REST identity and access management

About Johan

- Software architect
- Founder of secappdev.org
- Consultancy
- MVPs, PoCs, pilots
- Lecturer at EhB

https://www.johanpeeters.com
@YoPeeters
yo@johanpeeters.com

About Michael

- Independent cyber security consultant
- Security architect @ Colruyt Group
- Training / Security enthusiast

https://www.portasecura.com

https://www.linkedin.com/in/michaelboeynaems

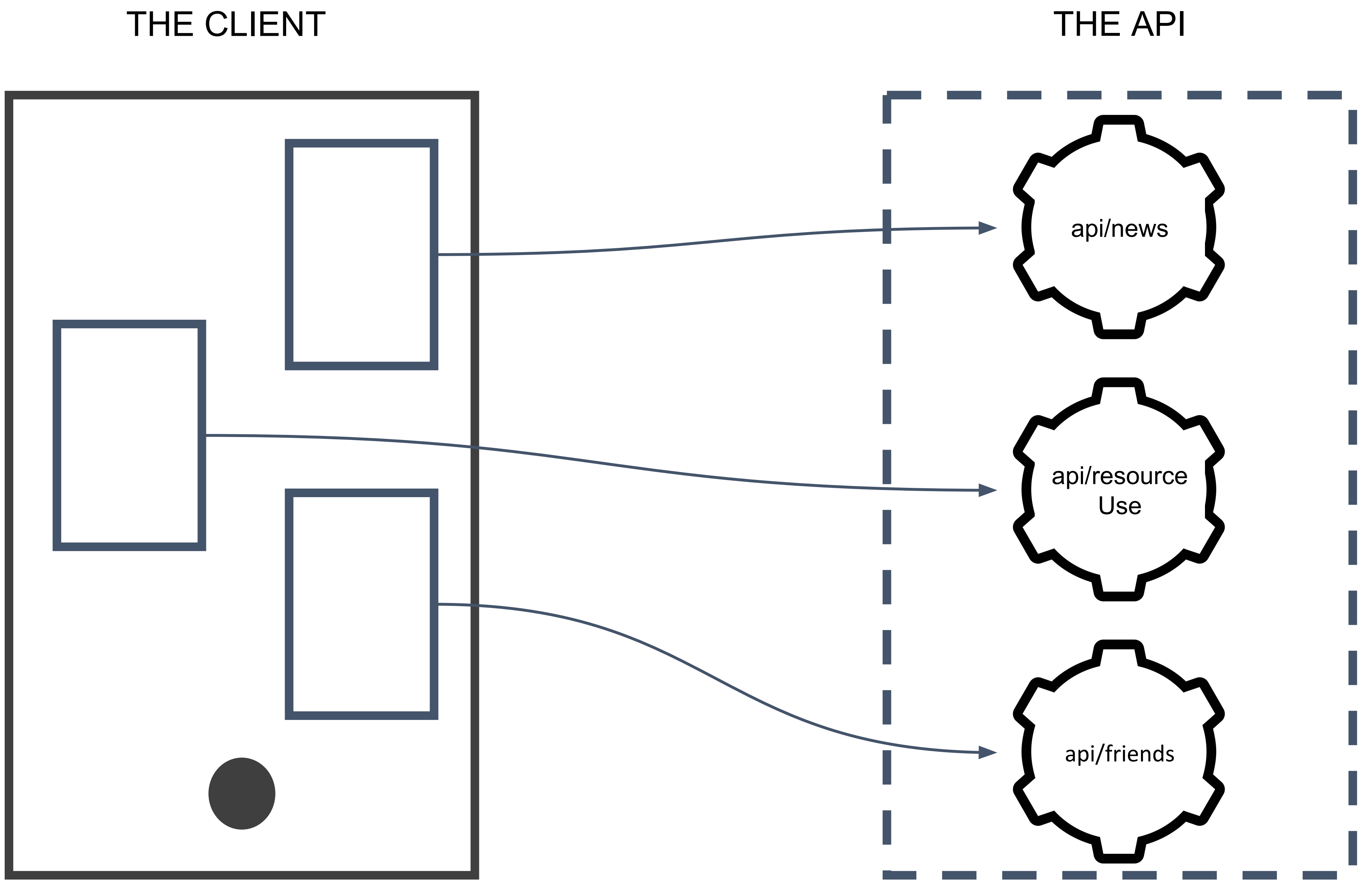
michael.boeynaems@portasecura.com

Case study

You are the developer of a social network for hackers set up by an ISP.

The client retrieves information from the ISP’s back-end APIs.

THE CLIENT
THE API
api


THE CLIENT
THE API
api/news
api/resource
Use
api/friends

Demo: API with Azure Functions

- Create function app
- Add function to the app which returns a news item

Challenge: approved clients only

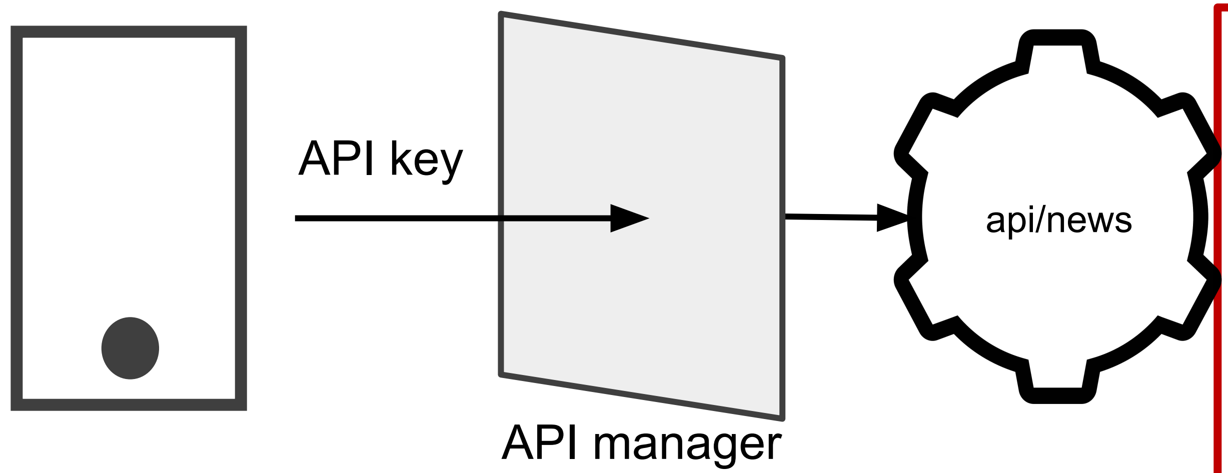
Solution: shared API key

- Not natively possible in the function app, so we use an API manager
- The API manager validates the API key

Challenge: approved clients only

Solution: shared API key

- Not natively possible in the function app, so we use an API manager
- The API manager validates the API key



It works! Yippee! But...

- What if multiple apps consume the API?
- How to securely distribute the API key?
- Can news be personalized?
- How to revoke/limit access in time?

Challenge: finer grained access control

Solution: access tokens!

- The API manager validates the incoming token

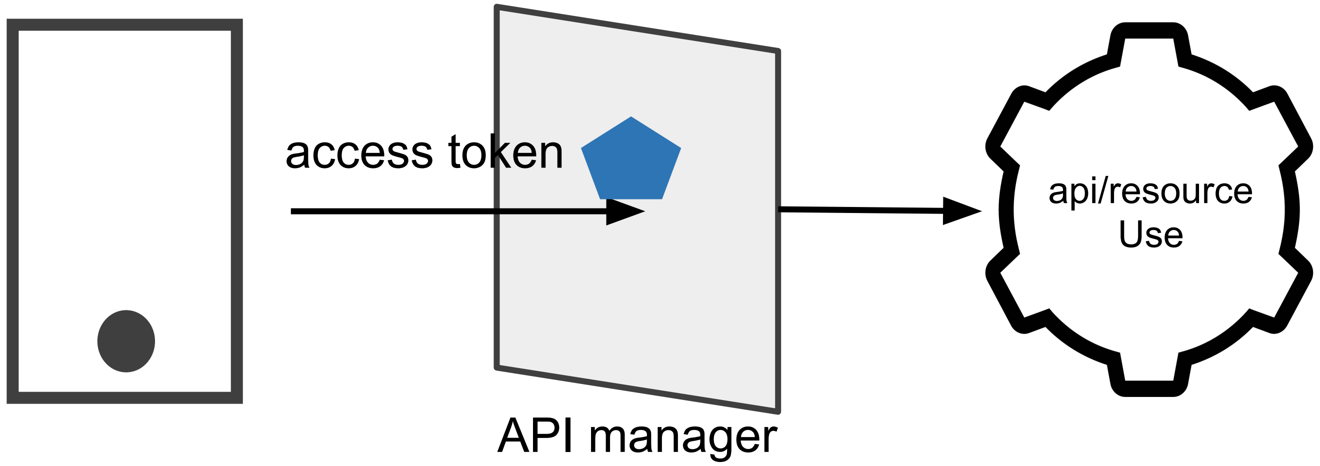
The access token

- Typically a JWT
- Signed by
  - the STS
  - the issuer
  - the authorization server
- Contains some useful claims (exp, sub, iss, aud)

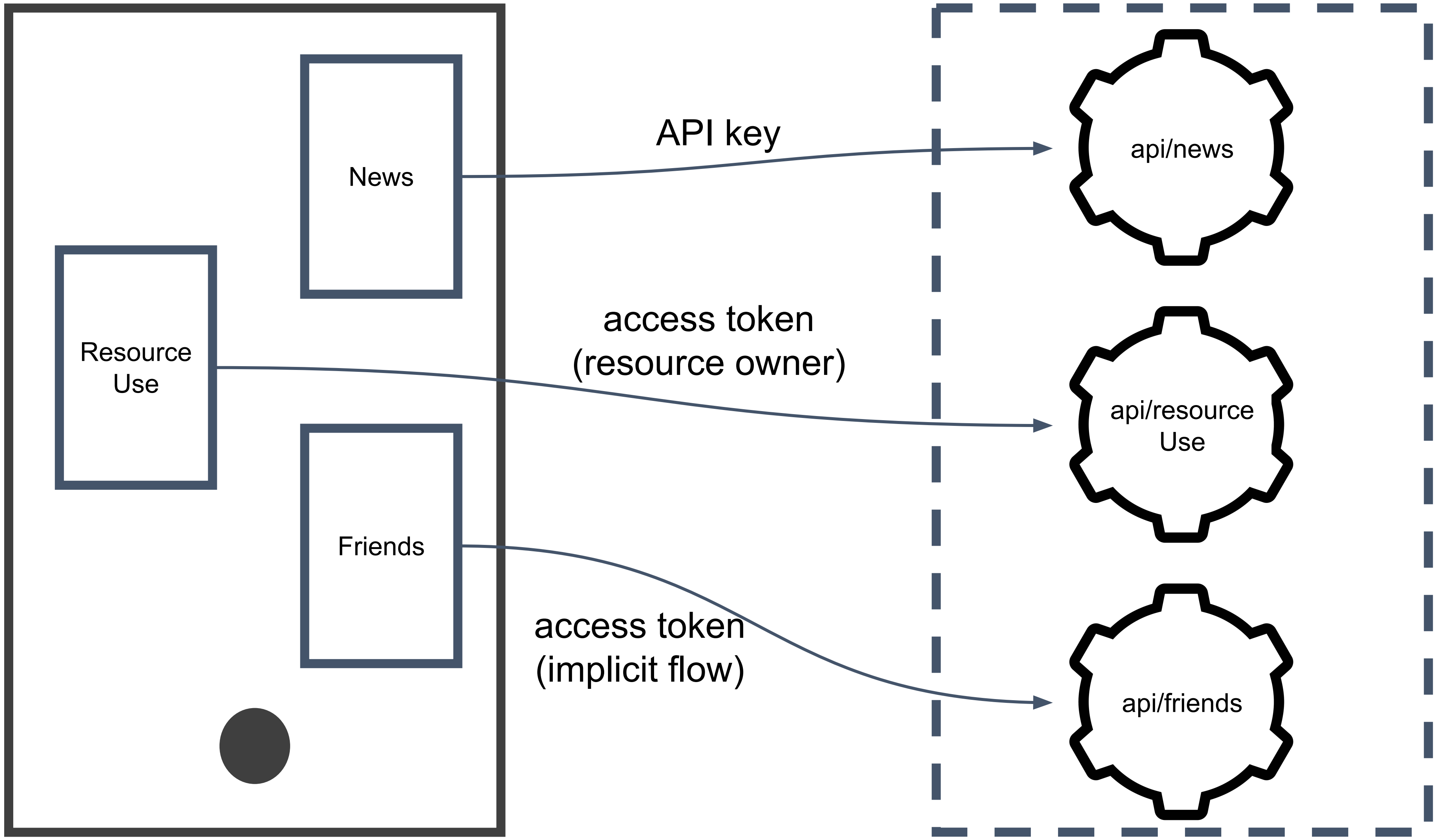
Challenge: finer grained access control

Solution: access tokens!

- The API manager validates the incoming token



It works! Yippee! But...
where did the token come from?

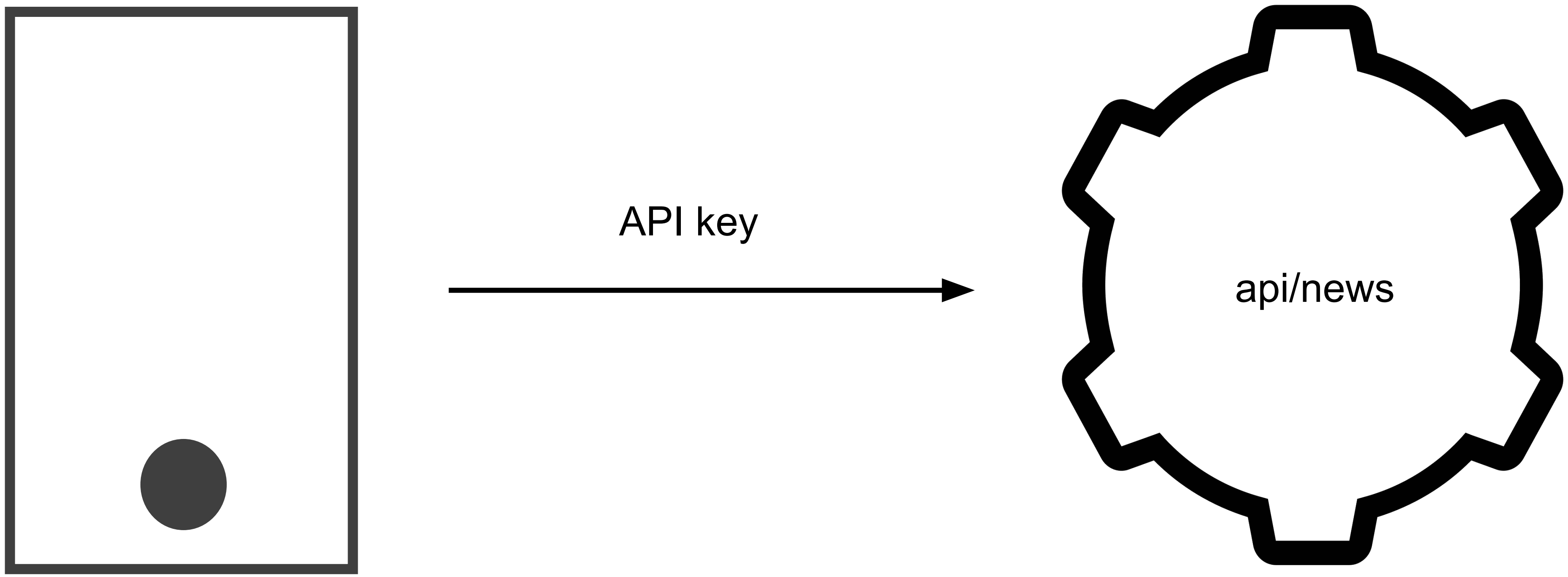

News
Resource Use
Friends
API key
access token (resource owner)
access token (implicit flow)
api/news
api/resource Use
api/friends

Demo: React SPA

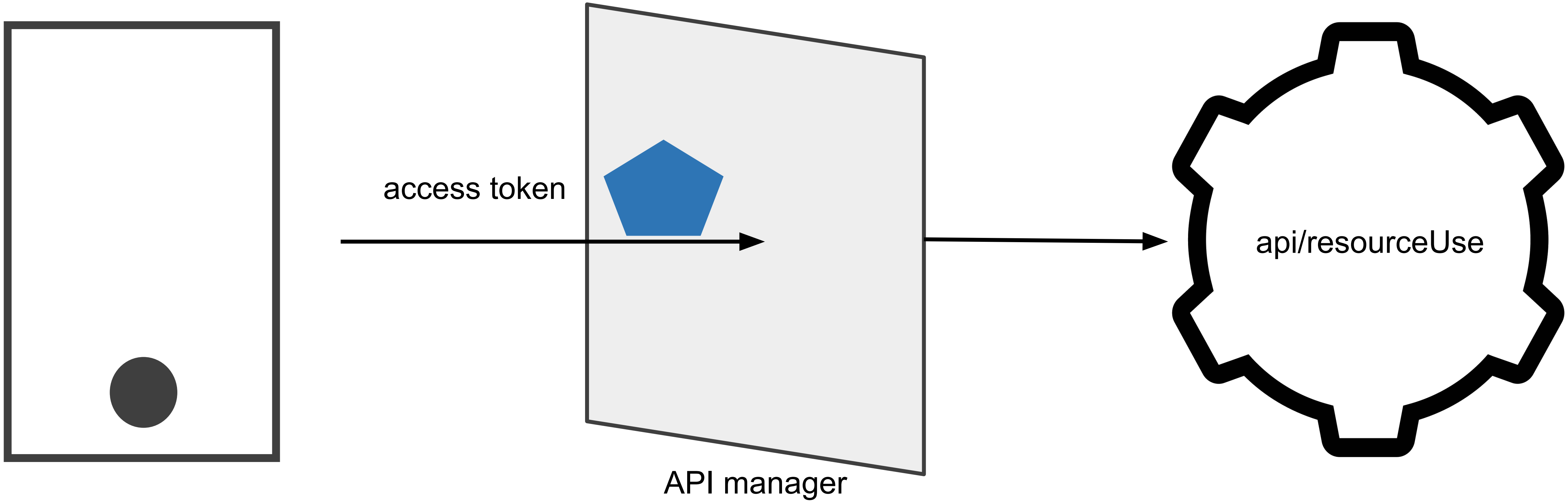
```
npx create-react-app react-rest-client
npm i auth0-js axios react-bootstrap
```

and add components

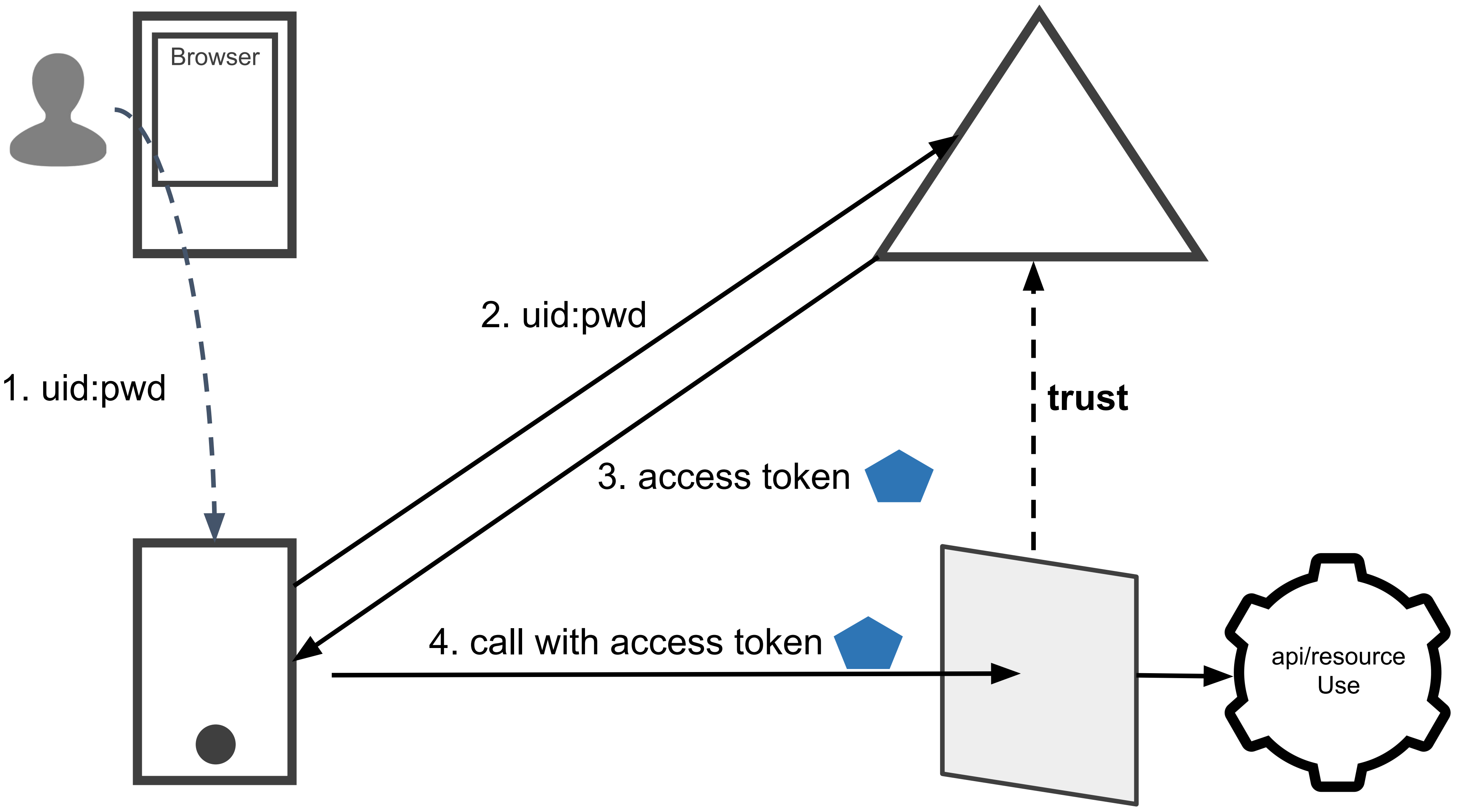
React Component 1: News



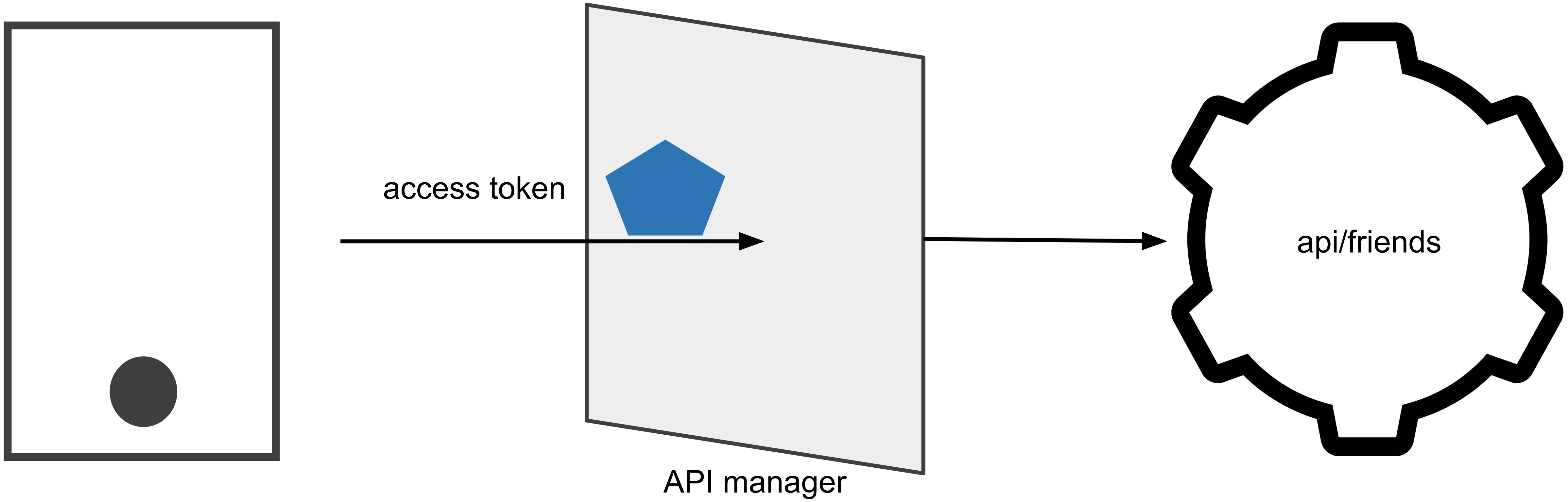
React Component 2: ResourceUse



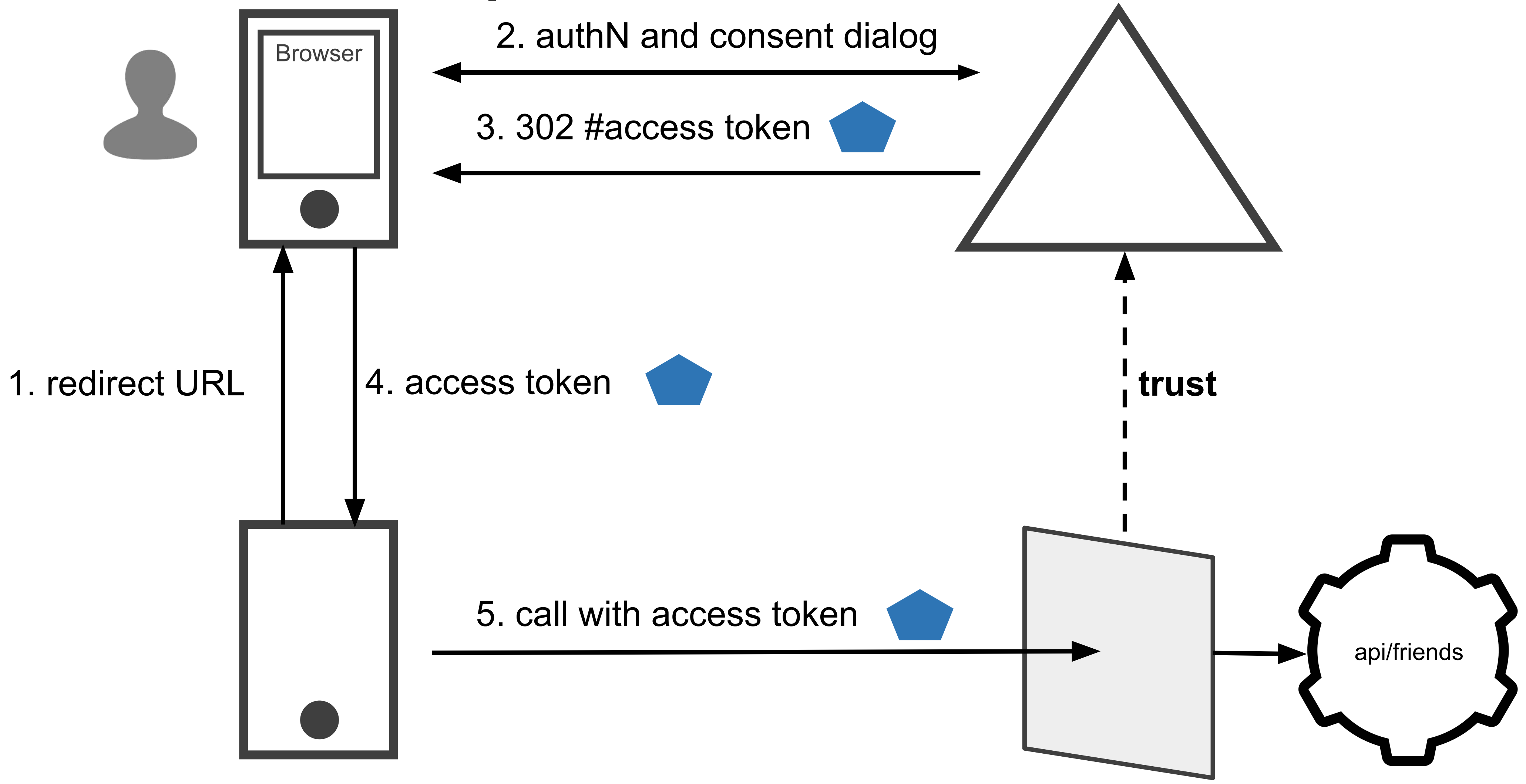
Obtaining an access token with a password



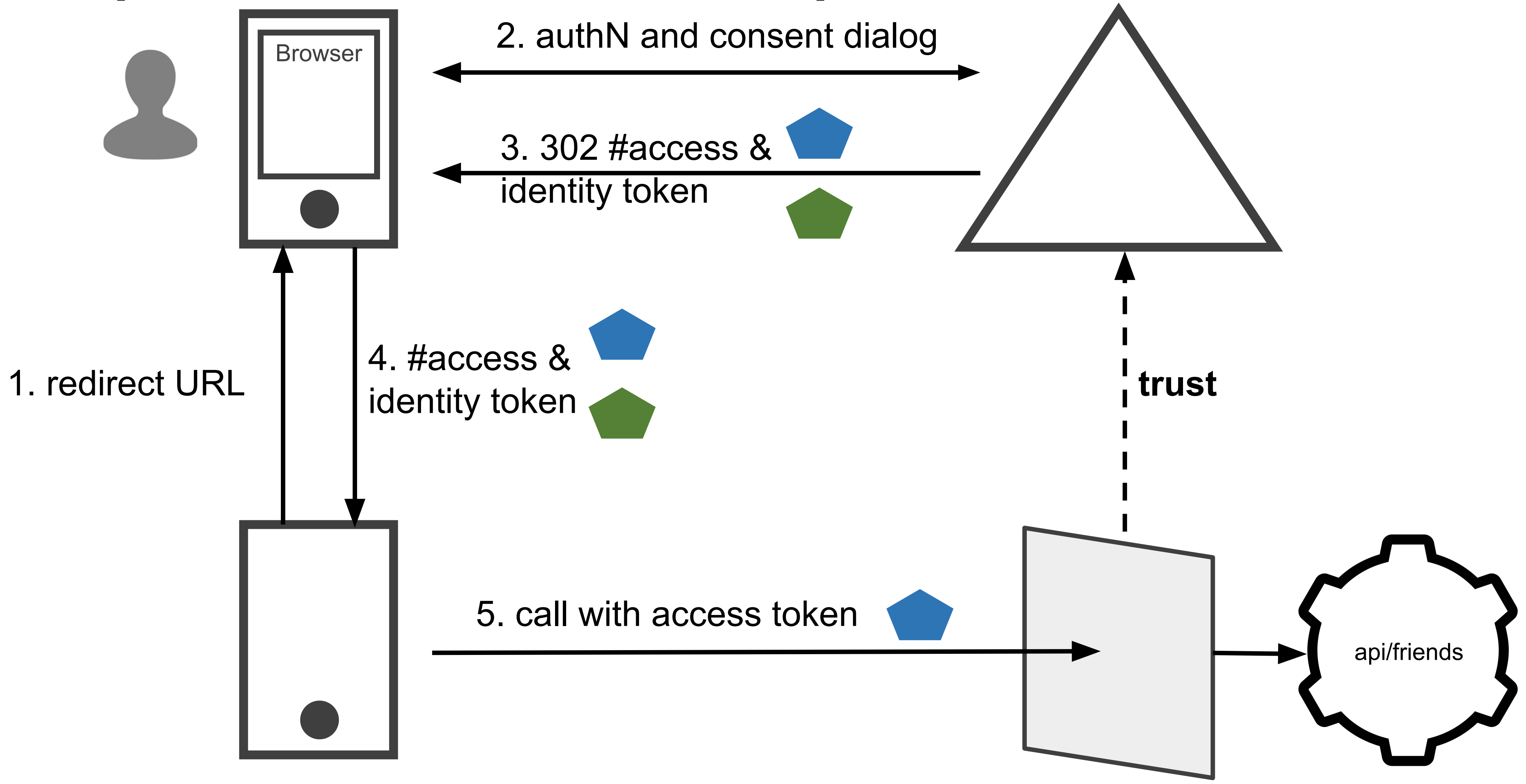
React Component 3: Friends



OAuth 2 Implicit Grant

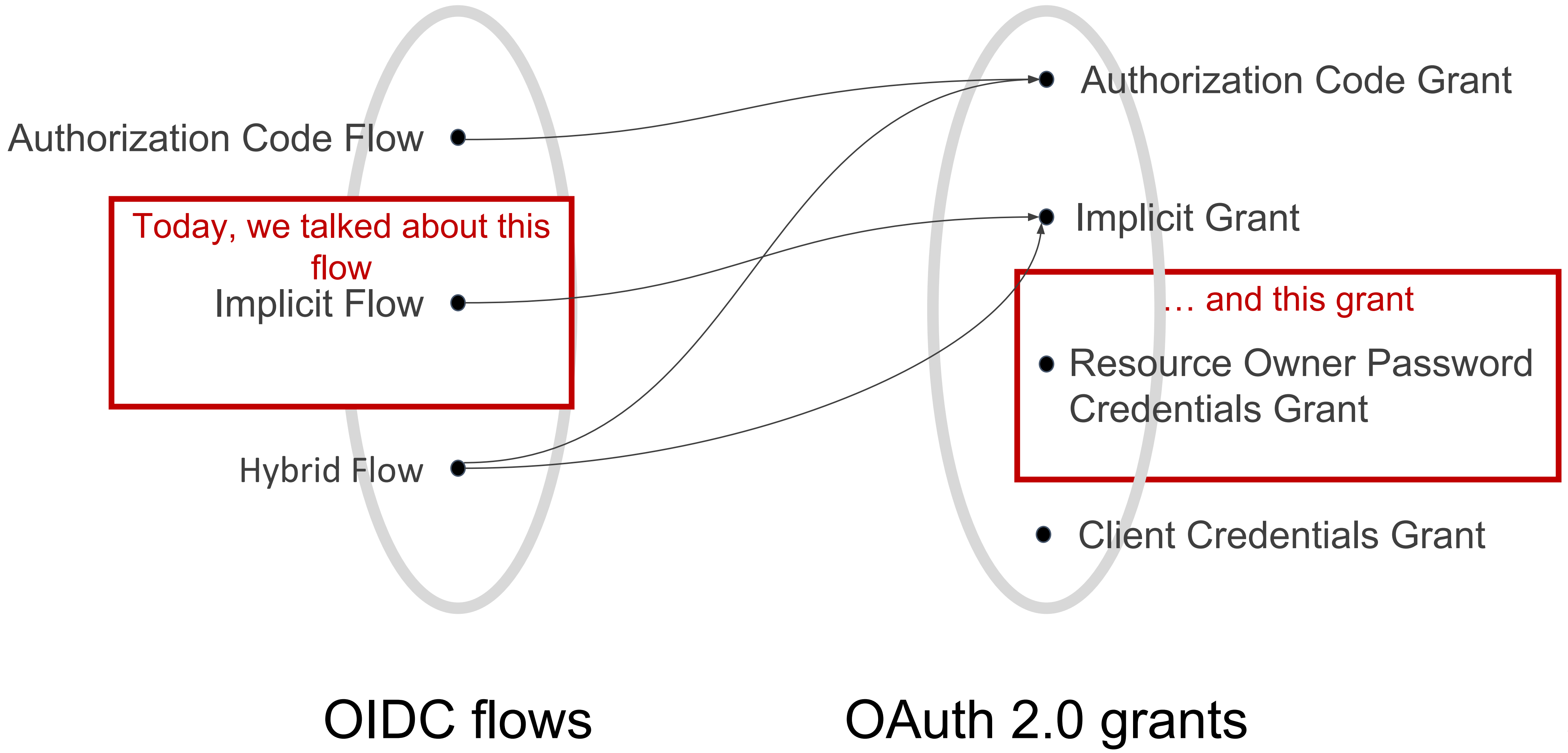


OpenID Connect Implicit Flow



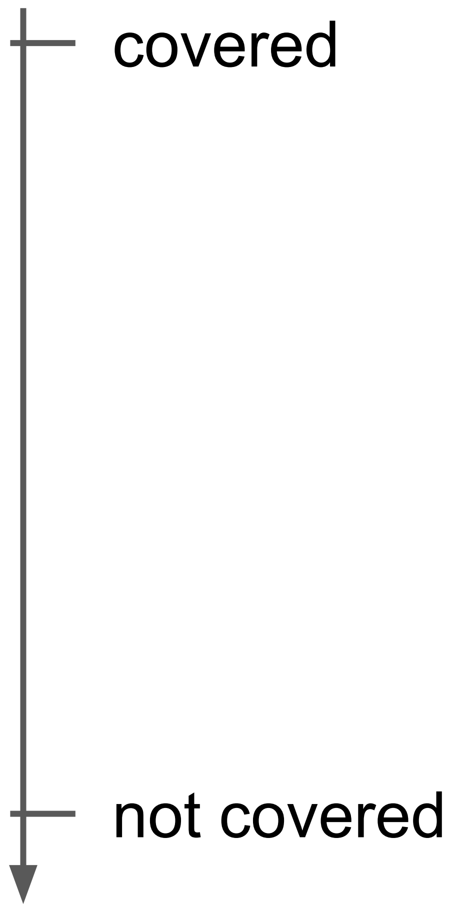
Browser
2. authN and consent dialog
3. 302 #access & identity token
1. redirect URL
4. #access & identity token
trust
5. call with access token
api/friends

More ways to obtain tokens



Architectural considerations

- Create choke points AKA principle of complete mediation
- Separation of concerns
  - Application logic and authentication
  - Application logic and authorization AKA policy-based access control
- Stateless services
- Minimize resource reservation
- Trade off security against performance
- ...



Further topics

- Identity Management
- Access rules
- Access control models
- Authentication
- Fan out
- Call chaining

Resources

- https://github.com/JohanPeeters/react-rest-client
- https://github.com/Mich-b/ModernACREST.git
- https://github.com/JohanPeeters/REST-IAM-demo
- https://jwt.io
- https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
- https://tools.ietf.org/html/rfc6749
- https://openid.net/specs/openid-connect-core-1_0.html