Teaching an old dog new tricks: Reusing security solutions in novel domains

Graham Bleaney
Security Engineer

Meta

Agenda

About Me

Graham Bleaney

Security Engineer at ~~Facebook~~ Meta

Focus on Python Security

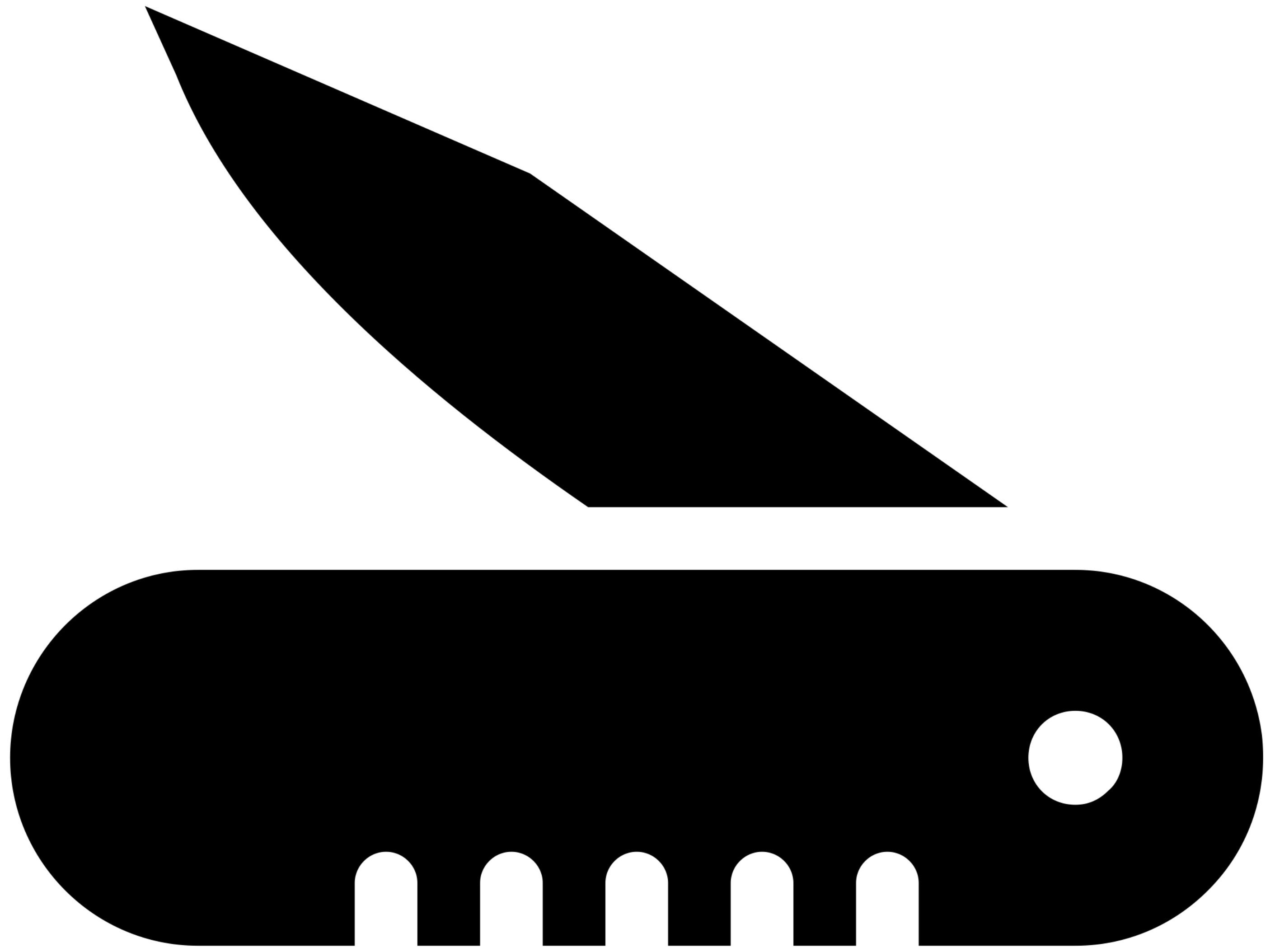
Motivation and Context

SECURITY AT META

We build *generalized* solution to help us *shift left* and *solve problems at scale*

Generalized Solution

Solutions like bug bounty and static analyzers can adapt to the next bug we don’t yet know exist



Shifting Left

Prevented > Found Automatically > Found Manually > Found Externally > Never Found > Exploited

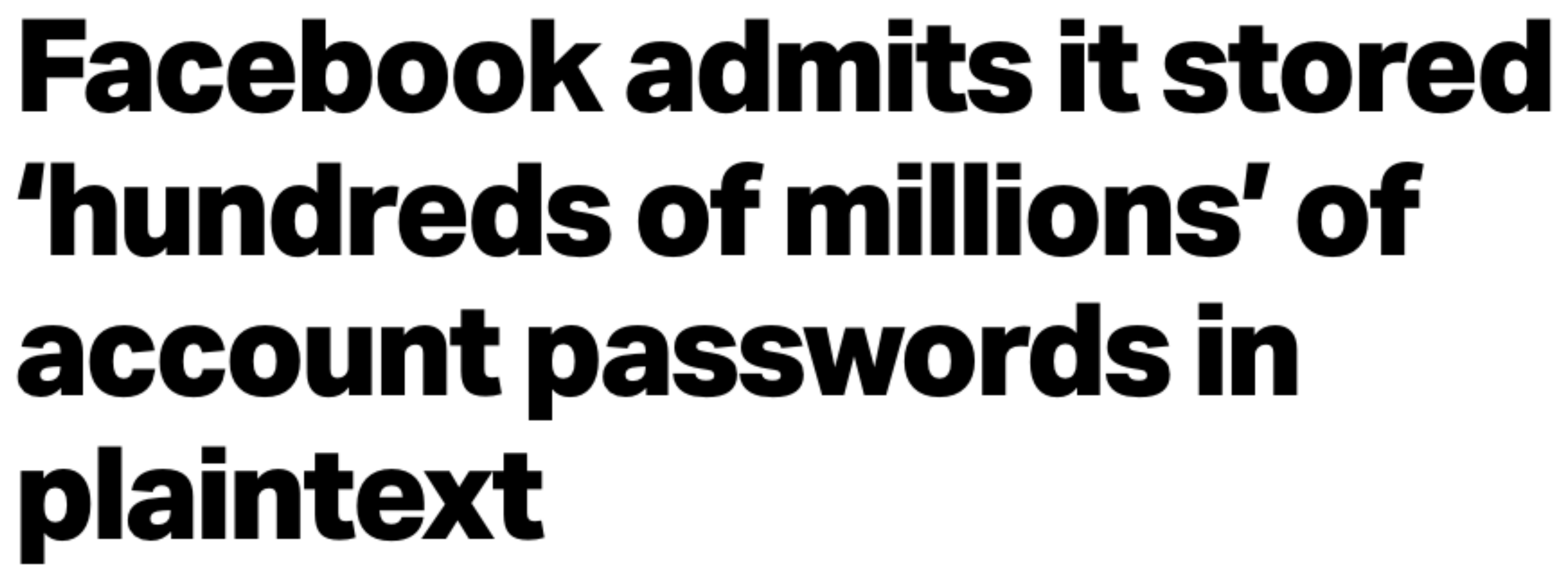
Solving problems at scale

To deal with the size of our codebase, we used tooling to find half of all bugs in 2021

In 2019, we detected a mistake

Initially found in a code review, then scaled detection with *generalized* tooling to detect data flows

TechCrunch:



Facebook admits it stored 'hundreds of millions' of account passwords in plaintext

https://techcrunch.com/2019/03/21/facebook-plaintext-passwords/

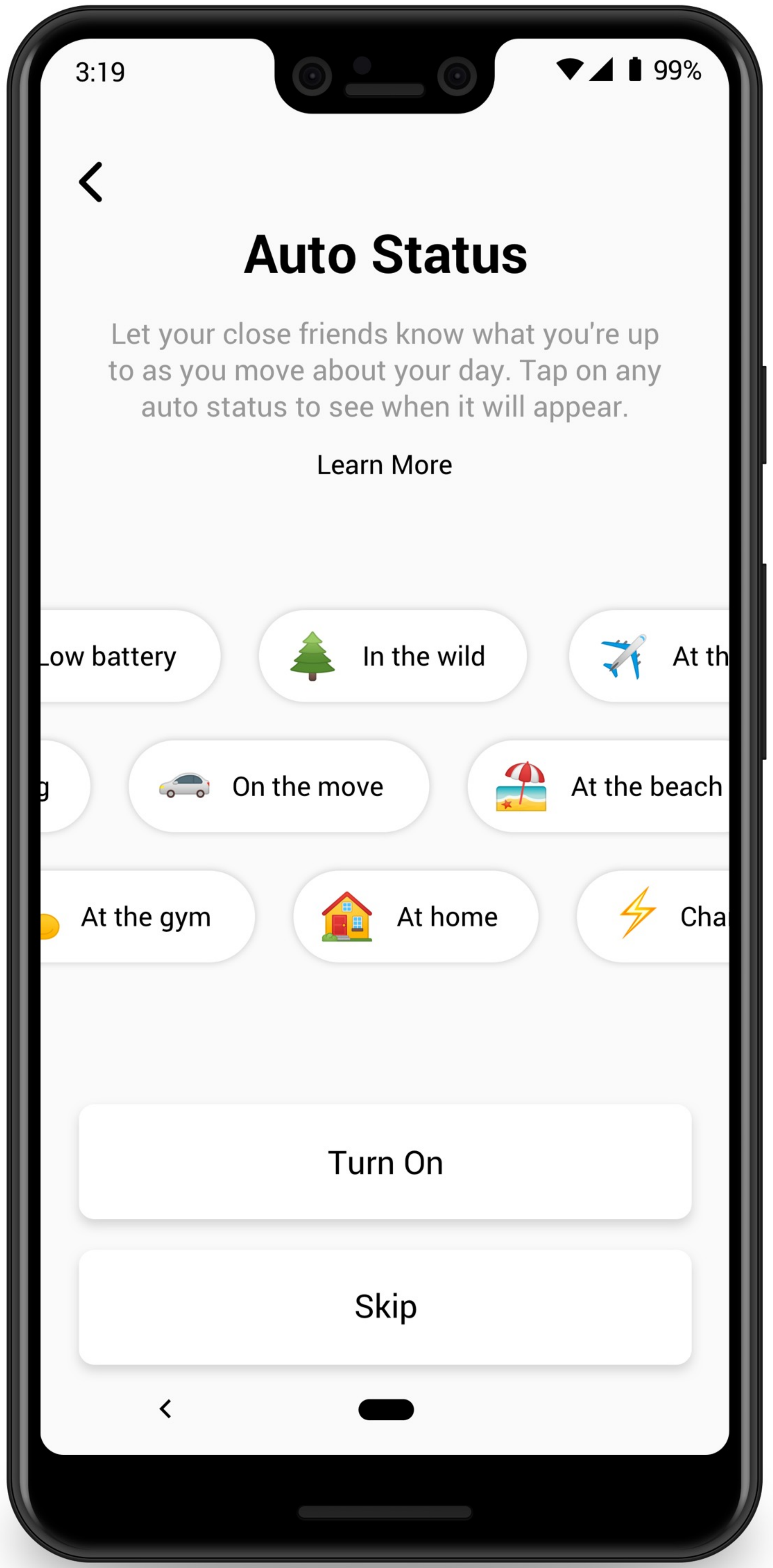
THESIS

We *can* and *should* apply security solutions to new problems outside the traditional space of security

Case Studies

Case 1: Instagram Threads Location Data

We want to use user locations to calculate status, but never store them


3:19
99%
Auto Status
Let your close friends know what you're up to as you move about your day. Tap on any auto status to see when it will appear.
Learn More
Low battery
In the wild
At th
On the move
At the beach
At the gym
At home
Cha
Turn On
Skip

GENERALIZED PROBLEM

Track data flows and make sure they don’t go where they shouldn’t

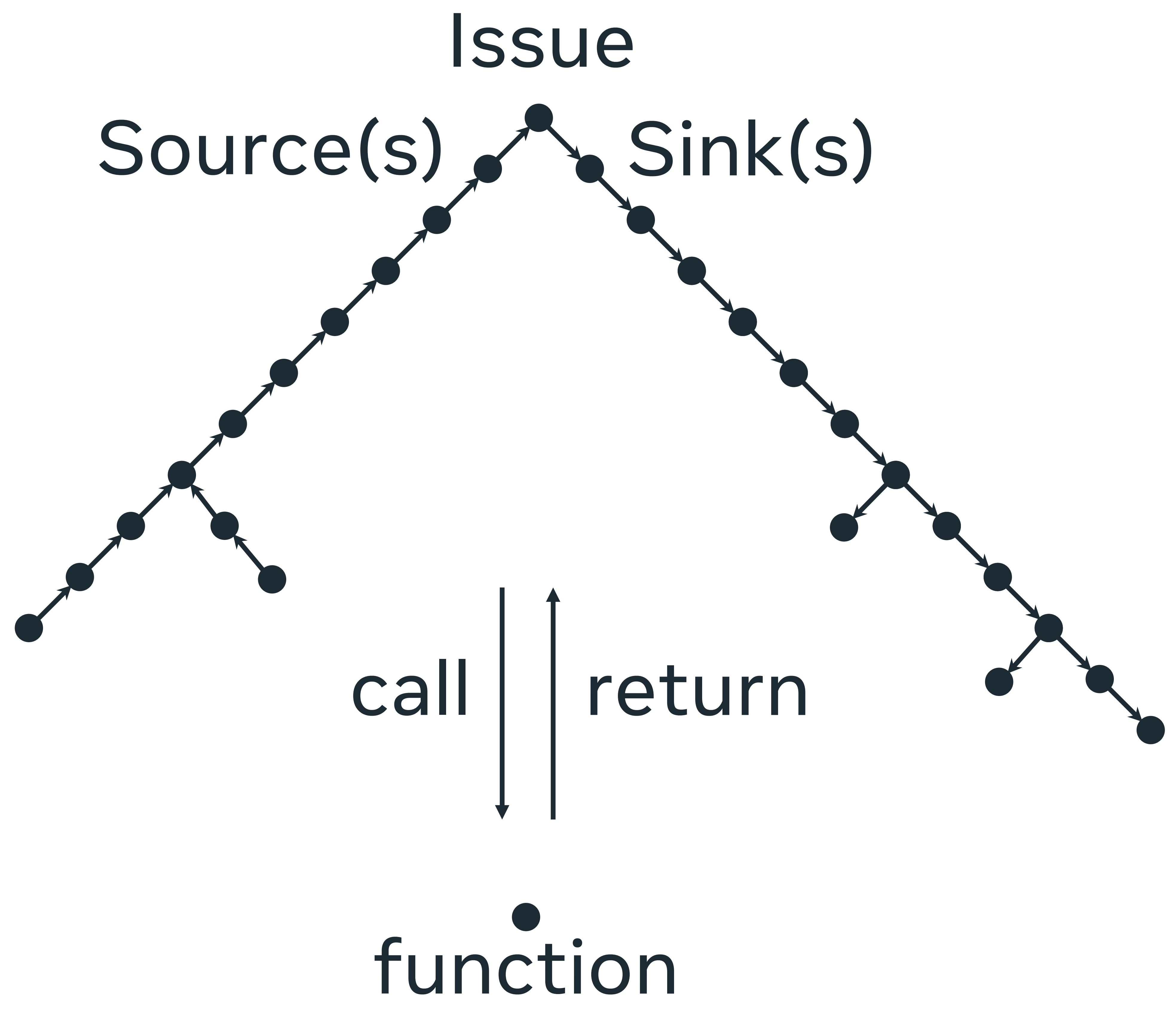
Static Taint Flow Analysis

Tainted Data = Data that originated from, or is influenced by, a source of data that we want to track

(Taint) Source = Where we define tainted data to originate

(Taint) Sink = Where want to detect tainted data ending up

Static Taint Flow Analysis = Tracking flows of *tainted* data from *source* to *sink*



Pretend SQL Injection Flow

```
# views/user.py
async def get_pictures(request: HttpRequest) -> HttpResponse:
  user_id = request.GET['user_id’]
  pictures = load_pictures user_id
  ...

# model/media.py
async def load_pictures user_id: str :
  query = f"SELECT * FROM pictures WHERE user_id = {user_id}'
  connection = create_sql_connection()
  result = await connection.execute(query)
  ...
```

Pretend Threads Flow

```
# views/threads.py
async def get_status(location: Coordinate) -> HttpResponse:
  """ Return a status for a given location """
  status = infer_status(location.lat, location.lng)
  ...

# model/status.py
async def infer_status (latitude: float, longitude: float):
  """ Infer a status for a given location """
  LOG.debug(f"Infering status for location: {latitude}, {longitude}"
  ...
```

Tools

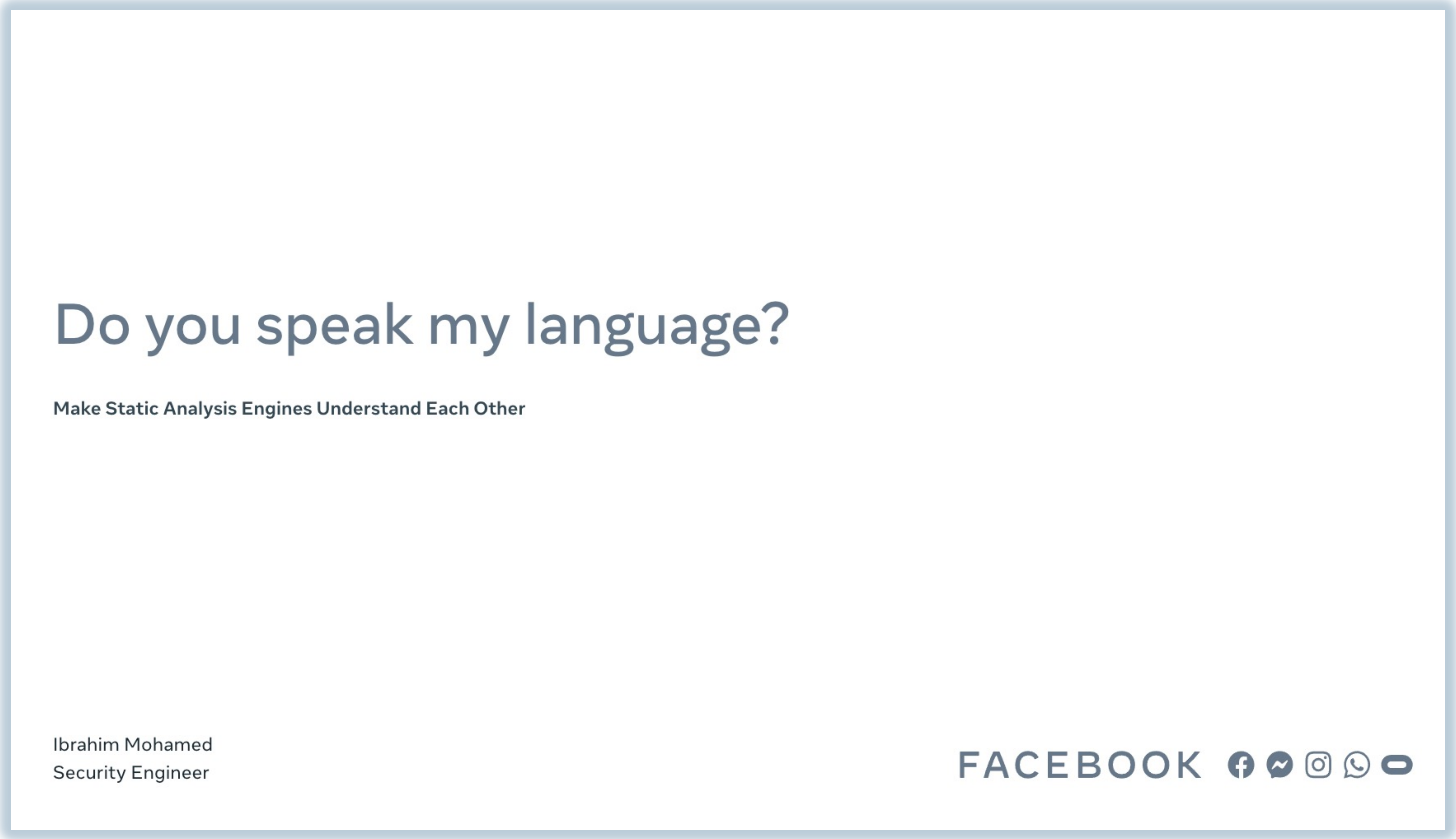
Zoncolan

Pysa

Mariana Trench

Extending Protections

Cross Repo Taint Exchange lets us stitch together analyzers



Do you speak my language?

Make Static Analysis Engines Understand Each Other

Ibrahim Mohamed
Security Engineer

FACEBOOK

Additional Use Cases

$$\begin{tikzcd}
\text{Passwords} \arrow[d] & \text{Private data} \arrow[d] & \text{Experimentation Framework} \arrow[d] \\
\text{Loggers} & \text{Returned to users} & \text{Conditional Statements}
\end{tikzcd}$$

Case 2:
Data Abuse

We want to know when bad actors are collecting data they shouldn't

CNET:

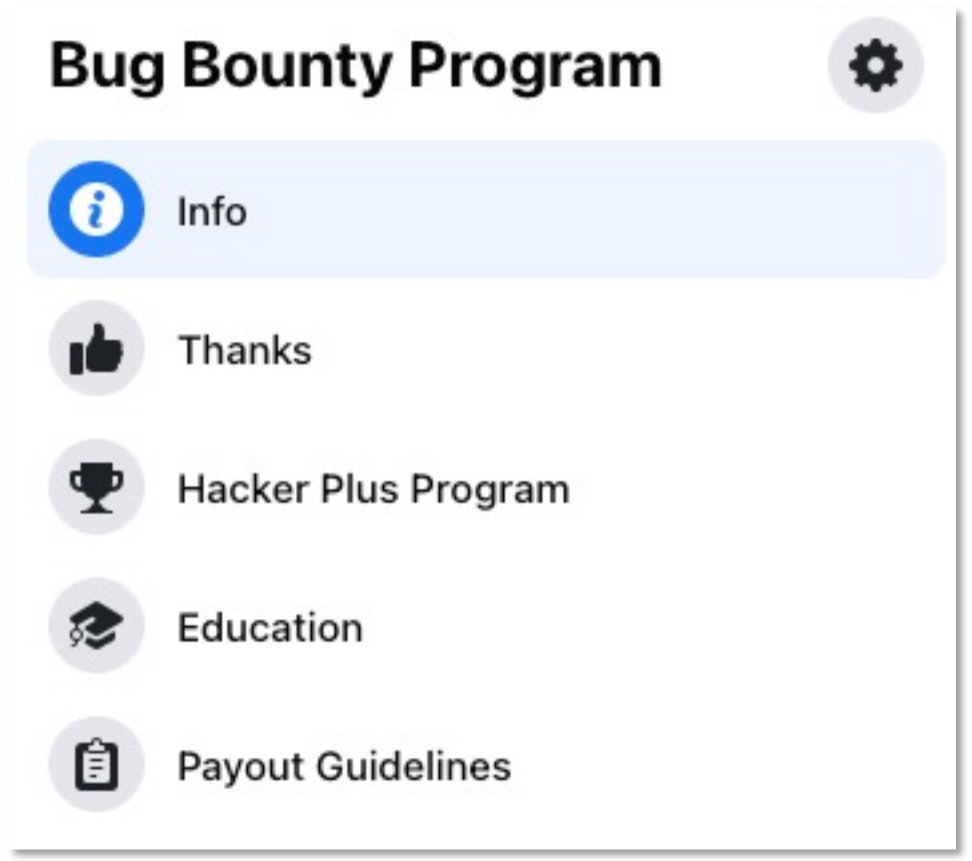
Facebook says data from 530M users was obtained by scraping, not hack

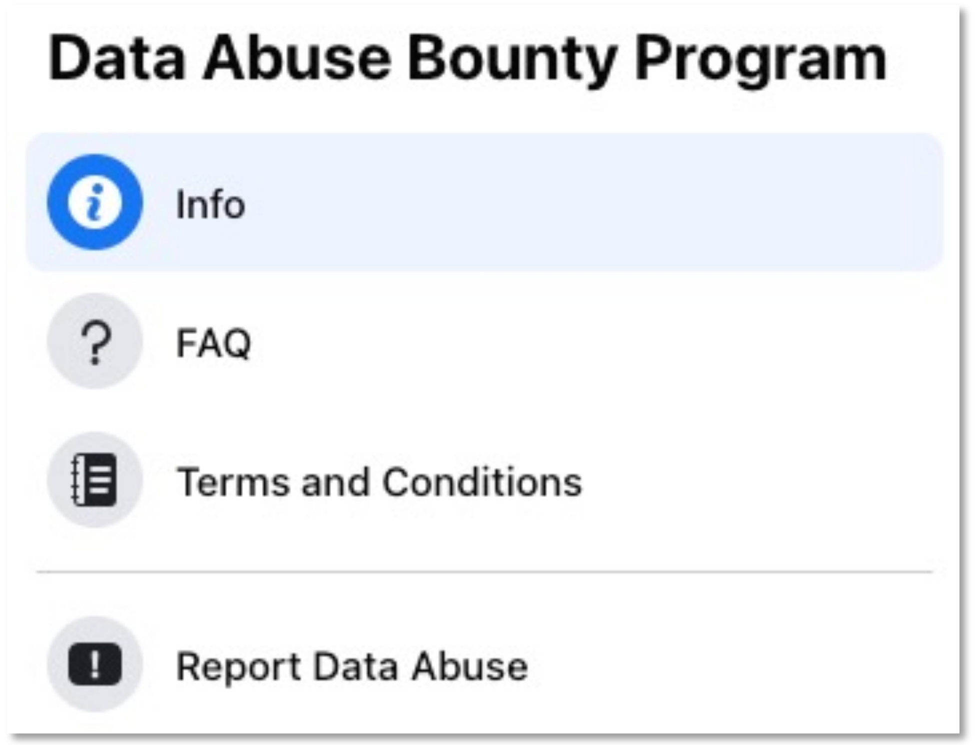
https://www.cnet.com/tech/services-and-software/facebook-says-data-leak-is-from-old-vulnerability-that-no-longer-exists/

GENERALIZED PROBLEM

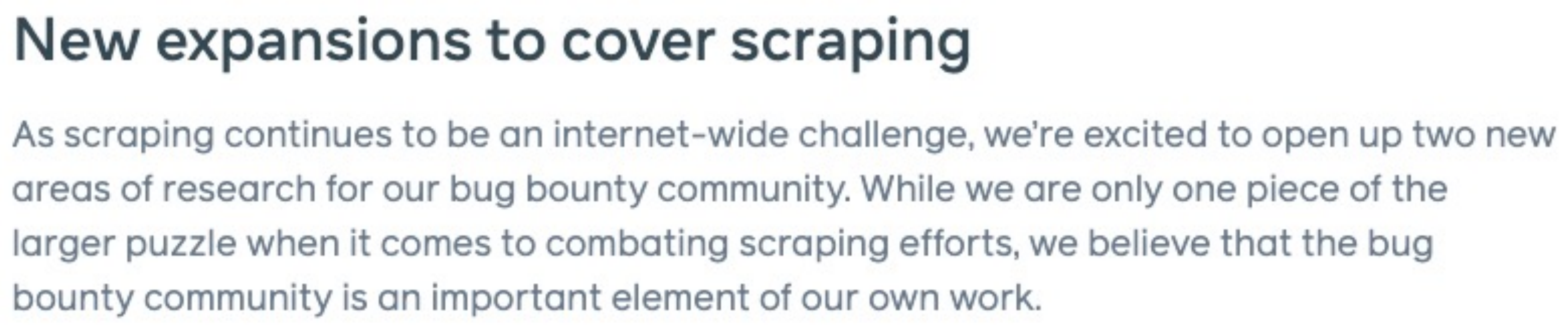
Incentivize people who spot an issue to warn us, so we can fix it before it’s exploited

Applications





+



New expansions to cover scraping

As scraping continues to be an internet-wide challenge, we're excited to open up two new areas of research for our bug bounty community. While we are only one piece of the larger puzzle when it comes to combating scraping efforts, we believe that the bug bounty community is an important element of our own work.

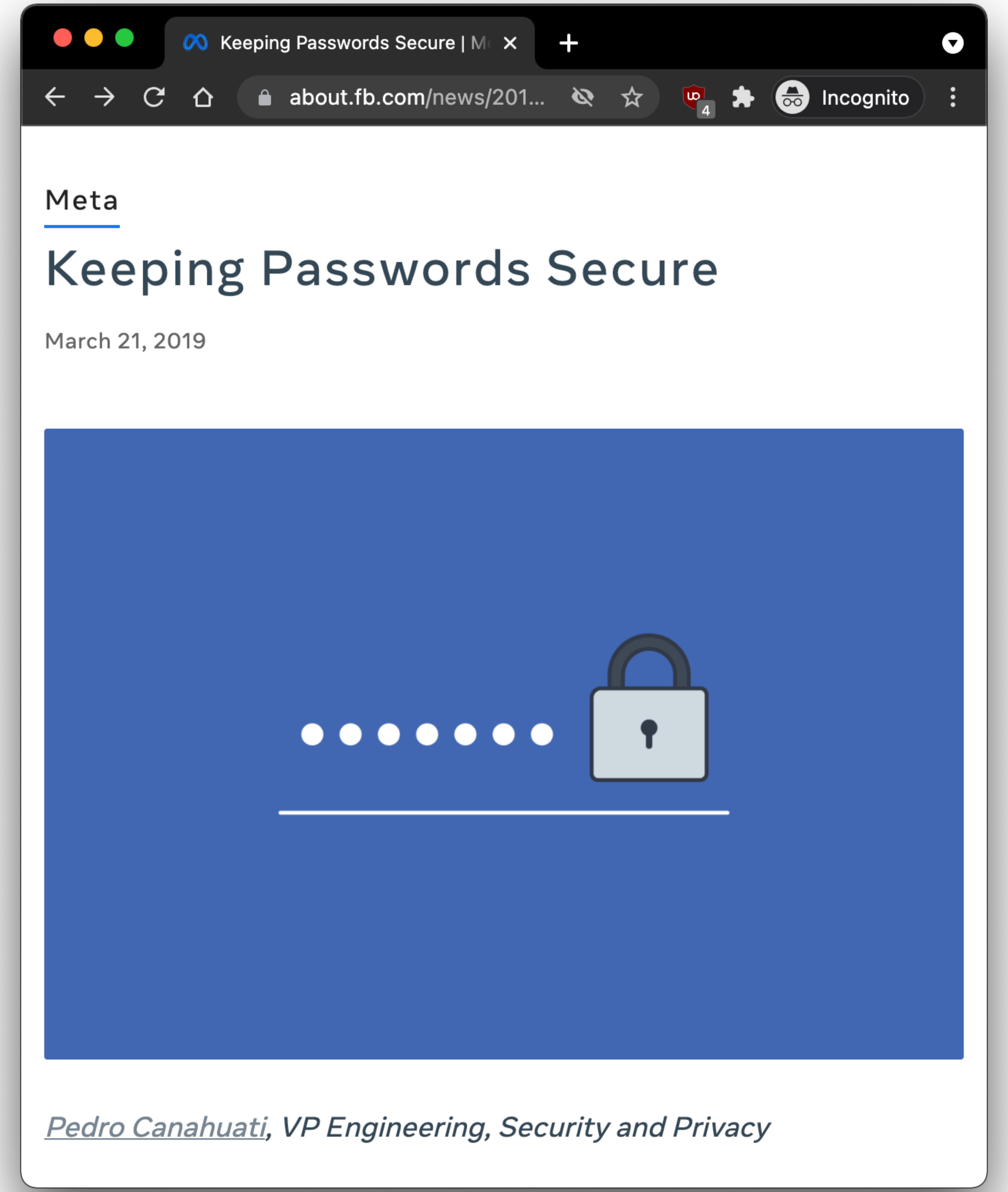
Security

Data Abuse / Scraping

CASE STUDIES

Case 3: Password Logging

We want to make it impossible for systems that don't need access to passwords to log them in plain text



Meta

Keeping Passwords Secure

March 21, 2019

Pedro Canahuati, VP Engineering, Security and Privacy

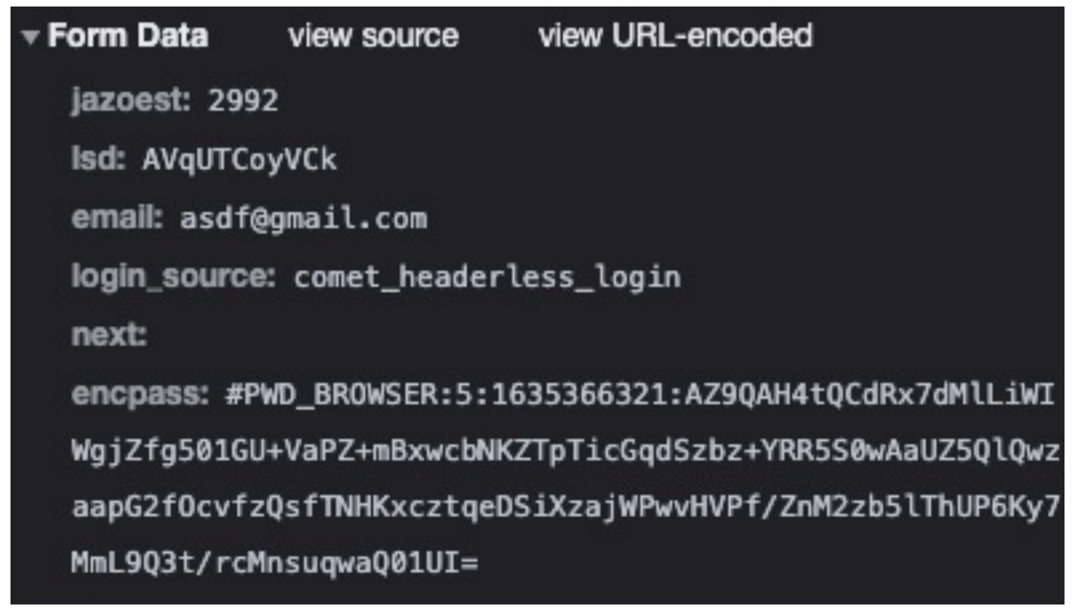
GENERALIZED PROBLEM

Obfuscate information in transit

Applications

HTTPS

Security



```
jazoest: 2992
lsd: AVqUTCoyVCk
email: asdf@gmail.com
login_source: comet_headerless_login
next:
encpass: #PWD_BROWSER:5:1635366321:AZ9QAH4tQCdRx7dMlLiWI
WgjZfg501GU+VaPZ+mBxwcbNKZTpTicGqdSzbz+YRR5S0wAaUZ5QlQwz
aapG2fOcvfzQsfTNHKxcztqeDSiXzajWPwvHVPf/ZnM2zb5lThUP6Ky7
MmL9Q3t/rcMnsuqwaQ01UI=
```

Password Logging

Case 4:
Unsafe Data Access

We want to make developers aware of the risks of APIs which can bypass the privacy checks built into database access, and discourage its use

GENERALIZED PROBLEM

Ensure developers understand the risks of an API, and use it sparingly

Applications

```
return <div dangerouslySetInnerHTML={{__html: value}} />;
```

Security

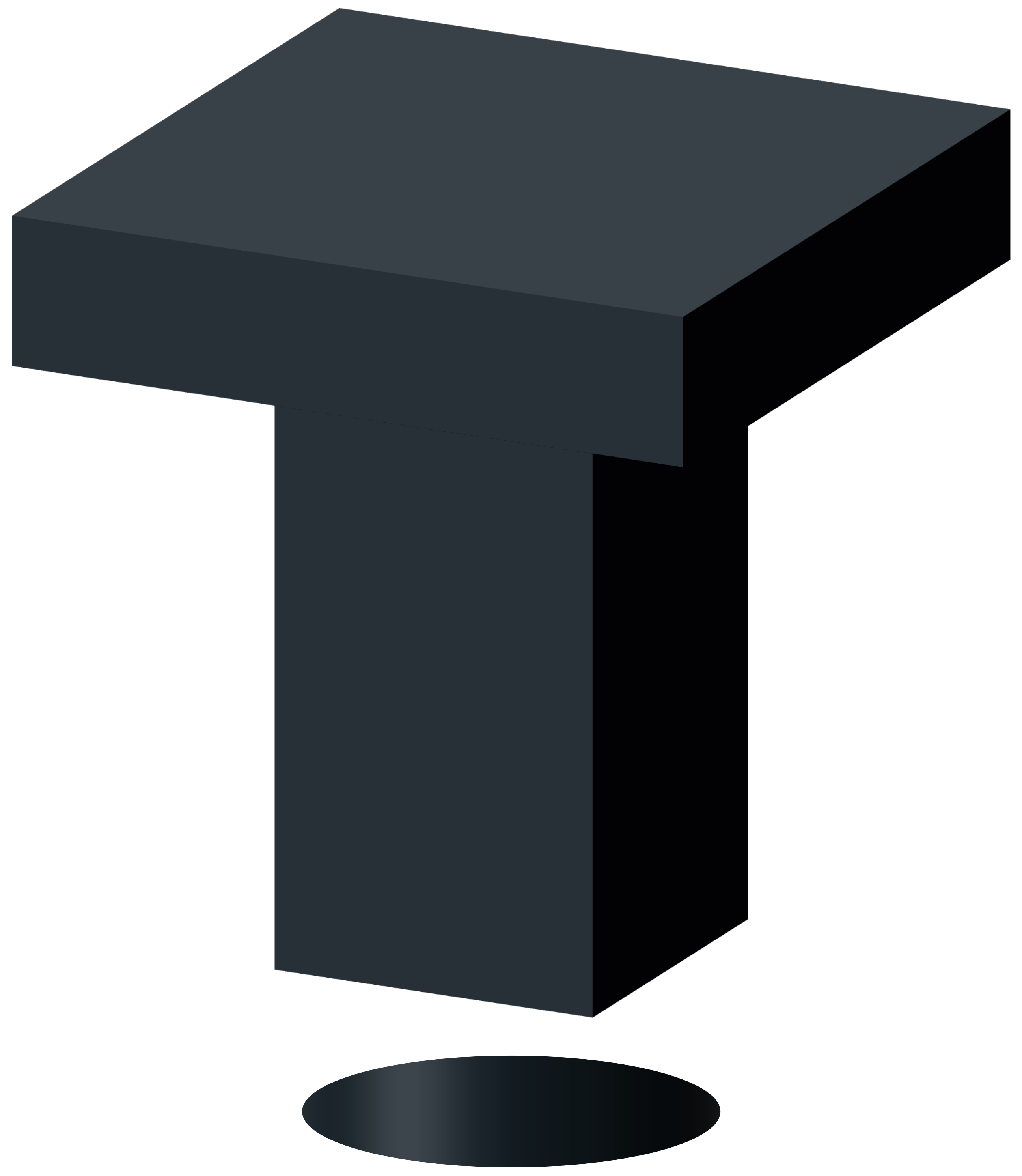
```
function omnicient_THIS_WILL_BYPASS_PRIVACY_CHECKS(
    ...
)
```

Privacy Checks

Limitations

Solution Design

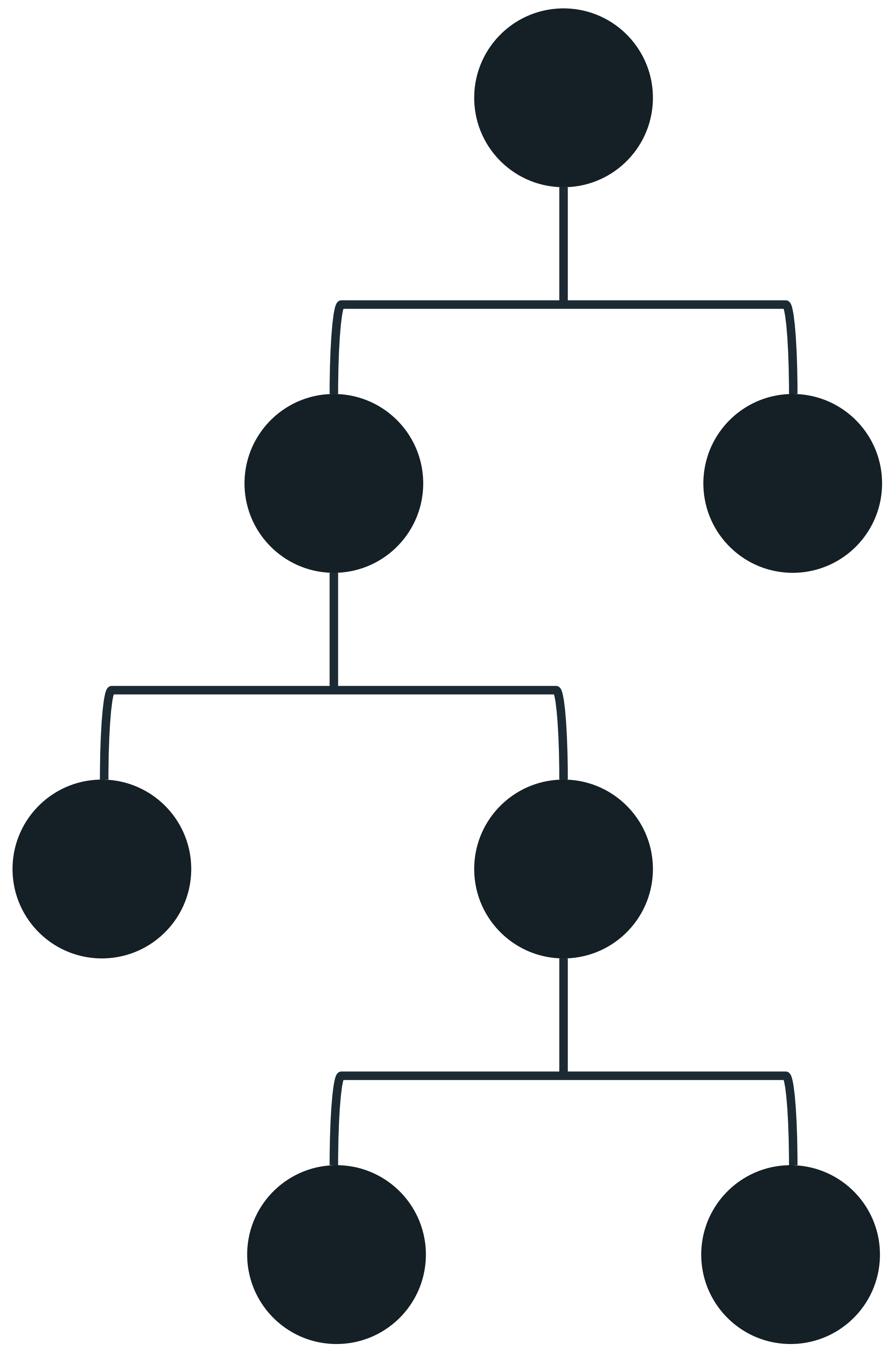
Not all solutions translate; avoid square pegs in round holes



LIMITATIONS

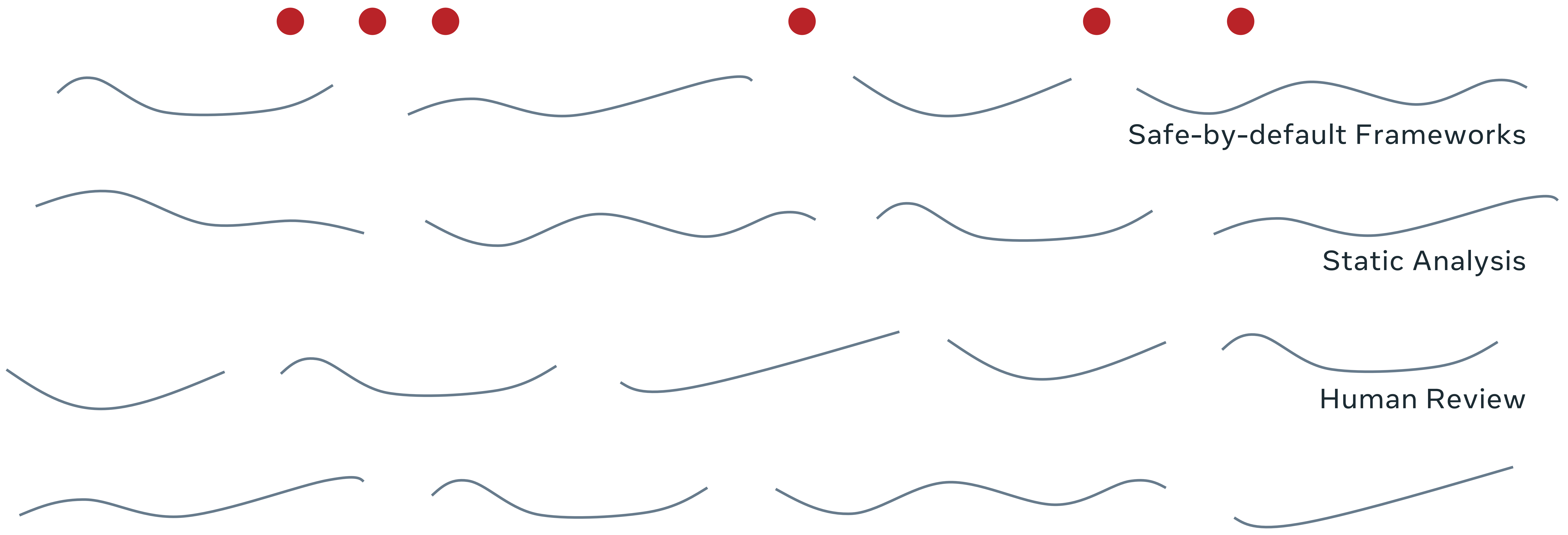
Organization Design

Solutions often have implicit dependencies on the structure of a security organization



Gaps in Coverage

Translate defense in depth, just like you translate your solutions



Conclusion

THESIS

We *can* and *should* apply security solutions to new problems outside the traditional space of security

Takeaways

Great security solutions solve generalized problems which also exist outside security

These solutions can help in domains such as performance, compliance, privacy, and data abuse

Recognize when reusing solutions *wont* work

Thanks

- Ted Reed
- David Molnar
- Kyle McEachern
- Ryan Nakamoto
- Parmeshwar Arewar
- Edward Qiu
- Otto Ebeling
- Swathi Joshi
- Pritam Dash

Interested in solving difficult problems like these?

Chat with me after the talk or shoot me an email:

gbleaney@fb.com

Questions

Meta