# Trust Issues

## trouble in package paradise

@nietaki

## whoami

Jacek Królikowski

## whoami

## Jacek Królikowski

## whoami

Jacek Królikowski

@nietaki

## whoami

Jacek Królikowski

@nietaki

elixir forum

## whoami

Jacek Królikowski

@nietaki

elixir forum

## Disclaimers

## Prototyping

## Prototyping

!= production

## "Reasonable Security"

## "Reasonable Security"

- Infrastructure

## "Reasonable Security"

- Infrastructure
- SSL, auth

## "Reasonable Security"

- Infrastructure
- SSL, auth
- review our code

## "Reasonable Security"

- Infrastructure
- SSL, auth
- review our code
- **not** review the libraries?

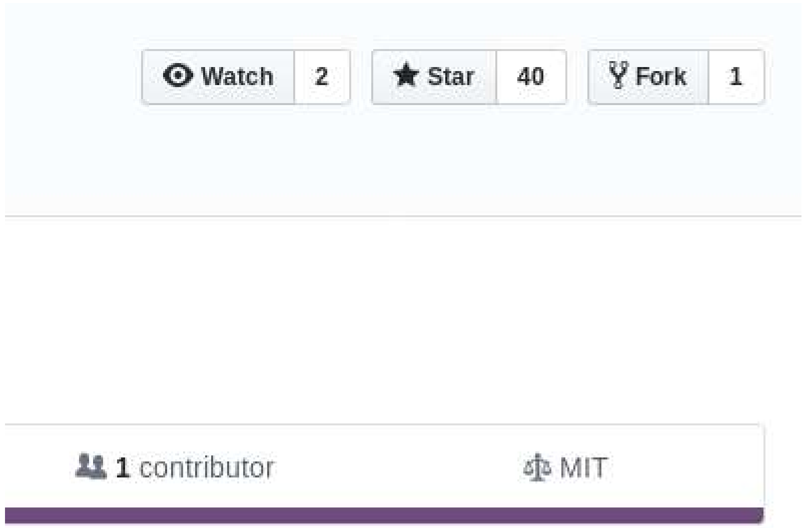
## "Reasonable Security"

- Infrastructure
- SSL, auth
- review our code
- **not** review the libraries?

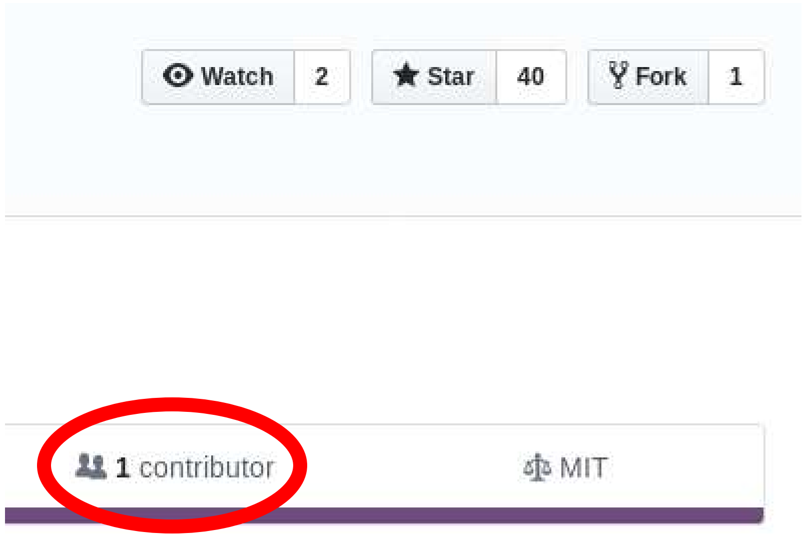
## Why **not** to look into the libraries

## The code gets peer-reviewed on GitHub!

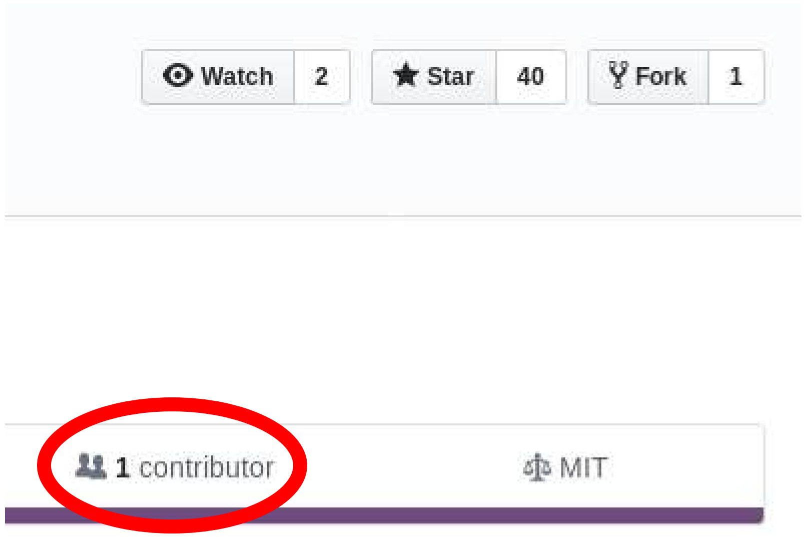
# The code gets peer-reviewed on GitHub!



## The code gets peer-reviewed on GitHub!



# The code gets peer-reviewed on GitHub!



$\overset{?}{=}$

hex.pm/packages/evil_left_pad

## I only use popular packages!

## I only use popular packages!

```
$ wc -l mix.lock
106 mix.lock
```

# I only use popular packages!

```
$ wc -l mix.lock
106 mix.lock
```

```
  "unicode_util_compat": {:hex, :unicode_util_compat, "0.3.1",
"a1f612a7b512638634a603c8f401892afbf99b8ce93a45041f8aaca99cadb85e",
[:rebar3], [], "hexpm"},
  "unsafe": {:hex, :unsafe, "1.0.0",
"7c21742cd05380c7875546b023481d3a26f52df8e5dfedcb9f958f322baae305",
[:mix], [], "hexpm"},
  "uuid": {:hex, :uuid, "1.1.8",
"e22fc04499de0de3ed1116b770c7737779f226ceefa0badb3592e64d5cfb4eb9",
[:mix], [], "hexpm"},
```

## I only use popular packages!

```
$ wc -l mix.lock
106 mix.lock
```

```
  "unicode_util_compat": {:hex, :unicode_util_compat, "0.3.1",
"a1f612a7b512638634a603c8f401892afbf99b8ce93a45041f8aaca99cadb85e",
[:rebar3], [], "hexpm"},
  "unsafe": {:hex, :unsafe, "1.0.0",
"7c21742cd05380c7875546b023481d3a26f52df8e5dfedcb9f958f322baae305",
[:mix], [], "hexpm"},
  "uuid": {:hex, :uuid, "1.1.8",
"e22fc04499de0de3ed1116b770c7737779f226ceefa0badb3592e64d5cfb4eb9",
[:mix], [], "hexpm"},
```

## I'm too small to be a target!

# I'm too small to be a target!

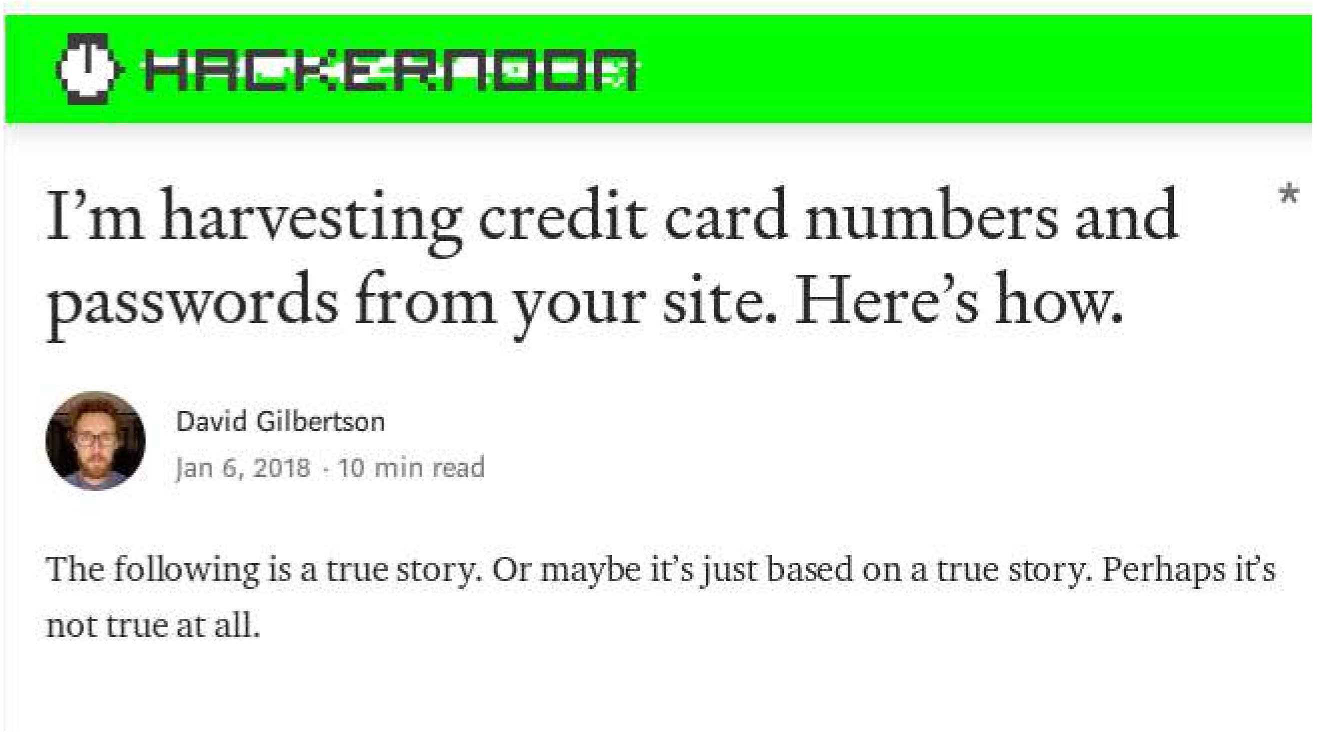
```
iex(1)> :code.all_loaded() |>
...(1)> Enum.map(fn {module, _path} -> module end) |>
...(1)> Enum.filter(fn module ->
...(1)>   behaviours = Keyword.get(module.module_info[:attributes], :behaviour, [])
...(1)>   Ecto.Repo in behaviours
...(1)> end)

[MyApp.Repo]
```

## I would spot it if I was getting hacked!

## I would spot it if I was getting hacked!

## I would spot it if I was getting hacked!


HACKERNOON

I'm harvesting credit card numbers and passwords from your site. Here's how.

David Gilbertson

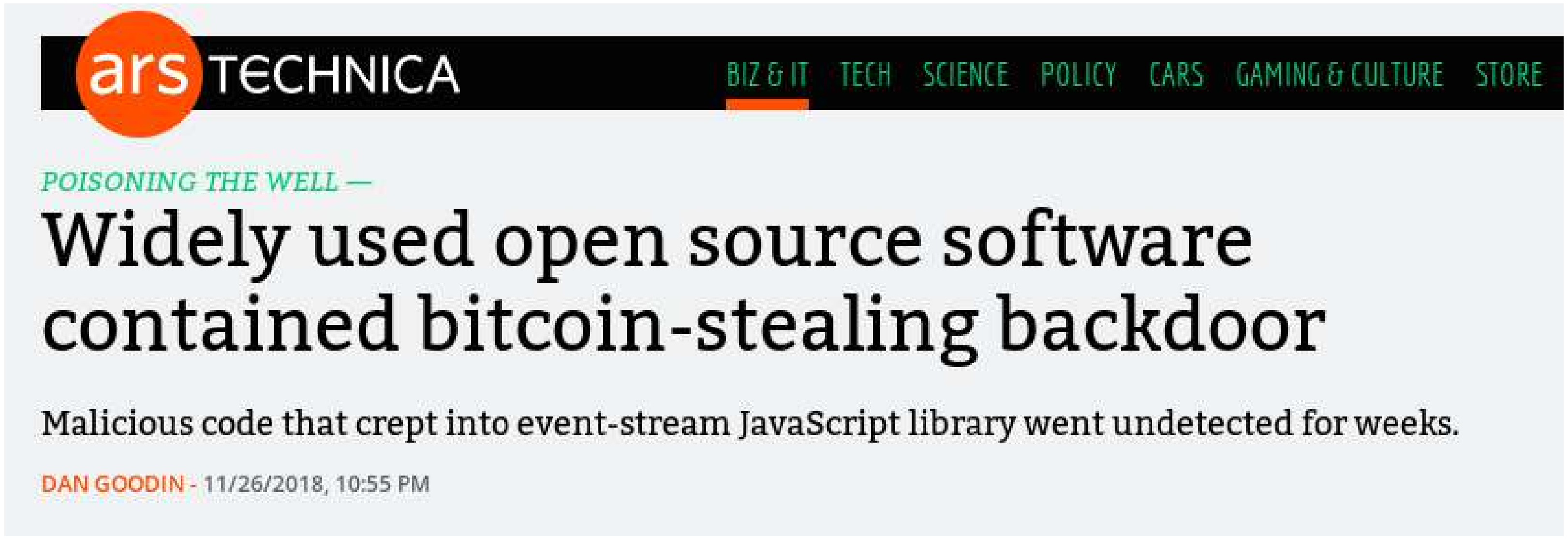
Jan 6, 2018 · 10 min read

The following is a true story. Or maybe it's just based on a true story. Perhaps it's not true at all.

https://hackernoon.com/im-harvesting-credit-card-numbers-and-passwords-from-your-site-here-s-how-9a8cb347c5b5

## Attacks don't happen in practice!

## Attacks don't happen in practice!



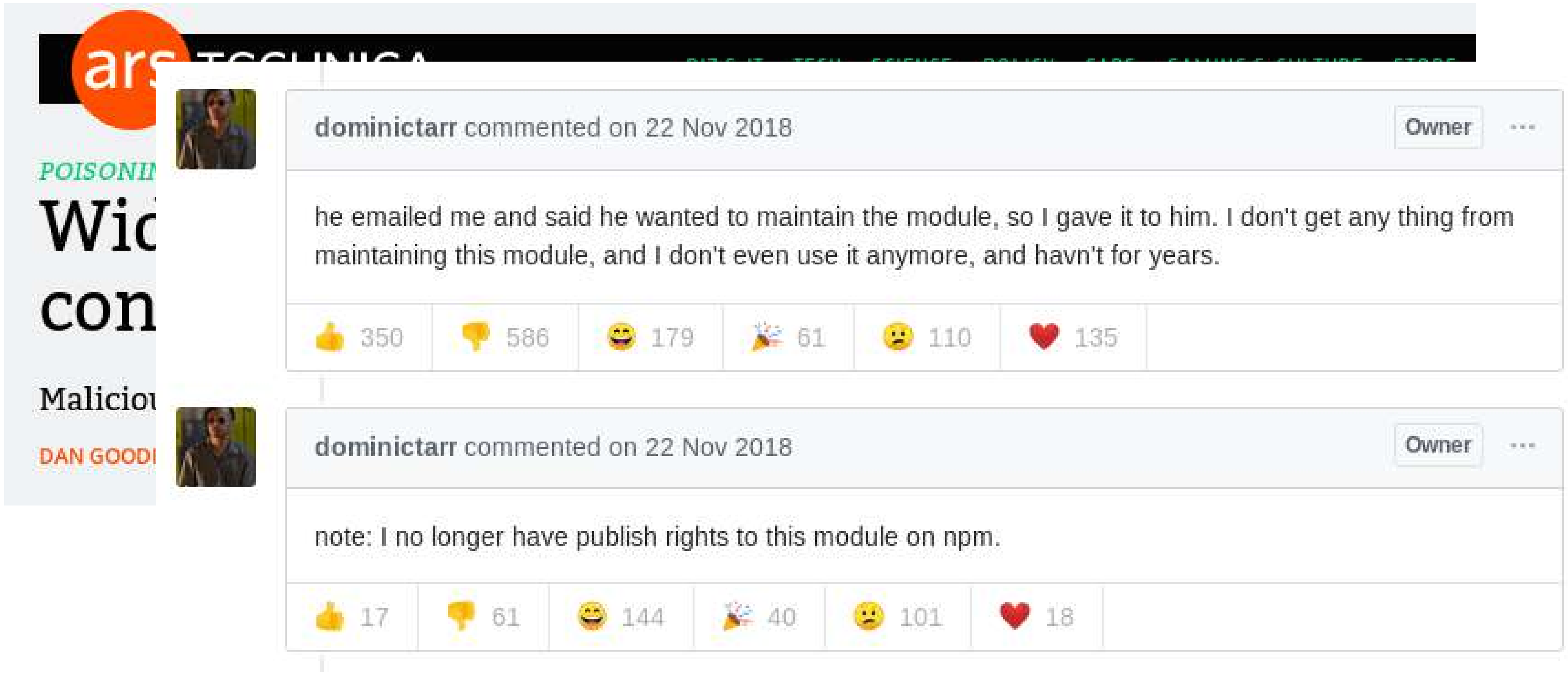
POISONING THE WELL —

Widely used open source software contained bitcoin-stealing backdoor

Malicious code that crept into event-stream JavaScript library went undetected for weeks.

DAN GOODIN - 11/26/2018, 10:55 PM

# Attacks don't happen in practice!



**dominictarr** commented on 22 Nov 2018

he emailed me and said he wanted to maintain the module, so I gave it to him. I don't get any thing from maintaining this module, and I don't even use it anymore, and havn't for years.

👍 350 👎 586 😄 179 🎉 61 😕 110 ❤️ 135

**dominictarr** commented on 22 Nov 2018

note: I no longer have publish rights to this module on npm.

👍 17 👎 61 😄 144 🎉 40 😕 101 ❤️ 18

https://github.com/dominictarr/event-stream/issues/116
https://arstechnica.com/information-technology/2018/11/hacker-backdoors-widely-used-open-source-software-to-steal-bitcoin/

## MIT License (excerpt)

> *“ THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, (...) INCLUDING BUT NOT LIMITED TO (...) FITNESS FOR A PARTICULAR PURPOSE (...).*
>
> *IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY (...) ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE (...).*

https://opensource.org/licenses/MIT
https://gist.github.com/dominictarr/9fd9c1024c94592bc7268d36b8d83b3a

## So what can we do?

## So what can we do?

> *“ Rule 6: Be proactively paranoid - paranoia does not work retroactively*

https://www.youtube.com/watch?v=S8GPTvq1m-w
(nsfw language)

## What can we do?

## What can we do?

- Stop using external libraries 👎👎👎

## What can we do?

- Stop using external libraries 👎👎👎
- Regularly read all our dependencies 👎👎

# What can we do?

- Stop using external libraries 👎👎👎
- Regularly read all our dependencies 👎👎
- Get dependencies directly from git 👎

## What can we do?

- Stop using external libraries 👎👎👎
- Regularly read all our dependencies 👎👎
- Get dependencies directly from git 👎
- Force hex.pm to do the verification for us 👎

## What can we do?

- Stop using external libraries 👎👎👎
- Regularly read all our dependencies 👎👎
- Get dependencies directly from git 👎
- Force hex.pm to do the verification for us 👎
- "Impound" all your dependencies 😐

# What can we do?

- Stop using external libraries 👎👎👎
- Regularly read all our dependencies 👎👎
- Get dependencies directly from git 👎
- Force hex.pm to do the verification for us 👎
- "Impound" all your dependencies 😐
- Static analysis of dependencies 😶

## What can we do?

- Stop using external libraries 👎👎👎
- Regularly read all our dependencies 👎👎
- Get dependencies directly from git 👎
- Force hex.pm to do the verification for us 👎
- "Impound" all your dependencies 😐
- Static analysis of dependencies 😶
- Something else? 🤔

## What do we need?

## What do we need?

- rely on manual reviews

## What do we need?

- rely on manual reviews
- balance risk vs effort

## What do we need?

- rely on manual reviews
- balance risk vs effort
- rely on community

## What do we need?

- rely on manual reviews
- balance risk vs effort
- rely on community
- explicit reviews

## What do we need?

- rely on manual reviews
- balance risk vs effort
- rely on community
- explicit reviews
- simple trust model

## What do we need?

- rely on manual reviews
- balance risk vs effort
- rely on community
- explicit reviews
- simple trust model
- secure by design

## What do we need?

- rely on manual reviews
- balance risk vs effort
- rely on community
- explicit reviews
- simple trust model
- secure by design
- OK for individual devs
- OK for companies

## What do we need?

- rely on manual reviews
- balance risk vs effort
- rely on community
- explicit reviews
- simple trust model
- secure by design
- OK for individual devs
- OK for companies
- generalizable (!)

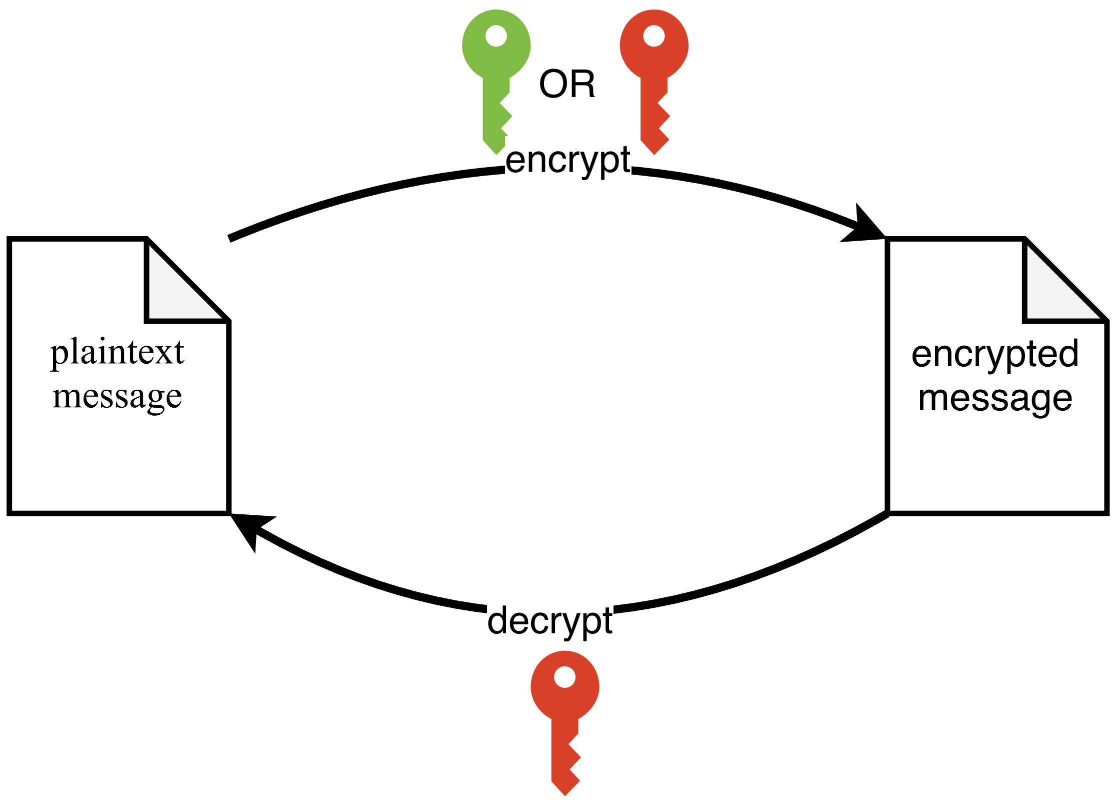
## What do we need?

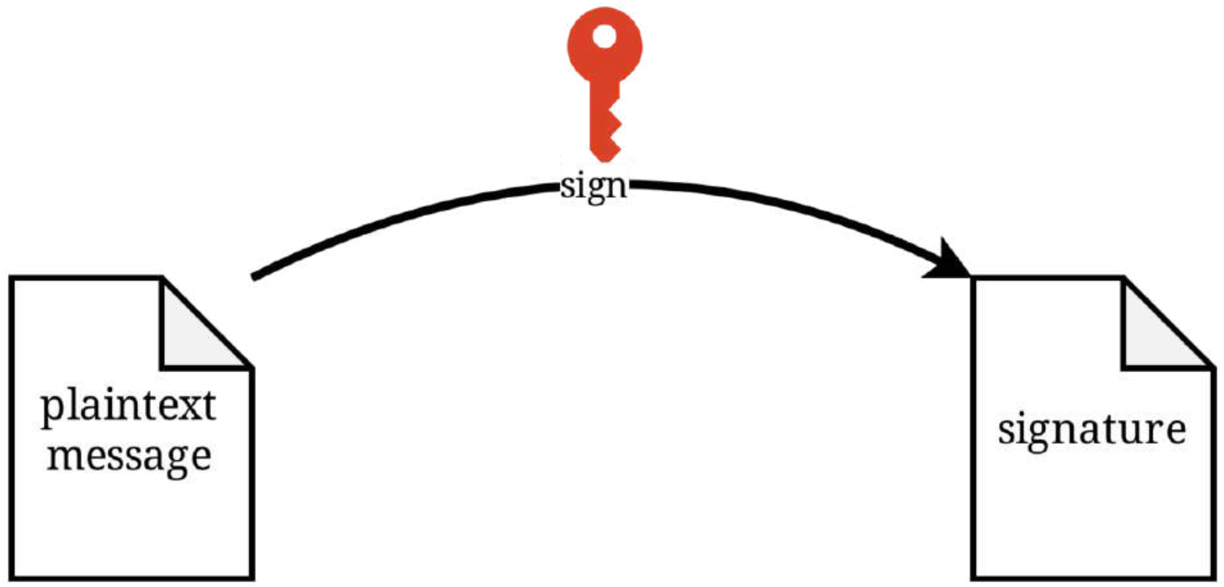
## What do we need?

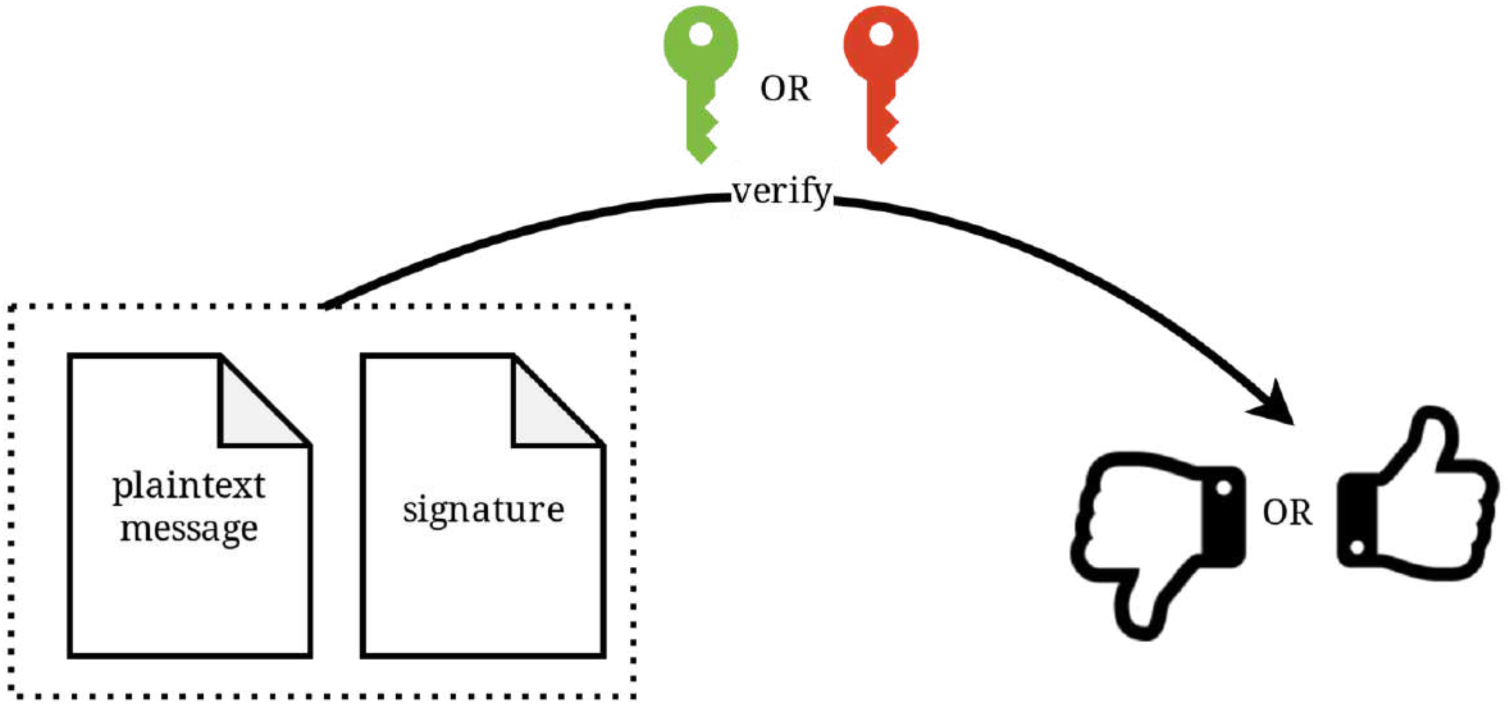
> *Your own decentralized package security audit network of trust*

## Public key cryptography refresher

# Public key cryptography refresher

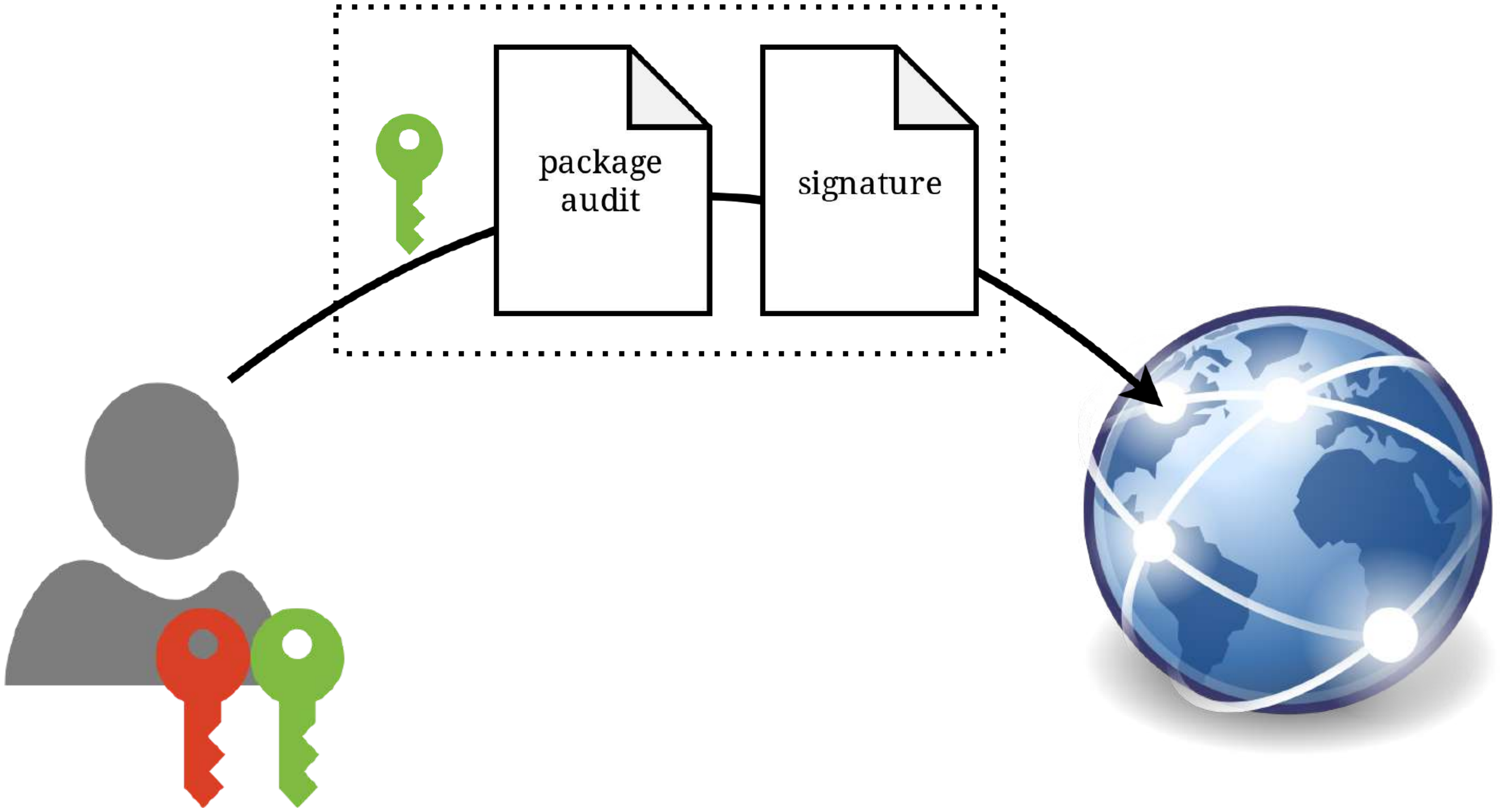



sign
plaintext message
signature

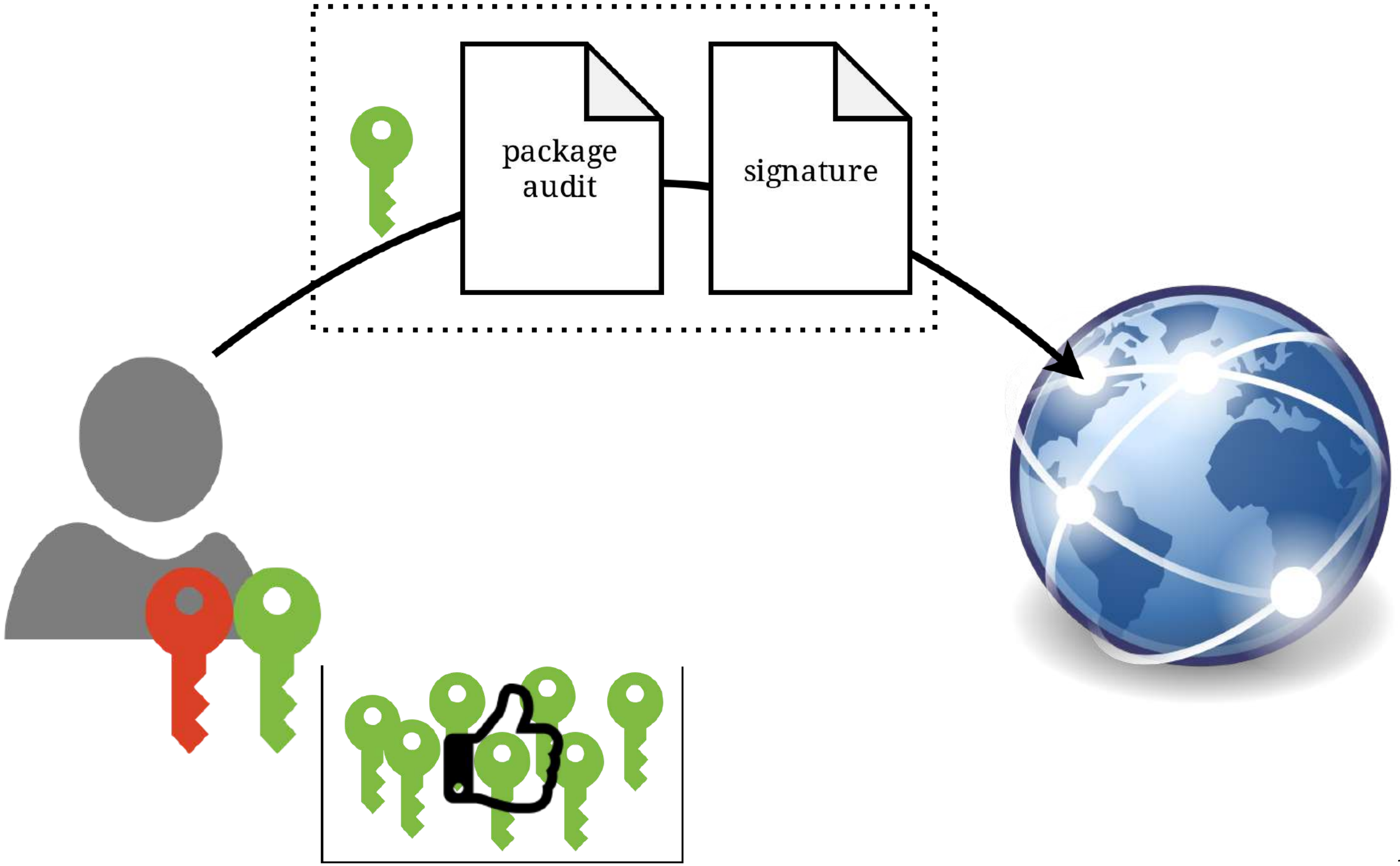

OR
verify
plaintext message
signature
OR

## How does Hoplon work?

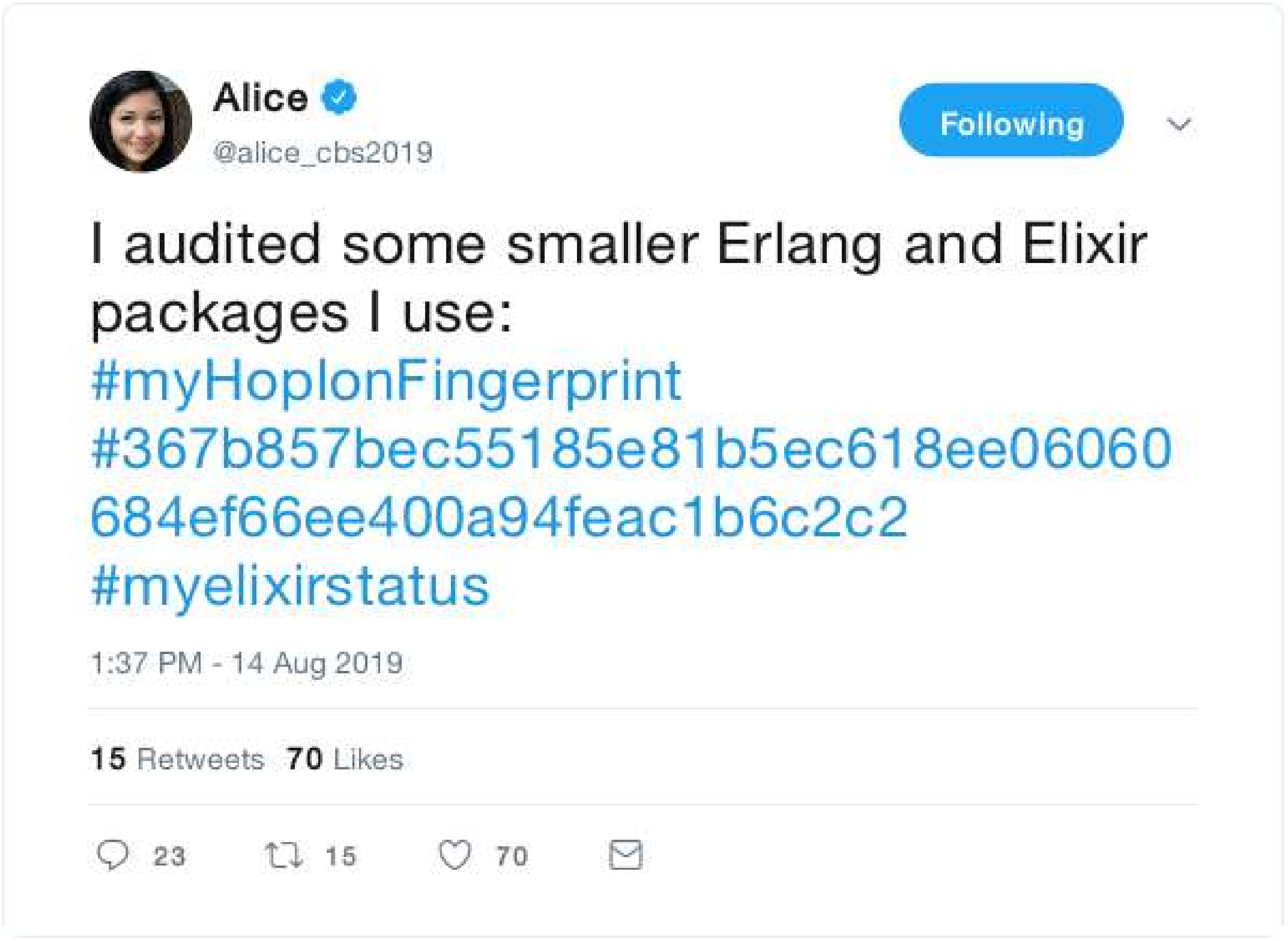
# How does Hoplon work?



# How does Hoplon work?



## Demo



Alice
@alice_cbs2019

Following

I audited some smaller Erlang and Elixir packages I use: #myHoplonFingerprint #367b857bec55185e81b5ec618ee06060684ef66ee400a94feac1b6c2c2 #myelixirstatus

1:37 PM - 14 Aug 2019

**15** Retweets **70** Likes

23 15 70

https://thispersondoesnotexist.com/

## Technical details

## Technical details

- No dependencies (!)
  - :public_key, :asn1ct, :httpc

## Technical details

- No dependencies (!)
  - :public_key, :asn1ct, :httpc
- ASN.1 DER message encoding

## Technical details

- No dependencies (!)
    - :public_key, :asn1ct, :httpc
- ASN.1 DER message encoding
- openssl-compatible 4096 bit, password protected, RSA keys

## Technical details

- No dependencies (!)
  - :public_key, :asn1ct, :httpc
- ASN.1 DER message encoding
- openssl-compatible 4096 bit, password protected, RSA keys
- sha-256 key fingerprints

## Technical details

- No dependencies (!)
    - :public_key, :asn1ct, :httpc
- ASN.1 DER message encoding
- openssl-compatible 4096 bit, password protected, RSA keys
- sha-256 key fingerprints
- server: Raxx/Ace

## Technical details

- No dependencies (!)
  - :public_key, :asn1ct, :httpc
- ASN.1 DER message encoding
- openssl-compatible 4096 bit, password protected, RSA keys
- sha-256 key fingerprints
- server: Raxx/Ace 🤘

## Why should we trust you?

## Why should we trust you?

- You shouldn't!

## Why should we trust you?

- You shouldn't!
- Can't the server withhold audits?

## Why should we trust you?

- You shouldn't!
- Can't the server withhold audits?
    - theoretically...

## Why should we trust you?

- You shouldn't!
- Can't the server withhold audits?
  - theoretically...
  - we can fix it

## Why should we trust you?

- You shouldn't!
- Can't the server withhold audits?
    - theoretically...
    - we can fix it
    - validate the workflow!

I need you!

## Follow-up work

## Follow-up work

- cleanup

## Follow-up work

- cleanup
- Erlang/rebar support

## Follow-up work

- cleanup
- Erlang/rebar support
- key revocation

## Follow-up work

- cleanup
- Erlang/rebar support
- key revocation
- features
    - utility APIs?
    - diffs between versions?
    - transitive trust?

# Thank You!

github.com/nietaki/hoplon

slides.com/nietaki/trust-issues

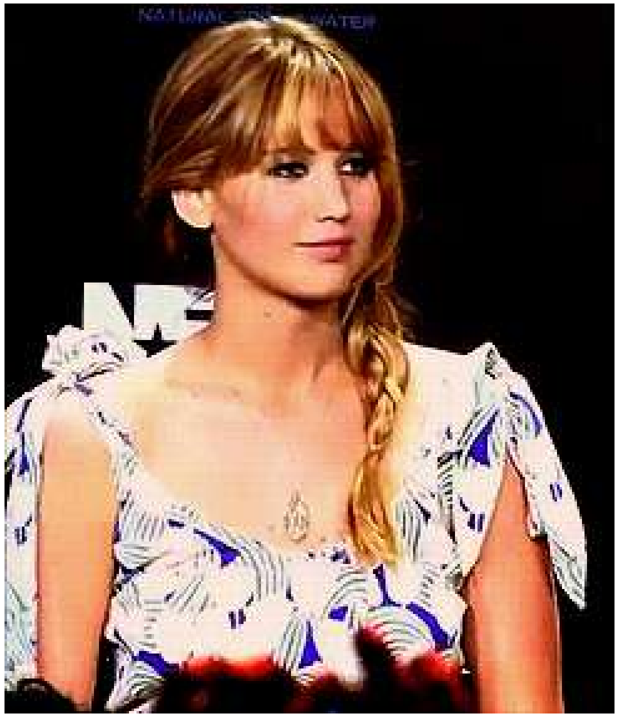
## Bonus slides...

## Learned along the way

- Cryptography
- ASN.1 is cool
- stream_data is cool
    - but I still want to play with PropEr
- Raxx and Raxx.Kit is very productive!
- Testing Mix Tasks (with I/O) can be alright

## I would spot it if I was getting hacked!

## I would spot it if I was getting hacked!



## I would spot it if I was getting hacked!

- https://hackernoon.com/im-harvesting-credit-card-numbers-and-passwords-from-your-site-here-s-how-9a8cb347c5b5

## I would spot it if I was getting hacked!

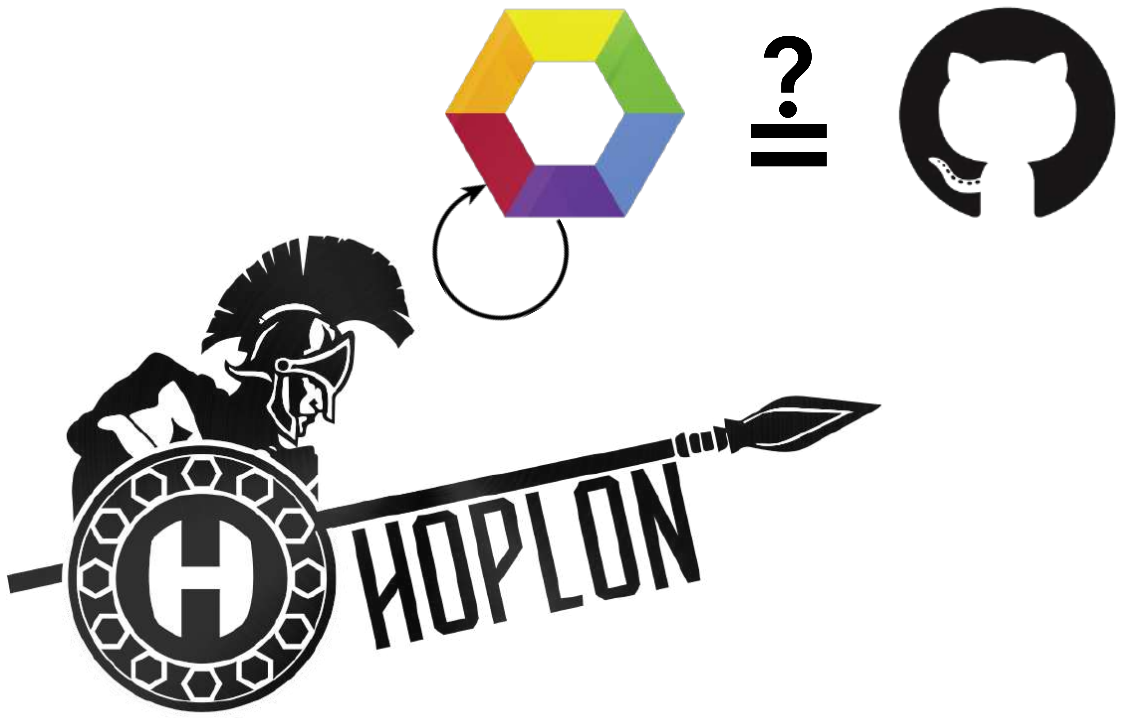
- https://hackernoon.com/im-harvesting-credit-card-numbers-and-passwords-from-your-site-here-s-how-9a8cb347c5b5
- https://gist.github.com/nietaki/4a842365e648f5ad73b4784ef05695c9

## What can we do? (2018)

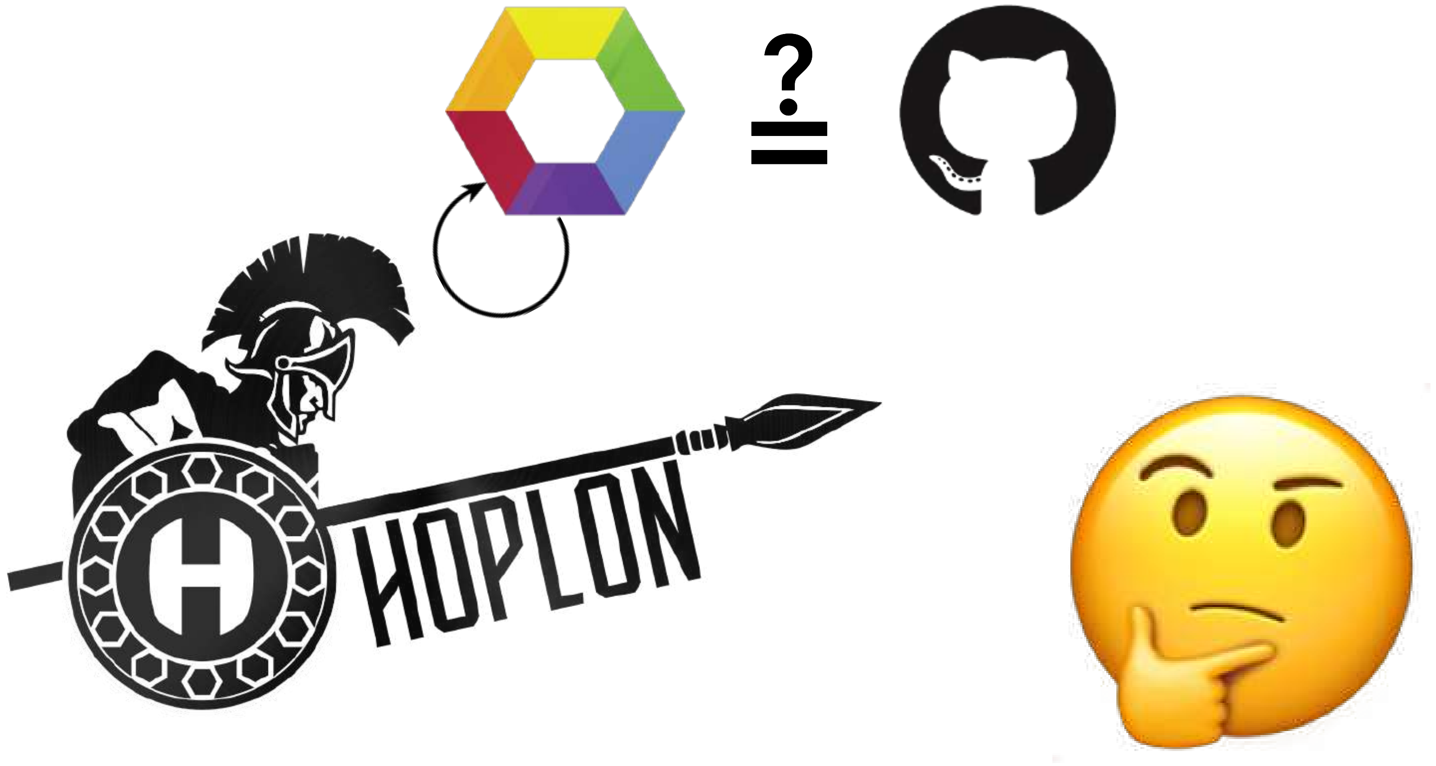
## What can we do? (2018)

$\overset{?}{=}$

# What can we do? (2018)



## What can we do? (2018)



## What's an audit?

https://github.com/nietaki/hoplon/blob/master/lib/HoplonMessages.asn1

## What's an audit?

- package definition
  - ecosystem (hex.pm)
  - name
  - hash
  - version (for convenience)

https://github.com/nietaki/hoplon/blob/master/lib/HoplonMessages.asn1

## What's an audit?

- package definition
  - ecosystem (hex.pm)
  - name
  - hash
  - version (for convenience)
- key fingerprint

https://github.com/nietaki/hoplon/blob/master/lib/HoplonMessages.asn1

## What's an audit?

- package definition
  - ecosystem (hex.pm)
  - name
  - hash
  - version (for convenience)
- key fingerprint
- verdict (dangerous|suspicious|lgtm|safe)

https://github.com/nietaki/hoplon/blob/master/lib/HoplonMessages.asn1

## What's an audit?

- package definition
  - ecosystem (hex.pm)
  - name
  - hash
  - version (for convenience)
- key fingerprint
- verdict (dangerous|suspicious|lgtm|safe)
- timestamp

https://github.com/nietaki/hoplon/blob/master/lib/HoplonMessages.asn1

## What's an audit?

- package definition
  - ecosystem (hex.pm)
  - name
  - hash
  - version (for convenience)
- key fingerprint
- verdict (dangerous|suspicious|lgtm|safe)
- timestamp
- comment

https://github.com/nietaki/hoplon/blob/master/lib/HoplonMessages.asn1

## How does Hoplon work?

## How does Hoplon work?

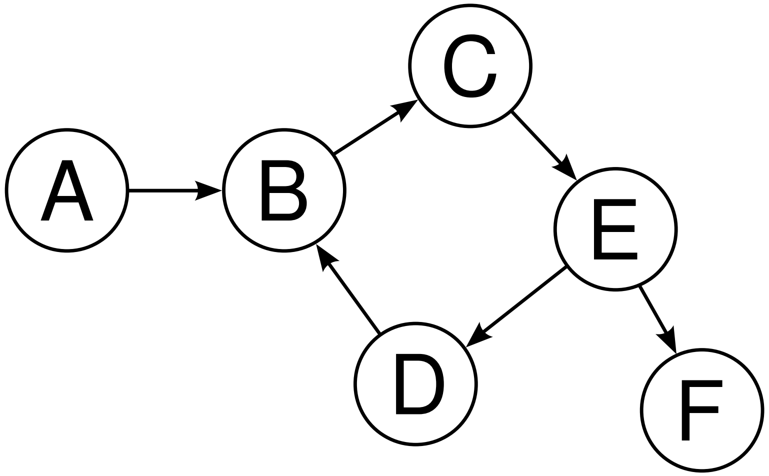
- Each of us has a public+private key pair

## How does Hoplon work?

- Each of us has a public+private key pair
- Each of us can publish a signed "audit" of a package

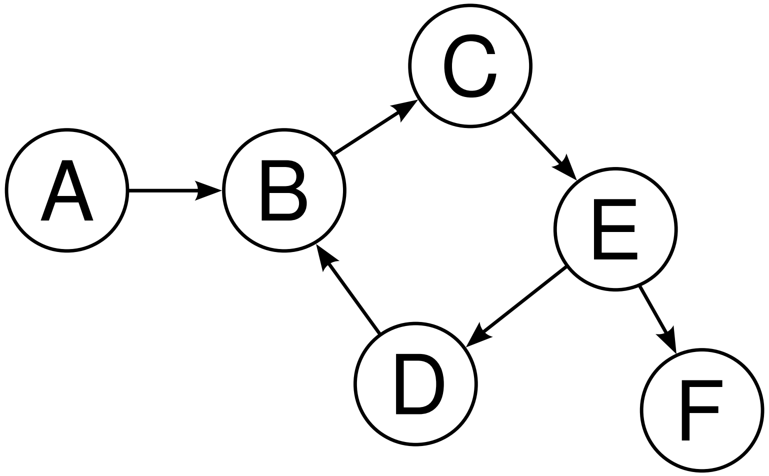
# How does Hoplon work?

- Each of us has a public+private key pair
- Each of us can publish a signed "audit" of a package
- Each of us trusts a set of people (public keys, fingerprints)



# How does Hoplon work?

- Each of us has a public+private key pair
- Each of us can publish a signed "audit" of a package
- Each of us trusts a set of people (public keys, fingerprints)
- You download (and verify!) audits for the packages you want to use, authored by the people you trust


$$\begin{tikzcd}
 & & C \arrow[dr] & \\
A \arrow[r] & B \arrow[ur] & & E \arrow[dl] \arrow[dr] \\
 & & D \arrow[ul] & & F
\end{tikzcd}$$

# How does Hoplon work?

- Each of us has a public+private key pair
- Each of us can publish a signed "audit" of a package
- Each of us trusts a set of people (public keys, fingerprints)
- You download (and verify!) audits for the packages you want to use, authored by the people you trust
- It can run in CI