• Defining security properties for OAuth-like protocols and client-side flows
  • Separate doc for long-term vision (BCP or separate draft?)

• Evaluating mitigations and protocol extensions

• Analyzing mitigations
Security properties for OAuth 2.0

• **Proof-of-possession**
  • Also a form of authentication, addressed with token bindings

• **Containment**
  • Eliminate infoleaks/extraction through Referrer, Fragment, server logs

• **Authentication**
  • Allow endpoints to identify sender and receiver (caller URL/origin)
Evaluating mitigations and protocol extensions

Implementation level:
• TLS vs. HTTP
• OS vs. browser vs. application
• Provider vs. client

Amount of protection:
• Which security properties it addresses?
• Does this cover the missing property(ies) fully?
• Which mitigations it obsoletes?

Implementation costs:
• Complexity and cost of deployment
  • People won’t implement what they don’t understand or what’s hard
• Deprecation costs
  • Every breaking change should have a very clear business objective
Evaluating mitigations and protocol extensions: Mix-Up: iss + client_id returned in response

Implementation level:
• Application-level
• Provider + client (requires protocol change)

Amount of protection:
• Property: Authentication
• Not covers authentication fully (URL params are spoofable from web attacker), just the Mix-Up

Implementation costs:
• Complexity: medium (new response_type + params check on client)
• Deprecation costs: no (backward compatible)
POST binding + Origin check

POST https://provider/oauth

Origin: client.com

... client_id={client_id}&redirect_uri={redirect_uri}&state={state}

↓

is client.com permitted for {client_id}?

↓

HTTP/1.1 200 OK

...

<form action="{redirect_uri}" method="POST">...
POST binding + Origin check to mitigate IdP MixUp

POST {redirect_uri}

Origin: provider.com
...

code={code}&state={state}
↓
  is provider.com the origin we expect to handle for this {state} or current session?
↓
  code → token exchange
  login
  ...
...
Evaluating mitigations and protocol extensions: POST binding with Origin check

Implementation level:
• Application-level
• Provider + client (requires protocol change)

Amount of protection:
• Property: Authentication + Containment
• Covers authentication (almost) fully (Origin header is not spoofable from web attacker)
  • Almost because Origin has domain, not full endpoint URL
• Covers containment (almost) fully
  • Except 307 redirect leaks

Implementation costs:
• Complexity: low
• Deprecation costs: high (migrate provider + client flows)