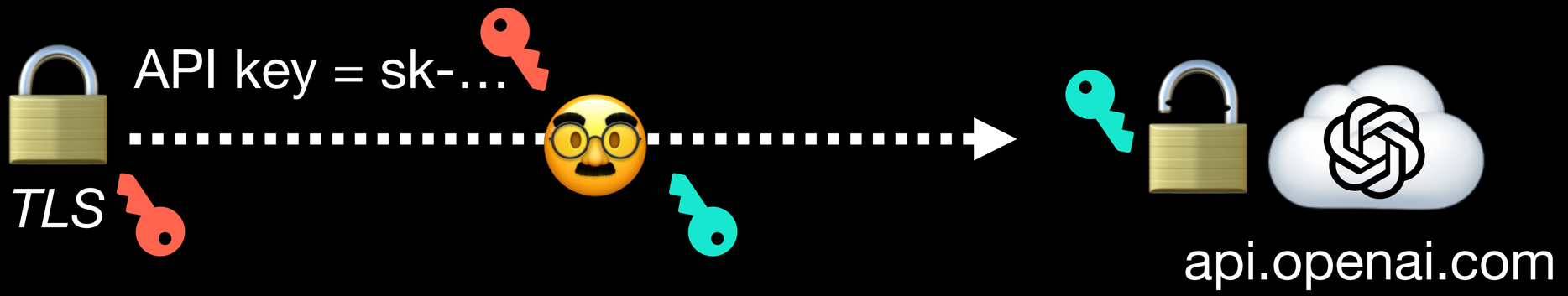
1. OS, please trust 🔑
2. intercept requests: 🔑 ⇔ 🔑



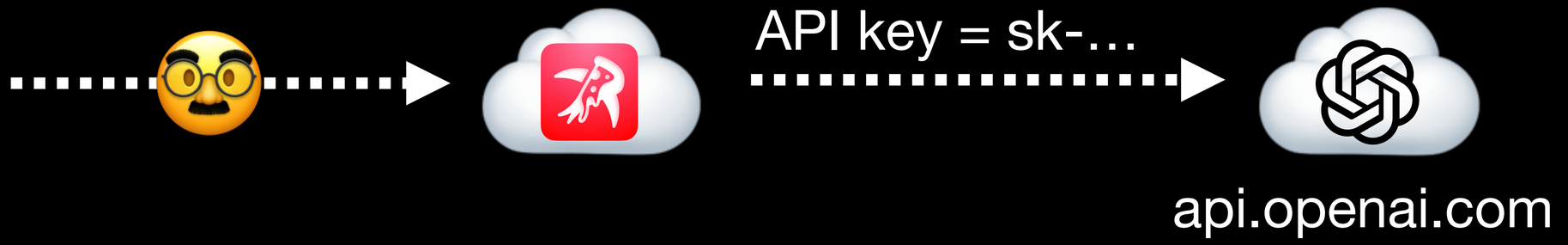
securing confidentiality

DO we secure against this?

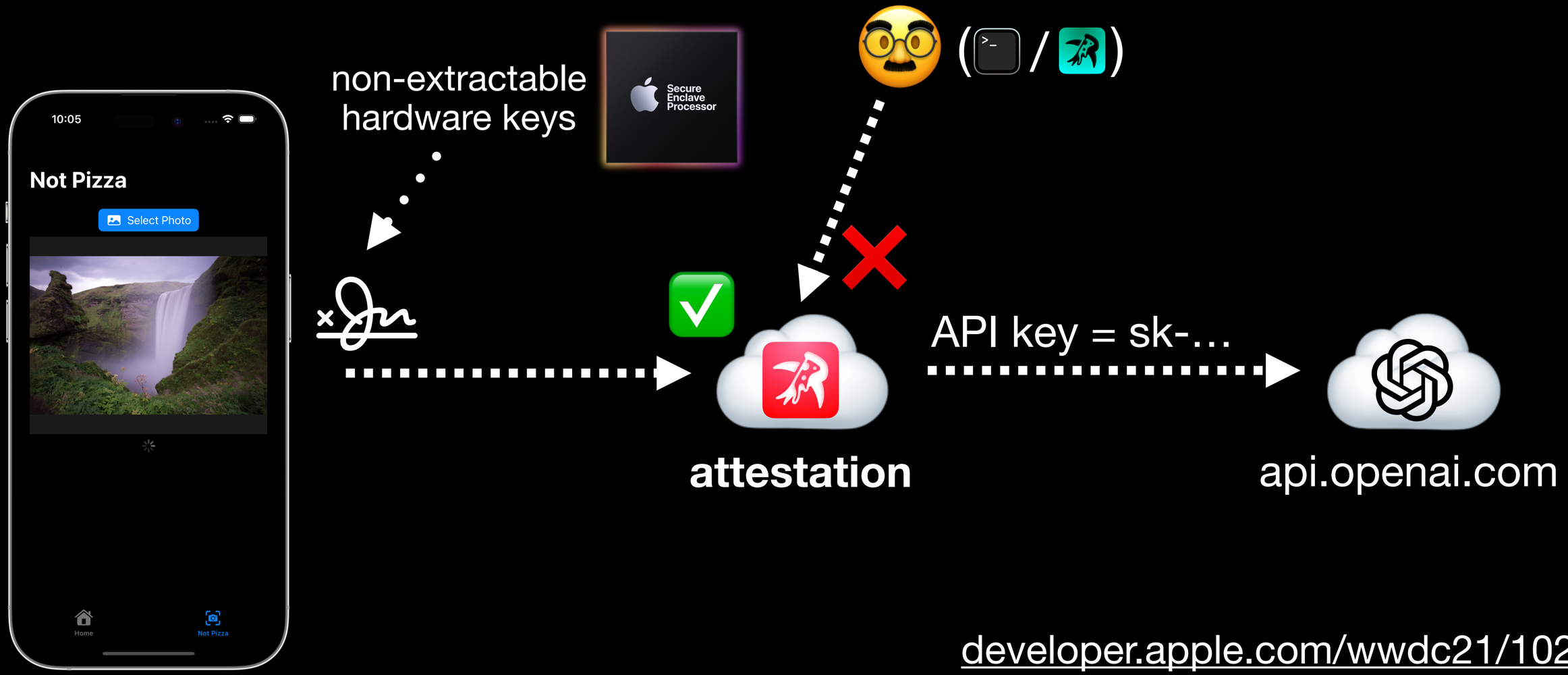
Kerckhoffs's Principle: *one ought to design systems under the assumption that the enemy will immediately gain full familiarity with them.* security through obscurity is bad.



securing confidentiality



securing confidentiality



developer.apple.com/wwdc21/10244

securing confidentiality

non-extractable
hardware keys

attestation

API key = sk-…

api.openai.com

developer.apple.com/wwdc21/10244

the big picture

“don’t trust the client”

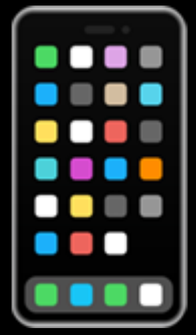
then why bother with client-side security?

different threat models!

securing availability

scenario: user is non-malicious. attacker can upload pizzerias.

securing availability



GET /pizzerias

```
{
  name: "My Pizzeria",
  photos: [
    {
      name: "photo.png",
      id: "123"
    }
  ]
}
```

securing availability


```
{
  name: "photo.png",
  id: "123"
}
```

securing availability


```
let dest = photosDir.appending(
  path: photo.name
)

save(image, to: dest)
```

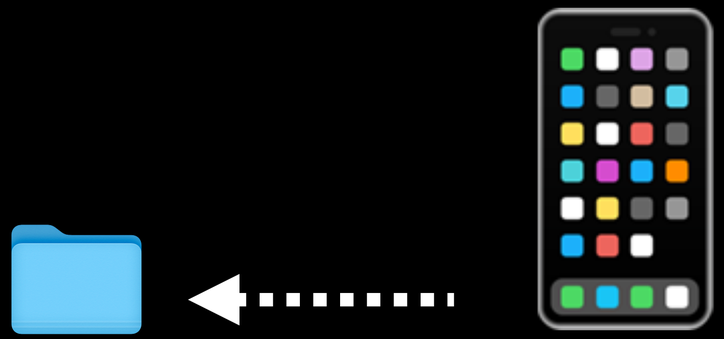
```
{
  name: "photo.png",
  id: "123"
}
```

securing availability


```
let dest = photosDir.appending(
  path: "../../Preferences.plist"
)

save(image, to: dest)
```

```
{
  name: "../../Preferences.plist",
  id: "123"
}
```

securing availability



```
let dest = photosDir.appending(
  path: photo.name
)
```

```
if !dest.isContained(in: photosDir) { throw }
```

```
save(image, to: dest)
```

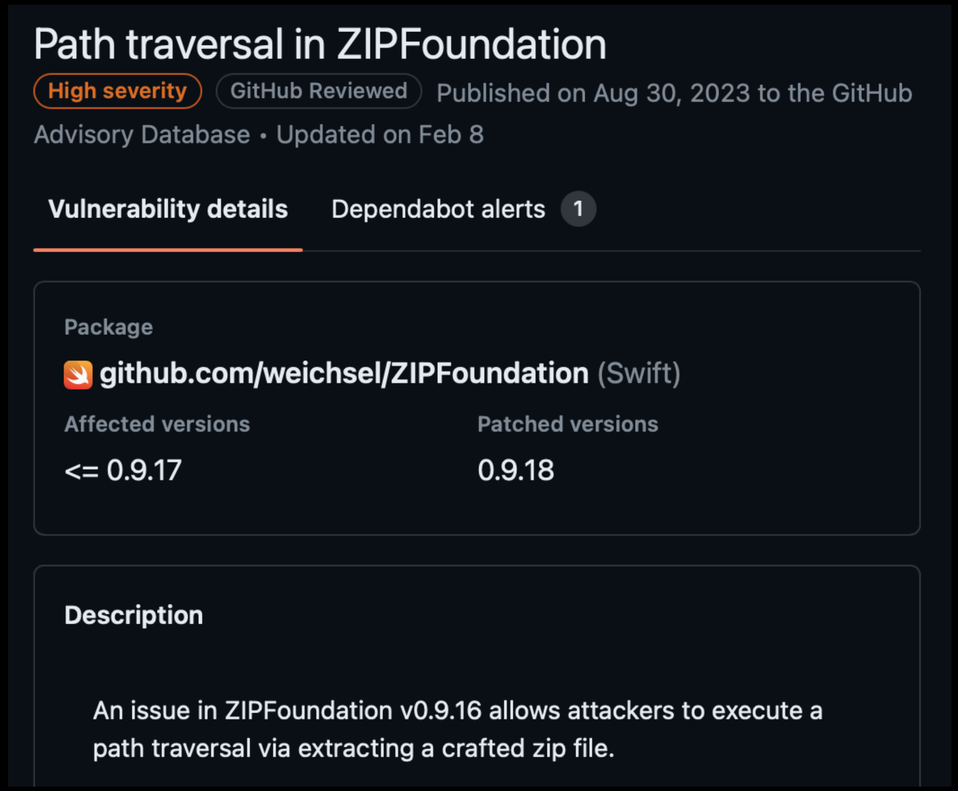
```
extension URL {
  func isContained(in parent: URL) -> Bool {
    let sanitizedParent = URL(
      filePath: parent.path(),
      directoryHint: .isDirectory
    ).standardized

    let sanitizedPath = URL(
      filePath: path()
        .replacingOccurrences(of: "//", with: "/")
    ).standardized

    return sanitizedPath.absoluteString
      .hasPrefix(sanitizedParent.absoluteString)
  }
}
```

```
struct FilePath {
  func lexicallyResolving(_ subpath: FilePath) -> FilePath?
}
```

securing availability

not theoretical: this one's from ZIPFoundation!

```swift
extension URL {
  func isContained(in parent: URL) -> Bool {
    let sanitizedParent = URL(
      filePath: parent.path(),
      directoryHint: .isDirectory
    ).standardized

    let sanitizedPath = URL(
      filePath: path()
        .replacingOccurrences(of: "//", with: "/")
    ).standardized

    return sanitizedPath.absoluteString
      .hasPrefix(sanitizedParent.absoluteString)
  }
}
```



demo

path traversal

conclusion

knowledge is power

🦺 what are you protecting?

📥 what goes in, what comes out?

❗ how can different adversaries violate your assumptions?

get in touch!

kabiroberai@mastodon.social
@kabiroberai
/in/kabiroberai
me@kabiroberai.com

slides & code
ober.ai/dds