A Comprehensive Formal Security Analysis of OAuth 2.0

Daniel Fett Ralf Küsters Guido Schmitz

Information Security and Cryptography
University of Trier
Germany

infsec.uni-trier.de

Our Contributions

OAuth 2.0:

- Developed formal model of the standard
  - * Based on most comprehensive model of the web to date (extension of S&P 2014).
- Formalized central security properties
  - * Authorization
  - * Authentication
  - * Session Integrity
- Tried to prove these properties, but found severe attacks (also on OpenID Connect)
- Proposed fixes
- Proved security for fixed standard

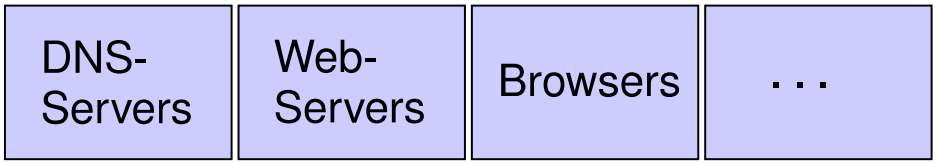
All details: TR available at infsec.uni-trier.de

Formal Analysis of Web Applications and Standards

- Many flaws and attacks have been discovered in web applications

- Web sites & systems are increasingly complex

- Interaction of different components:



- Important to **precisely specify security properties**

- Carry out **security proofs**

Sources

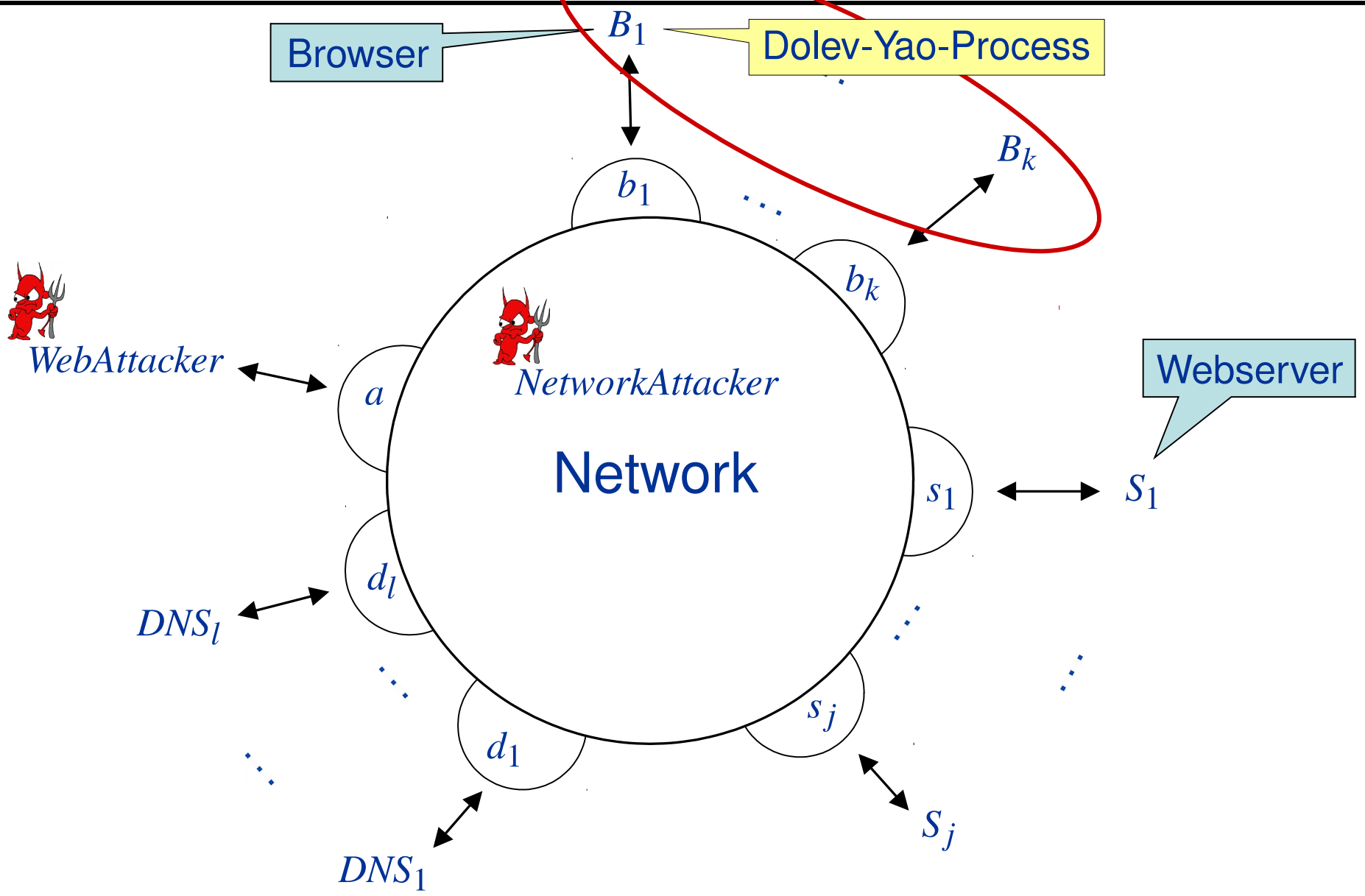
Specifications for the web are spread across many sources with mutual dependencies:

- Standards and RFCs
  - HTTP/1.1 and HTTP/2 Standards
  - W3C HTML5
  - W3C Web Storage
  - WHATWG Fetch
  - W3C Cross-Origin Resource Sharing
  - RFCs (6265, 6797, 6454, 2616, . . .)

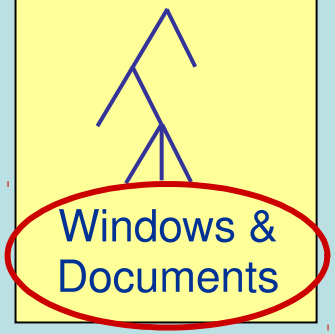
- Browser implementations
  - Google Chrome
  - Mozilla Firefox
  - . . .

➡ Model provides coherent view of core aspects of the web

Web Model


Browser
B_1
Dolev-Yao-Process
B_k
b_1
b_k
WebAttacker
NetworkAttacker
a
Network
Webserver
s_1
S_1
d_l
DNS_l
s_j
d_1
S_j
DNS_1

Web Browser State



Windows & Documents

domainA: cookie1
cookie2
domainB: cookie3
domainC: cookie4

Cookies

originA: key1: value1
key2: value2
originB: key1: value3
originC: key3: value4

LocalStorage

originA, win_1: key1:…
key2:…
originB, win_1: key1:…
originC, win_2: key3:…

SessionStorage

originA: ●●●●●
originB: ●●●●

User Secrets

domainA: k_1
domainB: k_2
domainC: k_3

Key Mapping

domainA,
domainB,
domainC

Strict Transport Security Domains

n_1: request1,
n_2: request2,
n_3: request3

Pending DNS Requests

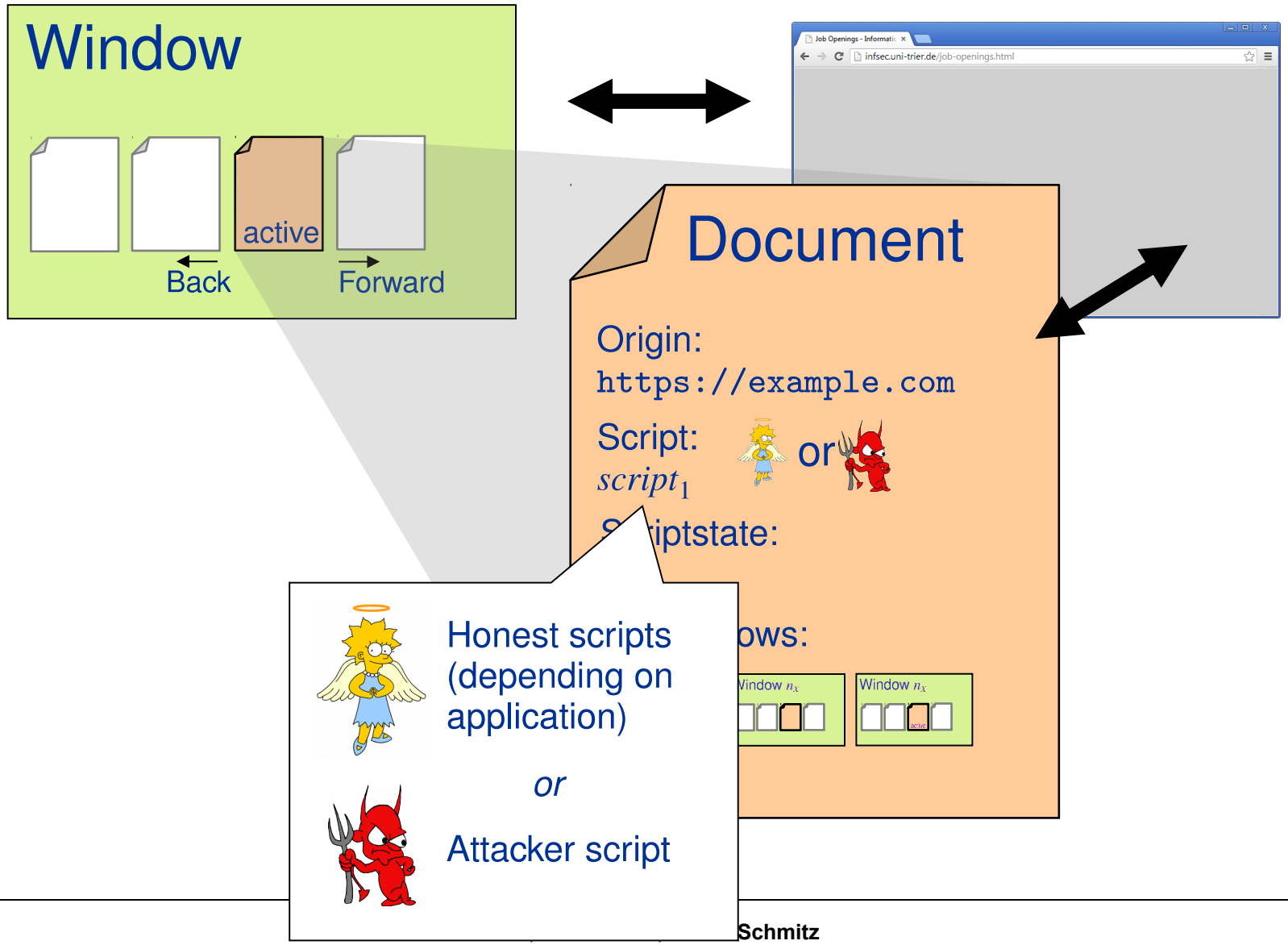
n_4: request4,
n_5: request5,
n_6: request6

Pending HTTP Requests

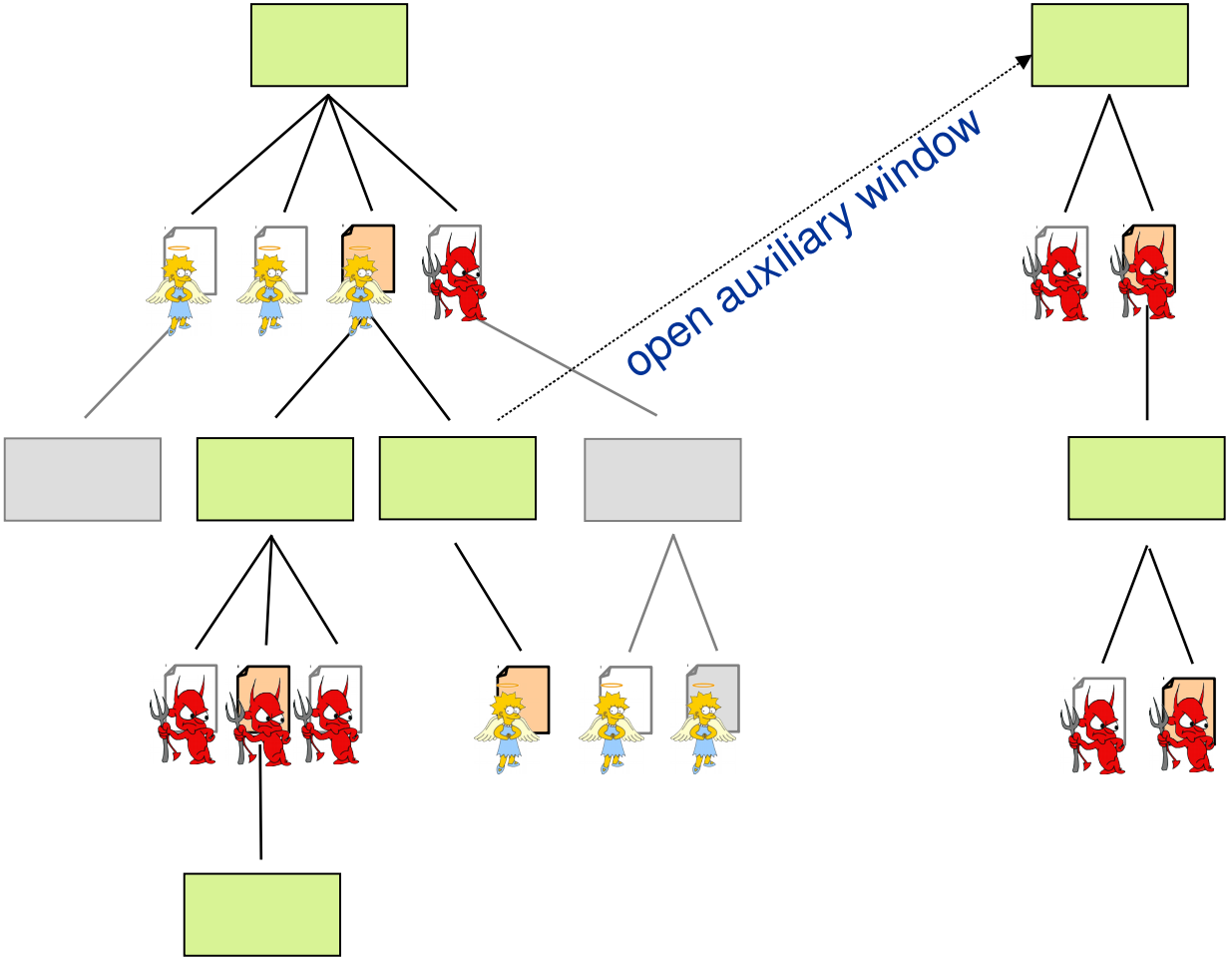
Is Browser Corrupted?
☐ no
☐ close-corrupted
☐ fully corrupted

DNS Address d_1

Windows and Documents



Window and Document Tree



Web Browser State

Windows & Documents

domainA: cookie1
cookie2
domainB: cookie3
domainC: cookie4

Cookies

originA: key1: value1
key2: value2
originB: key1: value3
originC: key3: value4

LocalStorage

originA, win_1: key1:...
key2:...
originB, win_1: key1:...
originC, win_2: key3:...

SessionStorage

originA: ●●●●●
originB: ●●●●

User Secrets

domainA: k_1
domainB: k_2
domainC: k_3

Key Mapping

domainA,
domainB,
domainC

Strict Transport Security Domains

n_1: request1,
n_2: request2,
n_3: request3

Pending DNS Requests

n_4: request4,
n_5: request5,
n_6: request6

Pending HTTP Requests

Is Browser Corrupted?
☐ no
☐ close-corrupted
☐ fully corrupted

DNS Address d_1

Browser Actions

Web browser transitions

Process DNS response

- Process DNS response
- Send HTTP(S) request that waited for this DNS response

Process HTTP(S) response

- Look up original request
- Perform redirections if requested
- Further process depending on request type

…including
- Origin header
- Strict-Transport-Security
- Cookies
- Location Redirects

Become corrupt

…ruption state (*full corrupt* or *close-corrupt*)

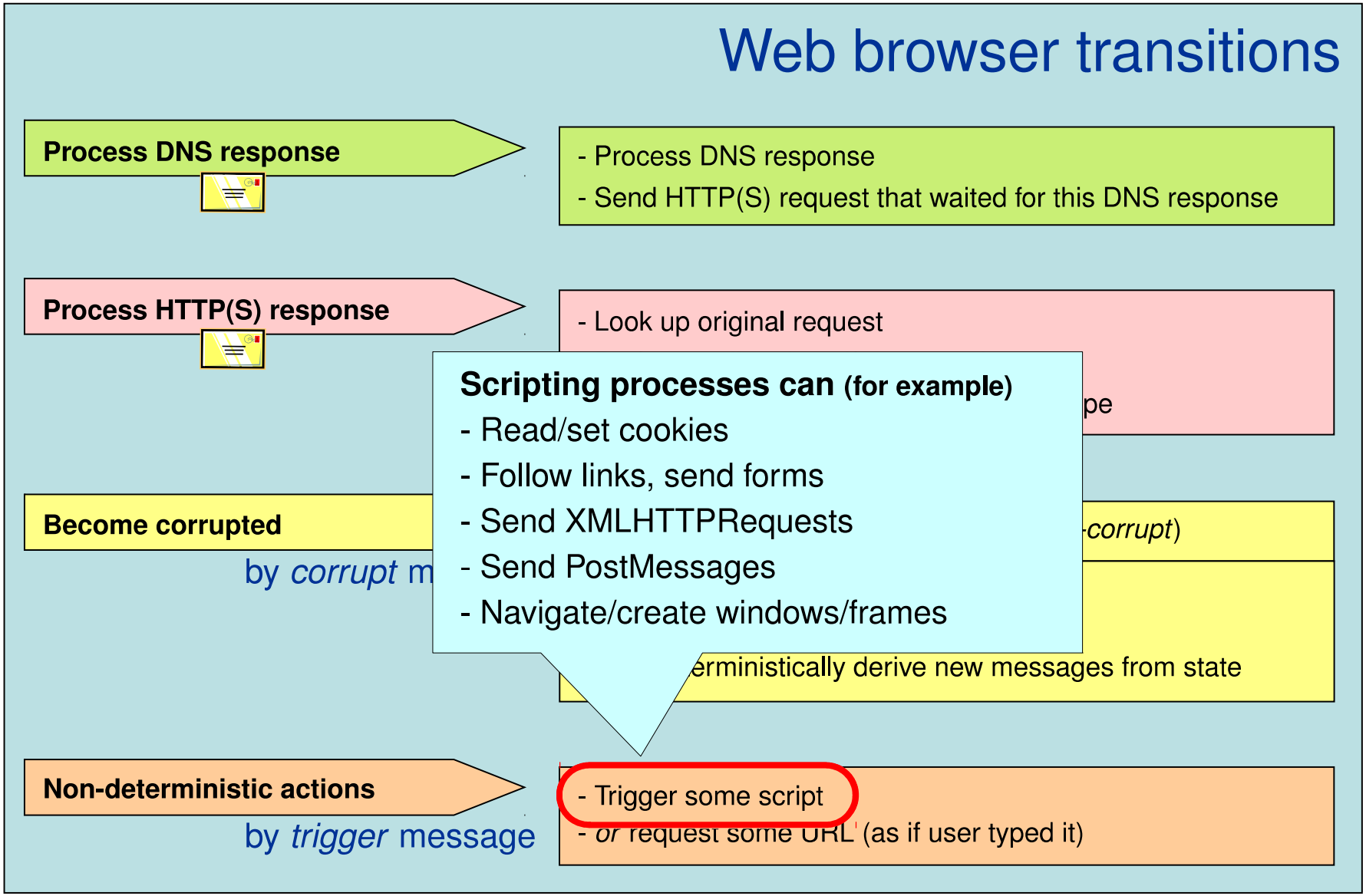
Browser:
- …incoming messages
- Non-deterministically derive new messages from state

Non-deterministic actions

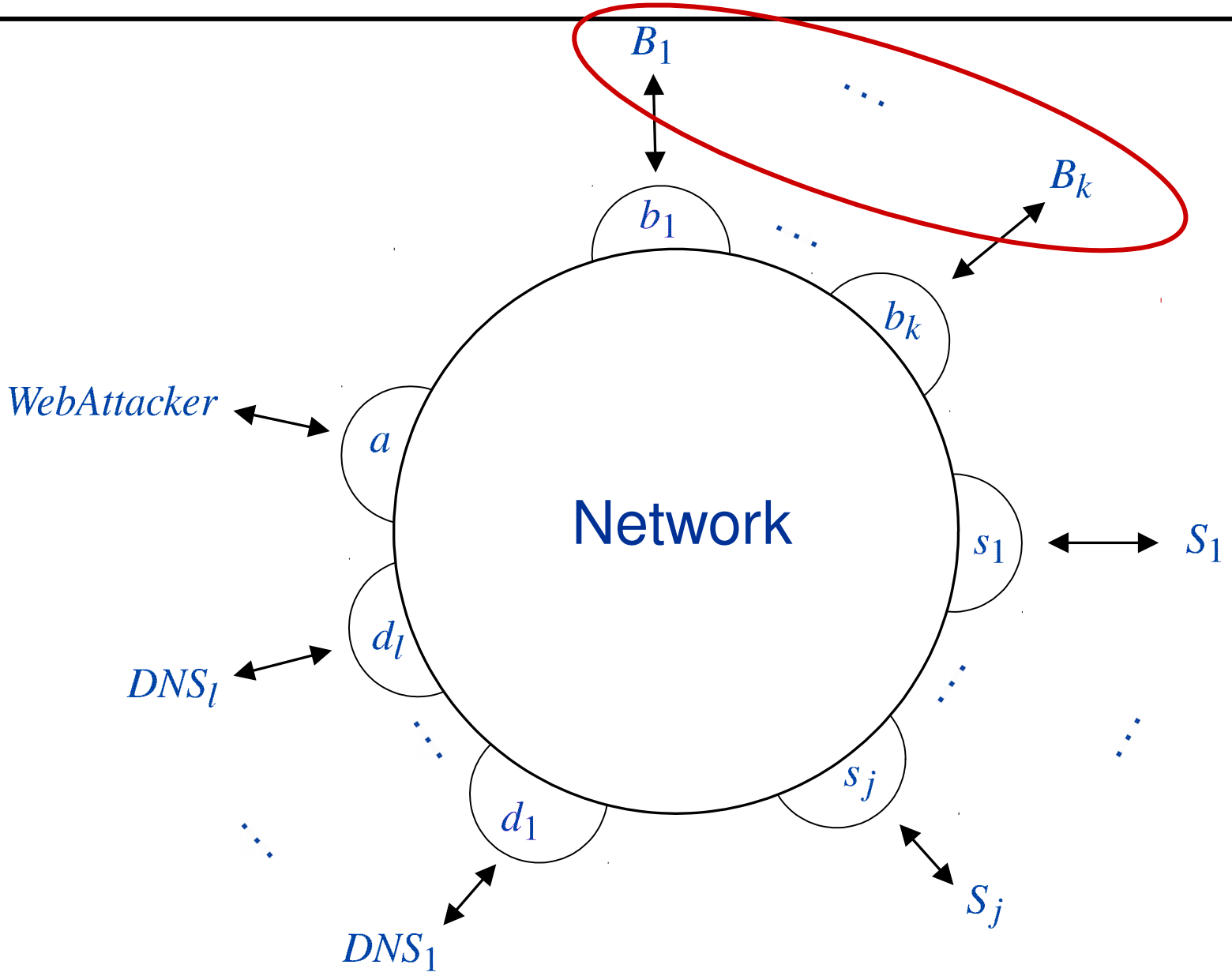
by *trigger* message

- Trigger some script
- *or* request some URL (as if user typed it)

Browser Actions



Web Model



Limitations

- No language details

- No user interface details

- No byte-level attacks (e.g., Buffer Overflows)

- Abstract view on cryptography

Previous Case Studies

[SP 2014, ESORICS 2015, CCS 2015]

- Formal analysis of Mozilla's BrowserID

Main design goal: privacy

* Found severe attacks

  awarded by Mozilla's bug bounty program

* Proposed fixes for authentication and proved security

* Privacy broken beyond repair

- Designed our own new system: SPRESSO

https://spresso.me

Provably provides strong authentication and privacy properties.

Very first SSO system with privacy.

Our Contributions

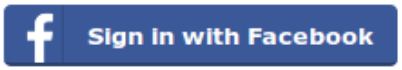
OAuth 2.0:

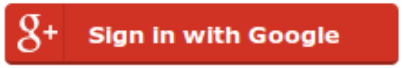
- Developed formal model of the standard
  * Based on most comprehensive model of the web to date (extension of S&P 2014).
- Formalized central security properties
  * Authorization
  * Authentication
  * Session Integrity
- Tried to prove these properties, but found severe attacks (also on OpenID Connect)
- Proposed fixes
- Proved security for fixed standard

All details: TR available at infsec.uni-trier.de

OAuth

OAuth 2.0: widely used standard for web authorization and SSO (RFC 6749)

- Predominant Web SSO, e.g., 

- Foundation for OpenID Connect. e.g., 

- Many modes and options:

- 4 different modes of interaction
- absence and presence of client secrets
- redirect URI limitations incl. URI patterns
- …

What happens if all modes and options are used in combination?

OAuth Modes

Implicit Mode

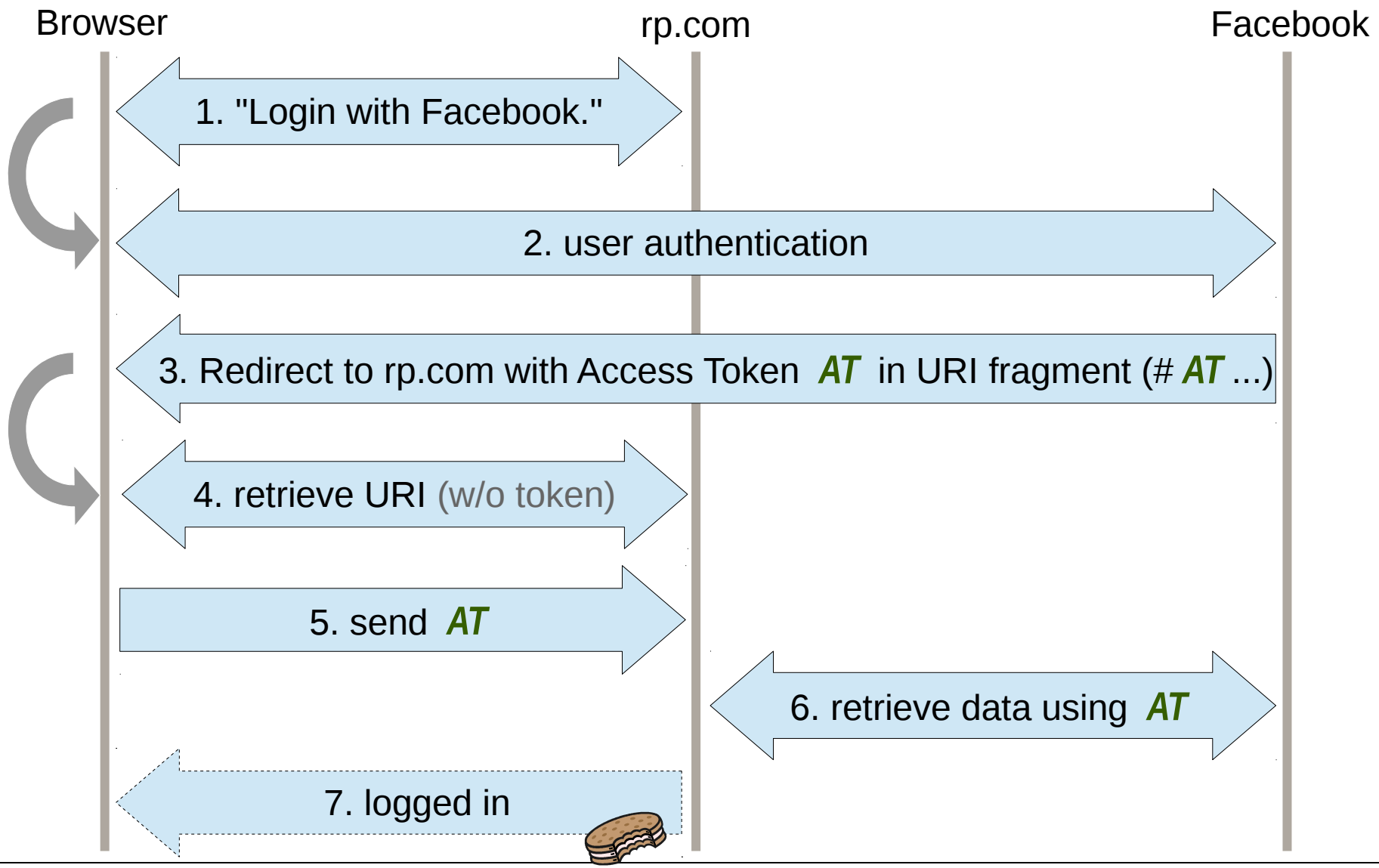
Authorization Code Mode

} most common

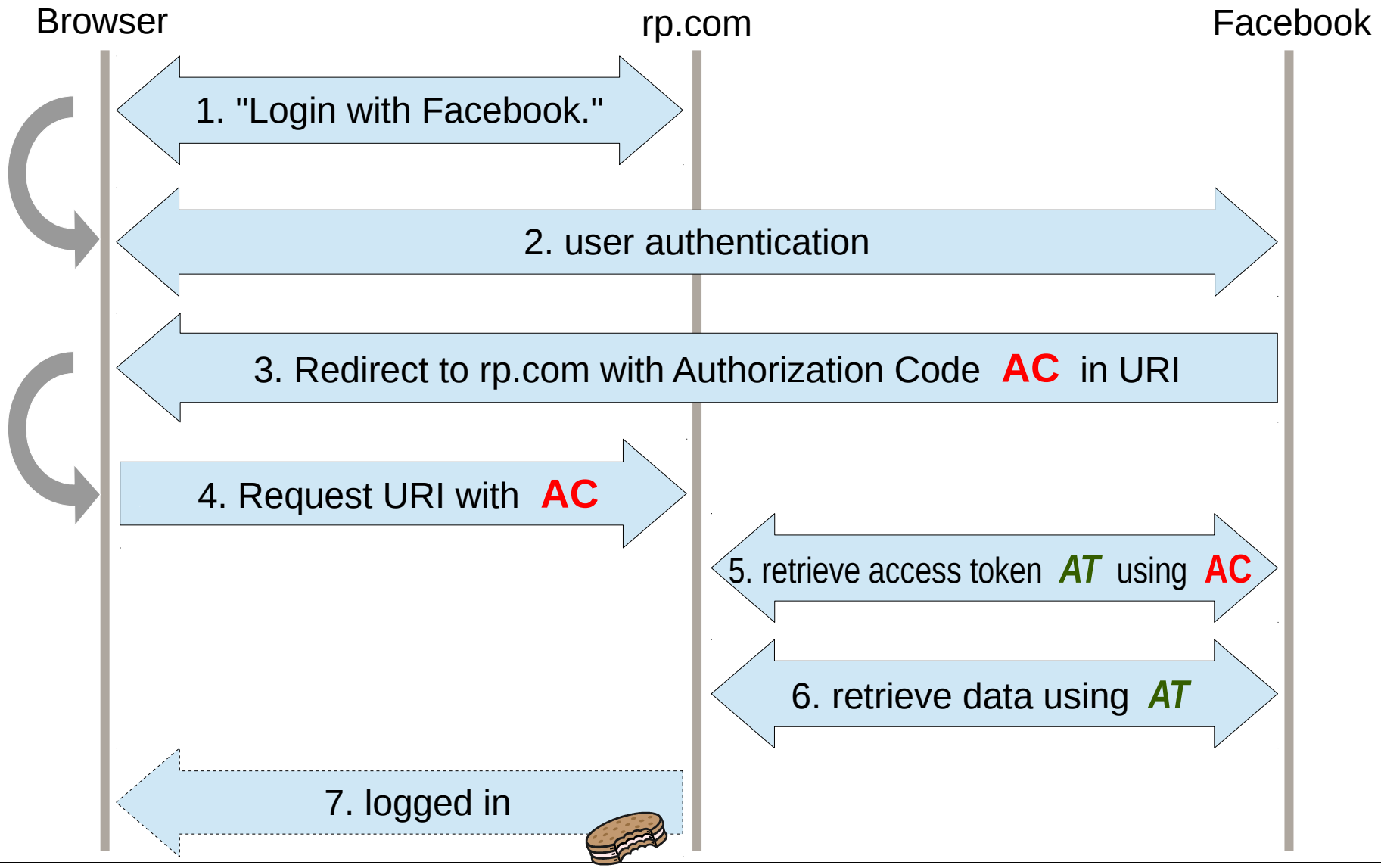
Resource Owner Password Credentials Mode

Client Credentials Mode

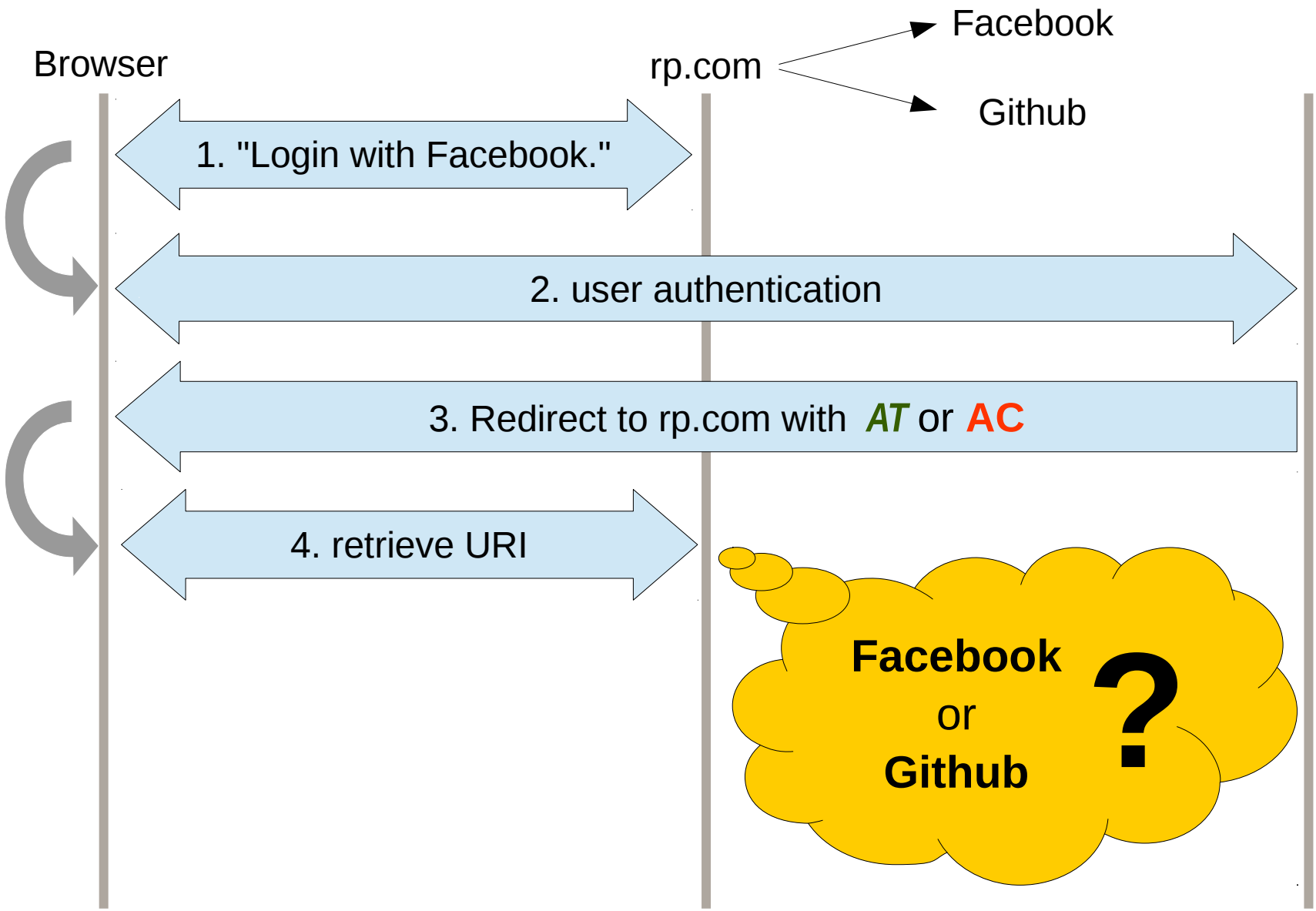
Implicit Mode



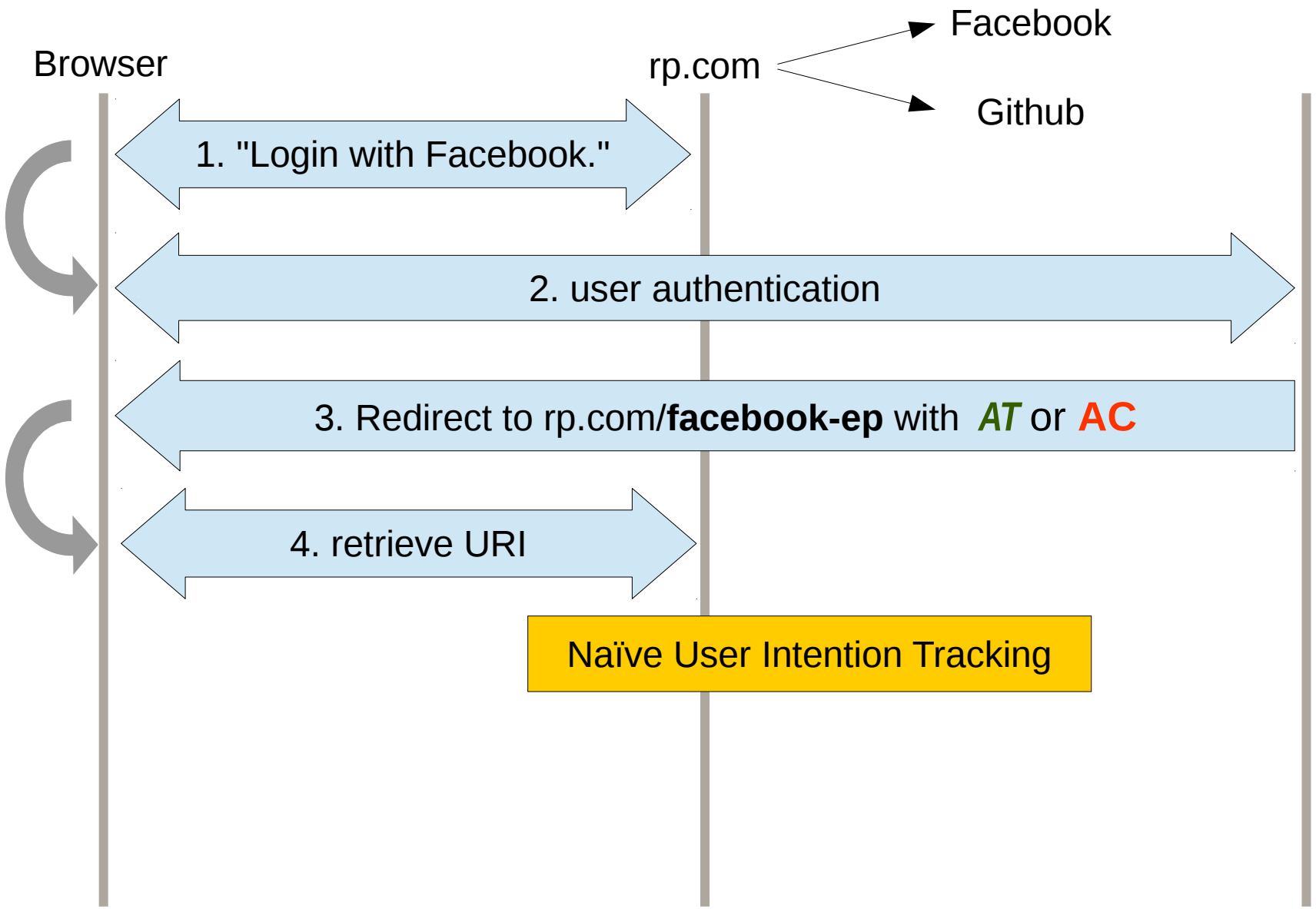
Authorization Code Mode



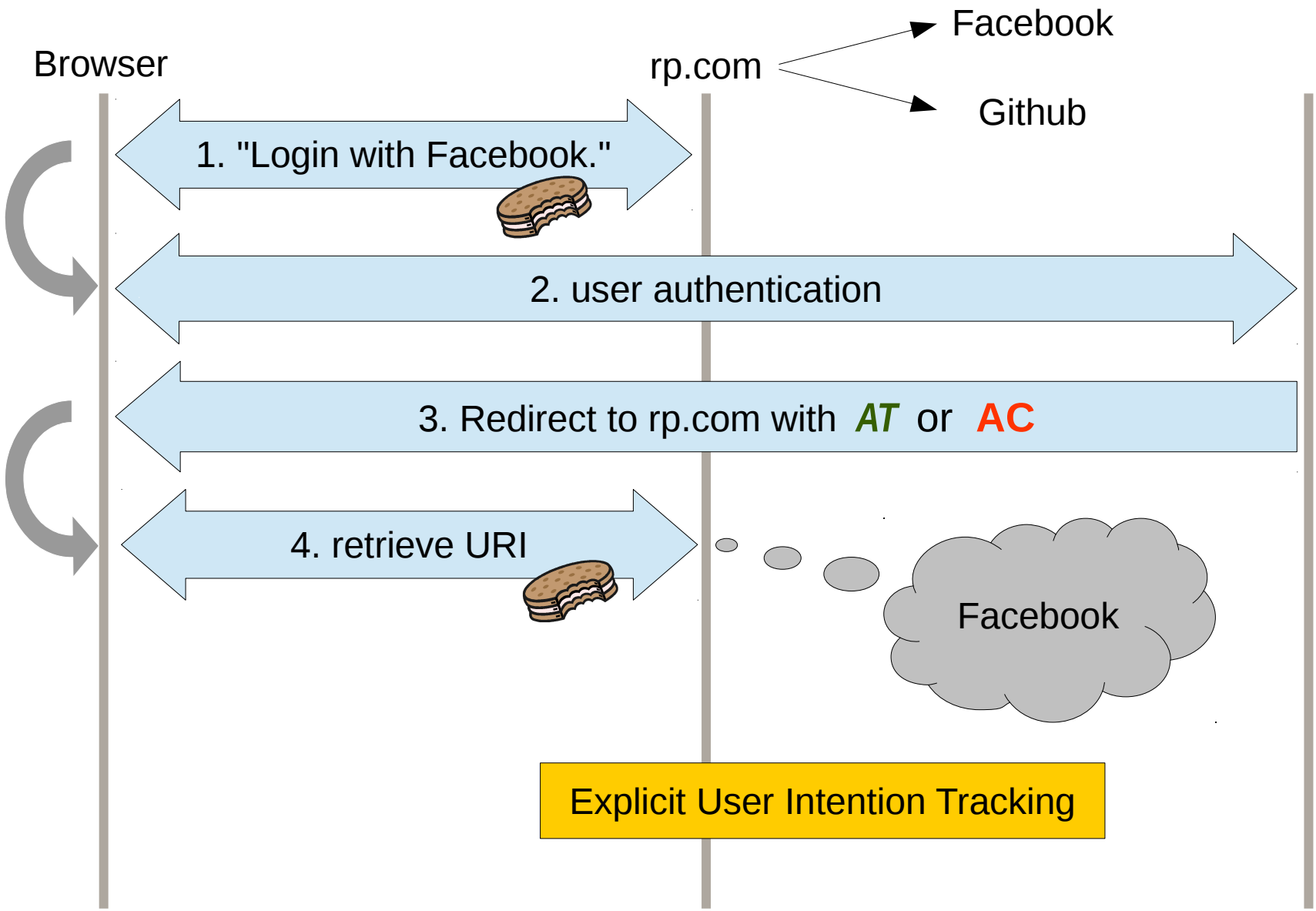
Multiple IdPs



Multiple IdPs



Multiple IdPs

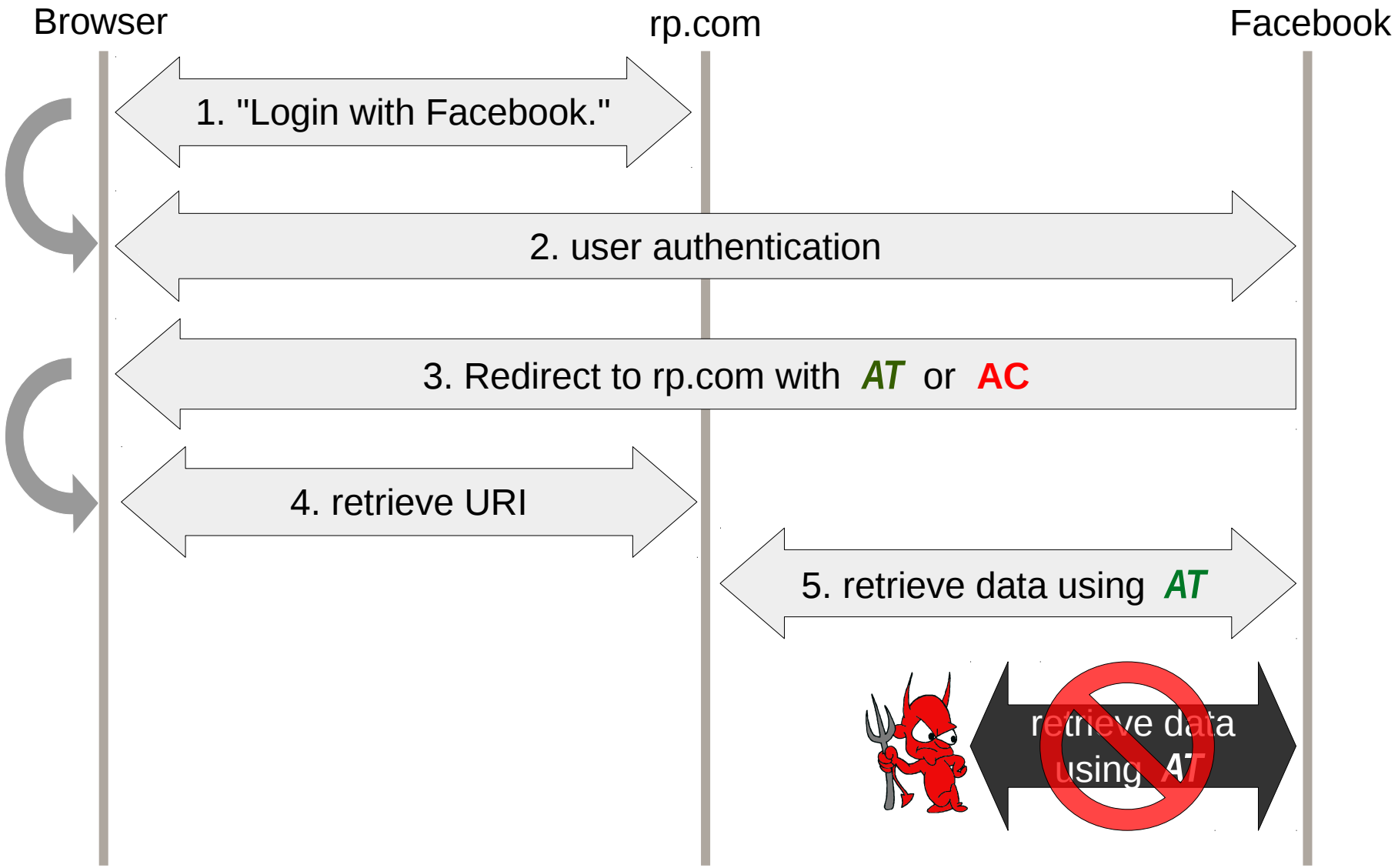


Our Contributions

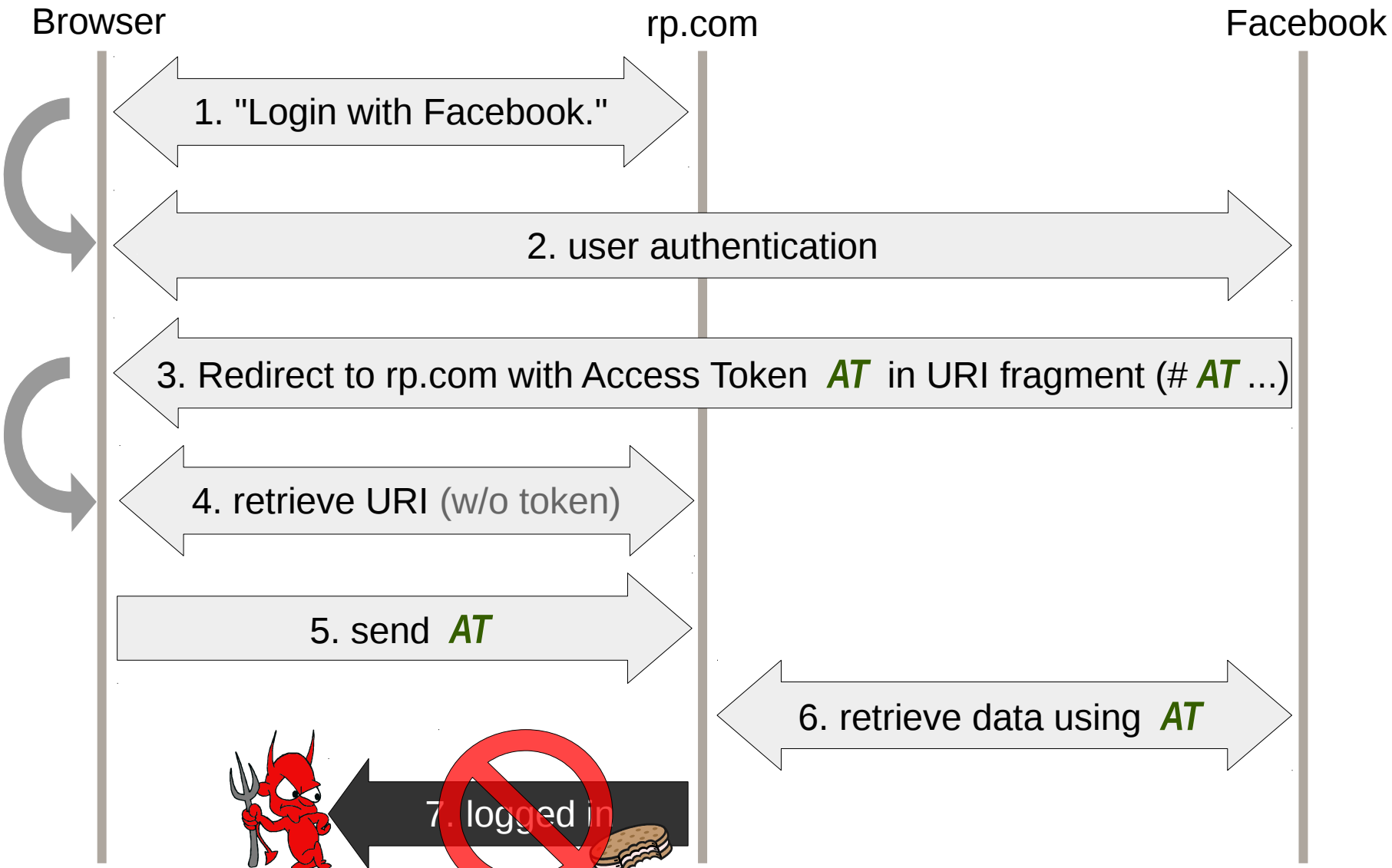
OAuth 2.0:

- Developed formal model of the standard
  * Based on most comprehensive model of the web to date (extension of S&P 2014).
- Formalized central security properties
  * Authorization
  * Authentication
  * Session Integrity
- Tried to prove these properties, but found severe attacks (also on OpenID Connect)
- Proposed fixes
- Proved security for fixed standard

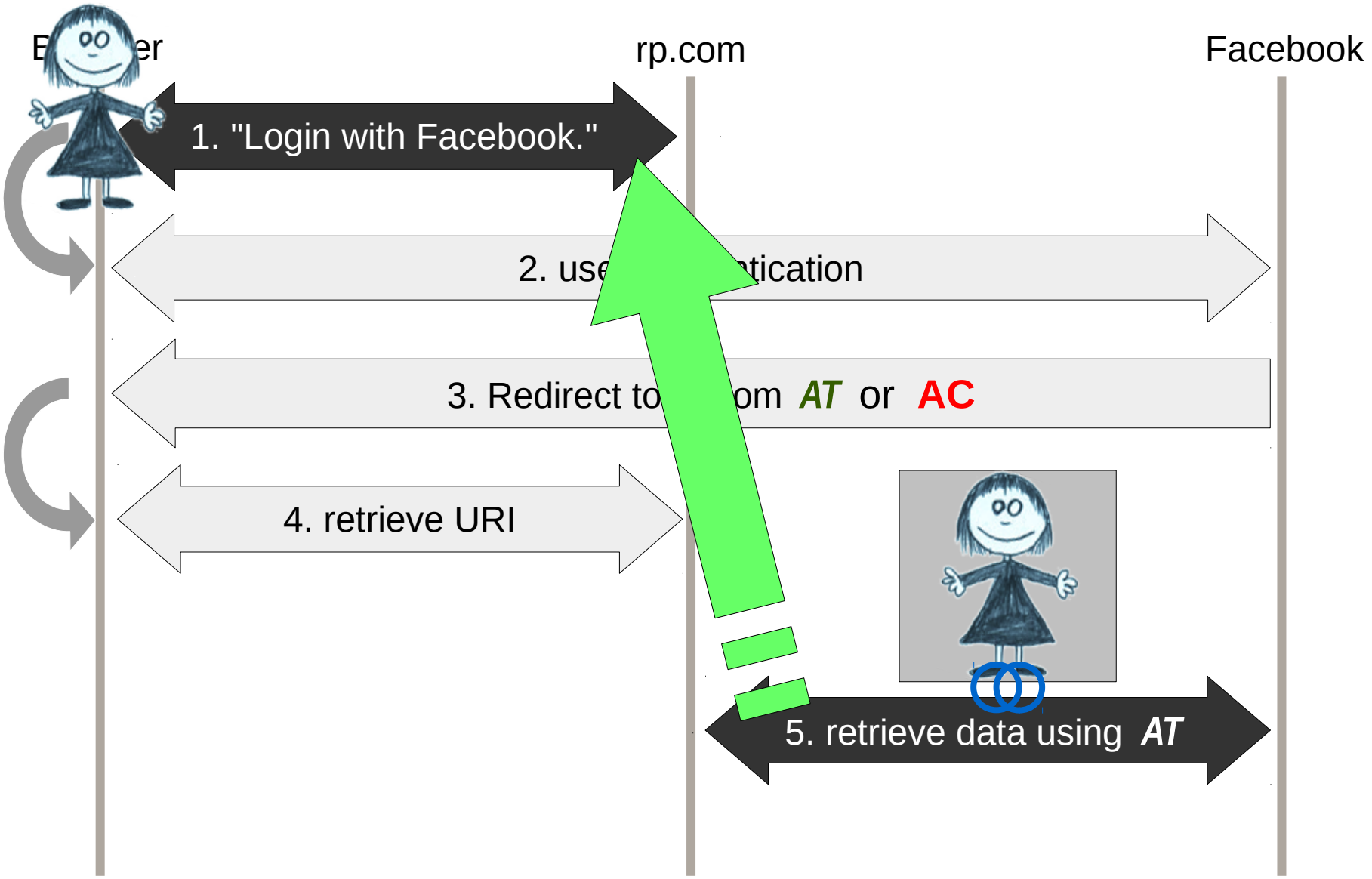
Authorization Property



Authentication Property



Session Integrity Property



Our Contributions

OAuth 2.0:

- Developed formal model of the standard
  * Based on most comprehensive model of the web to date (extension of S&P 2014).
- Formalized central security properties
  * Authorization
  * Authentication
  * Session Integrity
- Tried to prove these properties, but found severe attacks (also on OpenID Connect)
- Proposed fixes
- Proved security for fixed standard

All details: TR available at infsec.uni-trier.de

OAuth 2.0 Formal Analysis Efforts

- Mostly analysis of specific implementations, not the standard itself [2-5]
- Formal analysis in [1], but based on less detailed web model, not all options/modes.
- So far, no security proof

[1] C. Bansal, K. Bhargavan, A. Delignat-Lavaud, and S. Maffeis. Discovering Concrete Attacks on Website Authorization by Formal Analysis. Journal of Computer Security, 2014.

[2] E. Y. Chen, Y. Pei, S. Chen, Y. Tian, R. Kotcher, and P. Tague. OAuth Demystified for Mobile Application Developers. CCS '14.

[3] W. Li and C. J. Mitchell. Security issues in OAuth 2.0 SSO implementations. ISC 2014.

[4] M. Shehab and F. Mohsen. Towards Enhancing the Security of OAuth Implementations in Smart Phones. 2014 IEEE International Conference on Mobile Services.

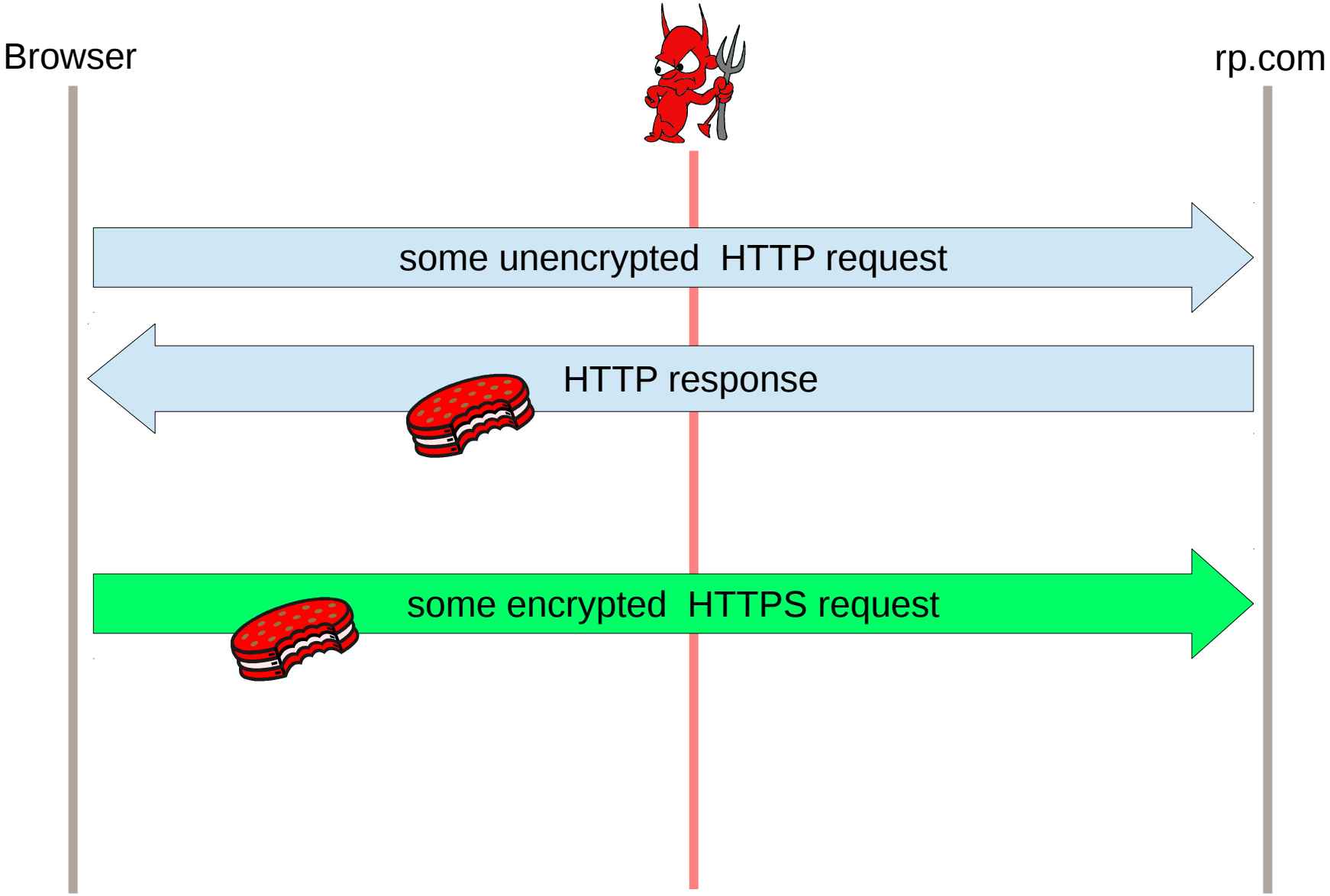
[5] S.-T. Sun and K. Beznosov. The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems. CCS'12.

...

Assumptions

- Follow RFC 6749 (OAuth 2.0 Standard),
  i.e., no stupid bugs
- Follow security recommendations for OAuth and common best practices,
  e.g., CSRF protection, HTTPS, Sessions.
- Malicious RPs, IdPs and browsers
- No open redirectors or untrusted JS on honest RPs/IdPs
  (a cause for known attacks)
- Authorization: Network attacker
  Authentication: Network attacker
  Session Integrity: Web attackers

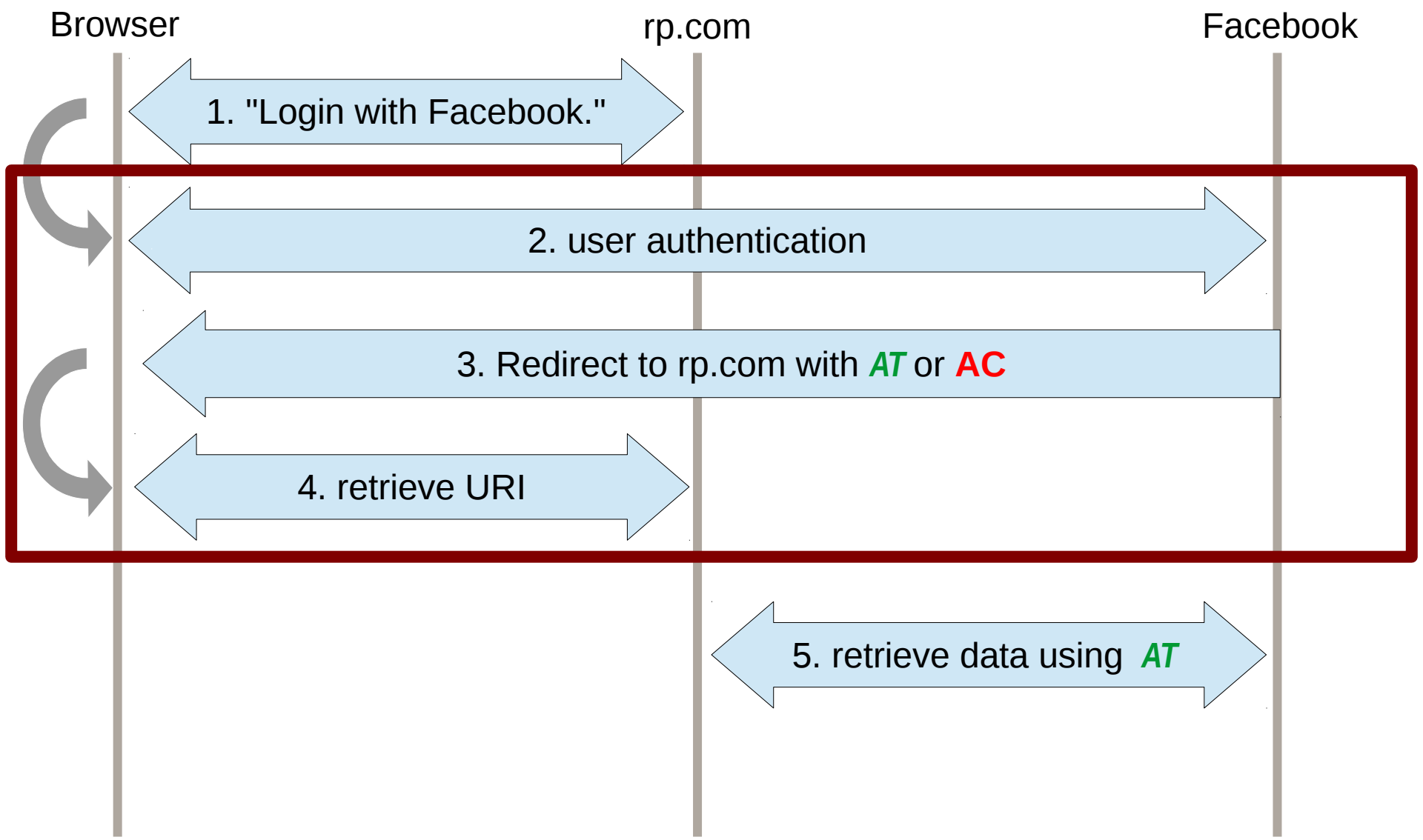
Session Integrity: Network Attacker



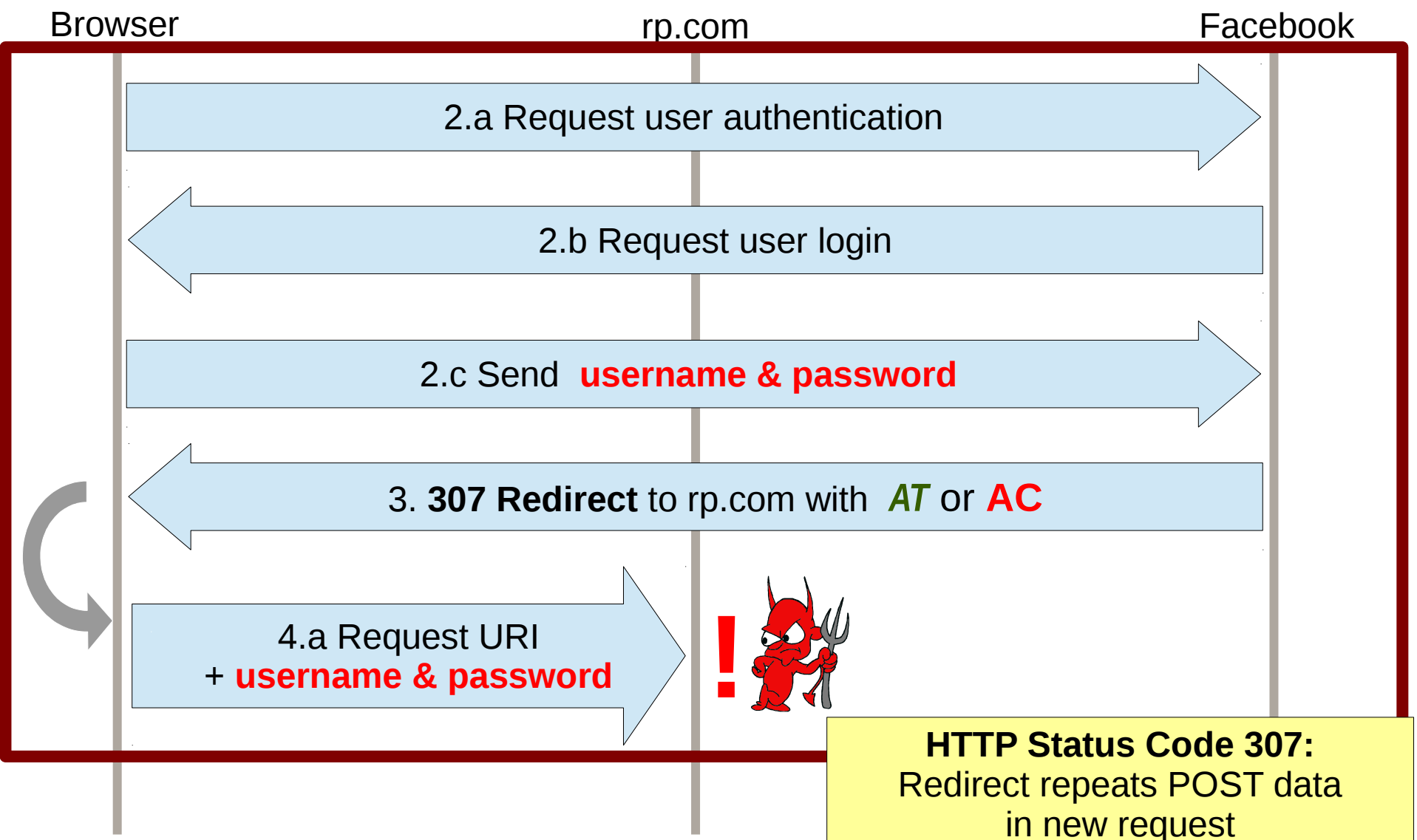
Attacks: Overview

- **307 Redirect Attack**
  (⚡ Authorization & Authentication)
- **IdP Mix-Up Attack**
  (⚡ Authorization & Authentication)
- **State Leak Attack**
  (⚡ Session Integrity)
- **Naïve RP Session Integrity Attack**
  (⚡ Session Integrity)

307 Redirect Attack



307 Redirect Attack



HTTP Status Code 307:
Redirect repeats POST data
in new request

307 Redirect Attack

OAuth standard says:

1.7. HTTP Redirections

This specification makes extensive use of HTTP redirections, in which the client or the authorization server directs the resource owner's user-agent to another destination. While the examples in this specification show the use of the HTTP 302 status code, any other method available via the user-agent to accomplish this redirection is allowed and is considered to be an implementation detail.

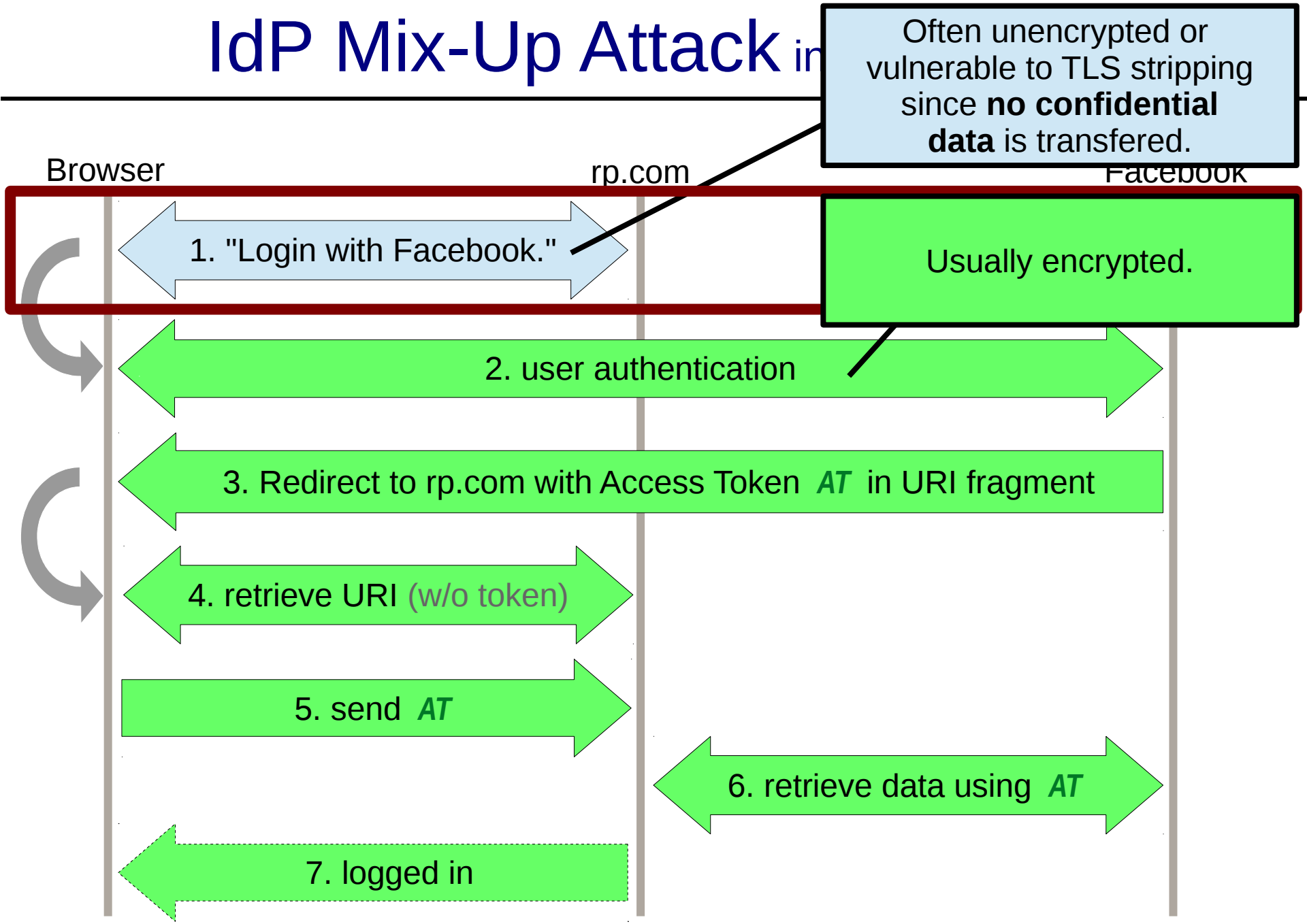
Mitigation:

Make 303 or any other method that does not forward POST data.

Attacks: Overview

- **307 Redirect Attack**
  (⚡ Authorization & Authentication)
- **IdP Mix-Up Attack**
  (⚡ Authorization & Authentication)
- **State Leak Attack**
  (⚡ Session Integrity)
- **Naïve RP Session Integrity Attack**
  (⚡ Session Integrity)

IdP Mix-Up Attack in



IdP Mix-Up Attack in Implicit Mode

IdP Mix-Up Attack

+ Variants

More details have to be taken care of:

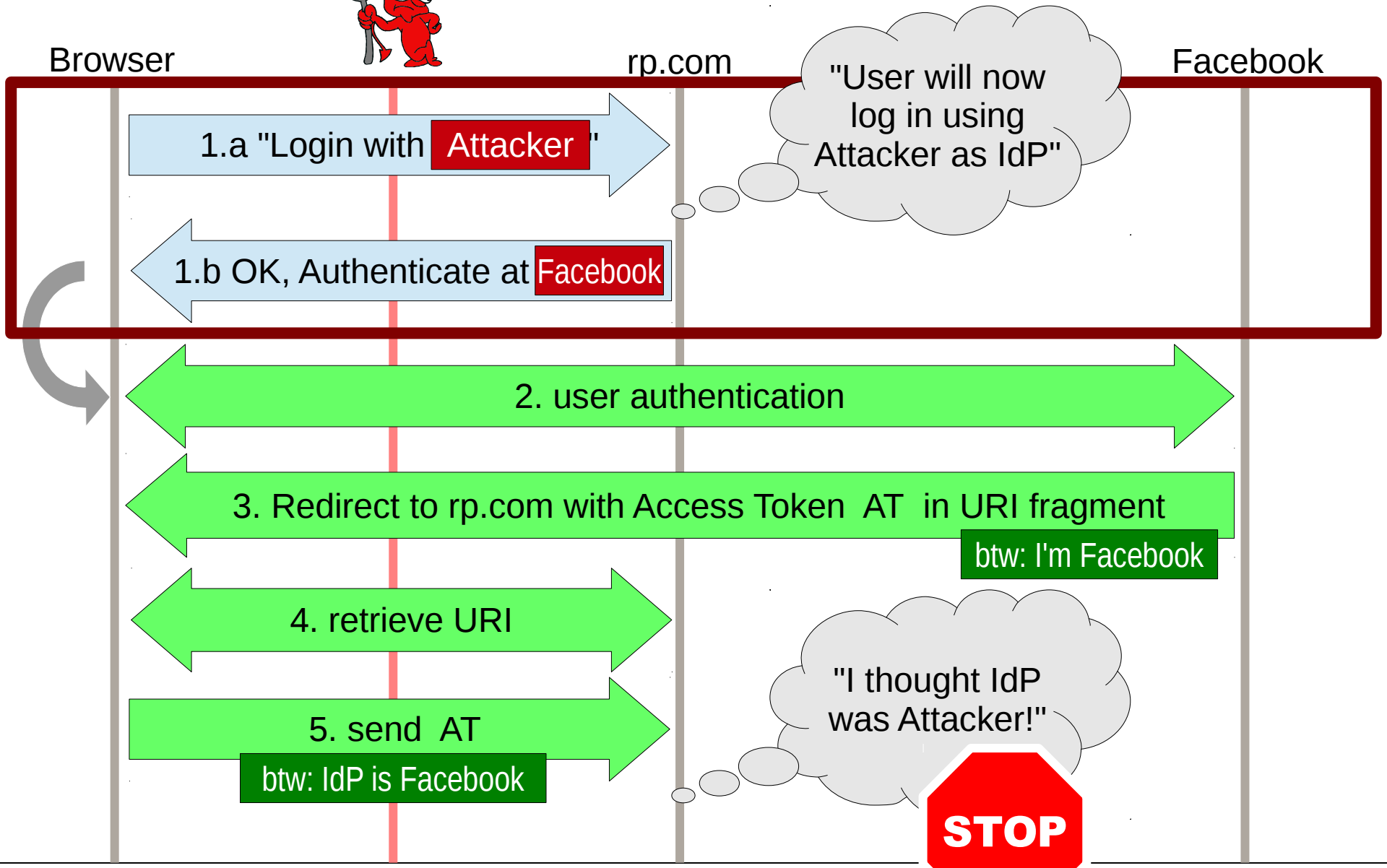
- Breaking authentication instead of authorization (some additional steps)
- Client Identifiers (i.e., RPs identify themselves to IdPs)
- Client Credentials (i.e., RPs authenticate to IdPs)
- In OpenID Connect: Message signing, "ID Token", endpoint discovery, etc.

→ **Sucessfully attacked real-world implementation.**

Multiple/malicious IdPs.
Related problems:

[Bansal et al. 2012]
[Mladenov et al. 2015]

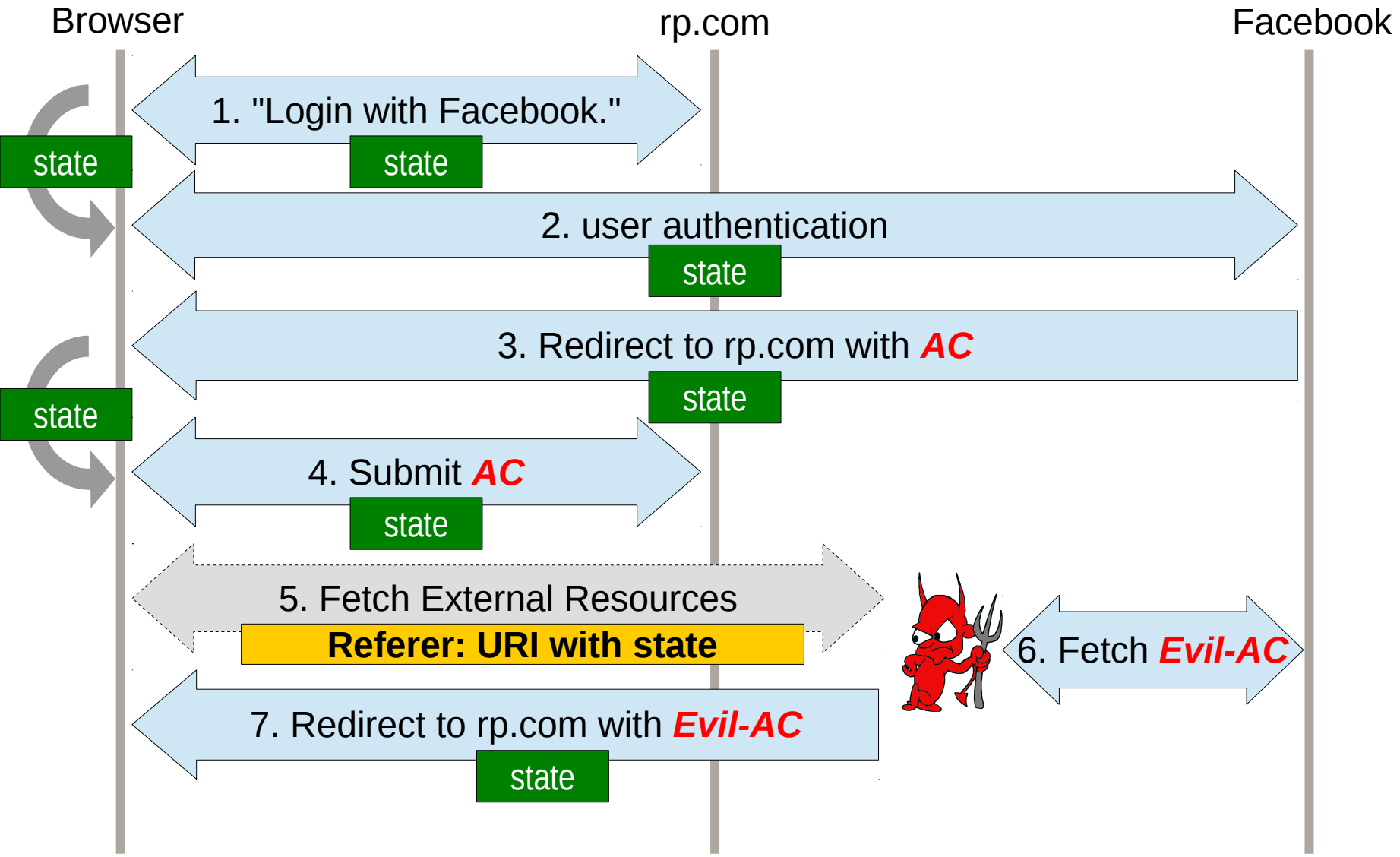
IdP Mix-Up Attack: Mitigation



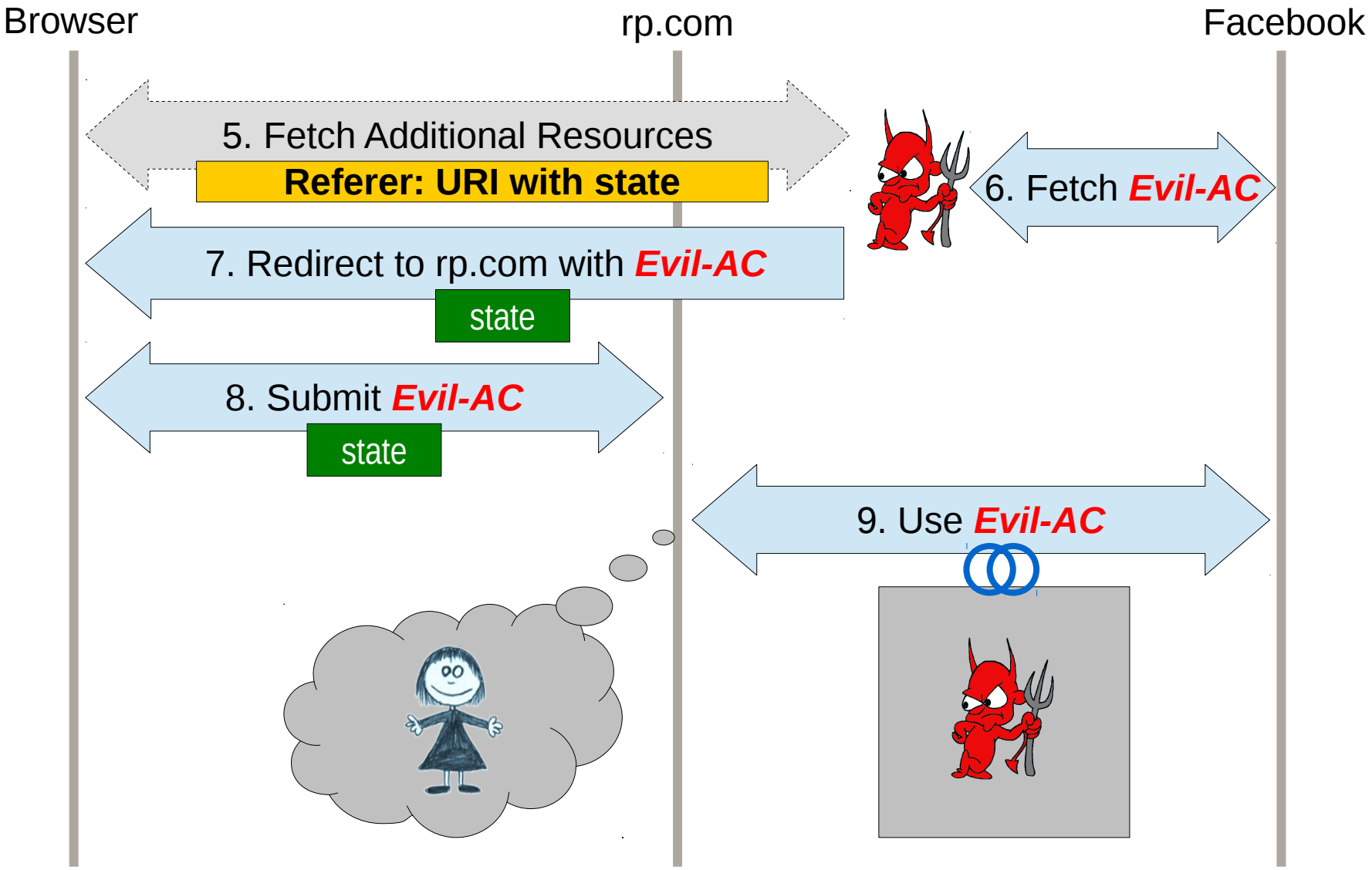
Attacks: Overview

- **307 Redirect Attack**
  (⚡ Authorization & Authentication)
- **IdP Mix-Up Attack**
  (⚡ Authorization & Authentication)
- **State Leak Attack**
  (⚡ Session Integrity)
- **Naïve RP Session Integrity Attack**
  (⚡ Session Integrity)

State Leak Attack



State Leak Attack



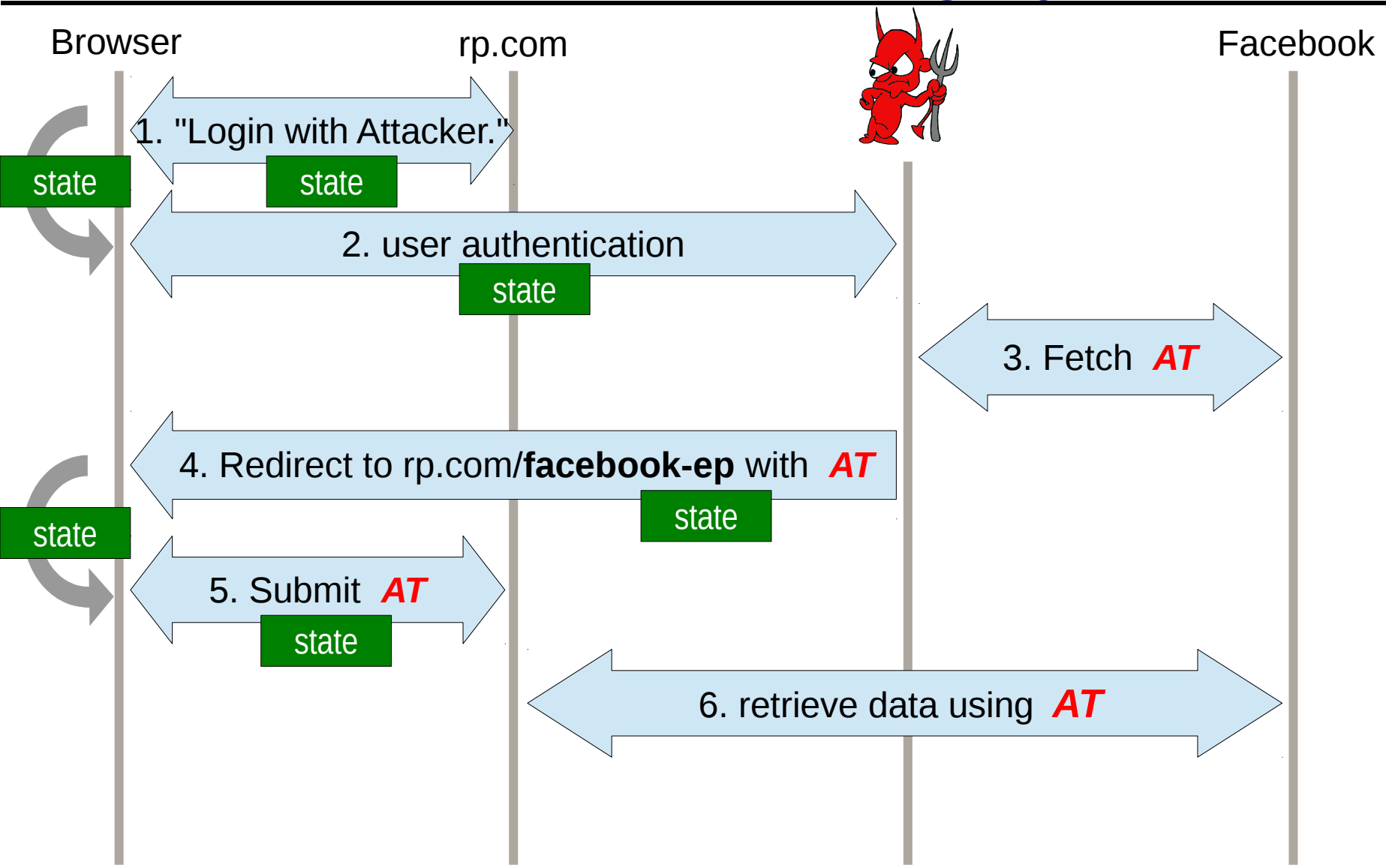
State Leak Attack: Mitigation

- Make state single-use
  (so far, not suggested by the standard)
- Avoid state to be leaked
  → Referrer Policies (W3C Draft)

Attacks: Overview

- **307 Redirect Attack**
  (⚡ Authorization & Authentication)
- **IdP Mix-Up Attack**
  (⚡ Authorization & Authentication)
- **State Leak Attack**
  (⚡ Session Integrity)
- **Naïve RP Session Integrity Attack**
  (⚡ Session Integrity)

Naïve RP Session Integrity Attack



Naïve RP SI Attack: Mitigation

→ Use explicit user intention tracking only.

Attacks: Overview

- **307 Redirect Attack**
  (⚡ Authorization & Authentication)
- **IdP Mix-Up Attack**
  (⚡ Authorization & Authentication)
- **State Leak Attack**
  (⚡ Session Integrity)
- **Naïve RP Session Integrity Attack**
  (⚡ Session Integrity)

Our Contributions

OAuth 2.0:

- Developed formal model of the standard
  * Based on most comprehensive model of the web to date (extension of S&P 2014).
- Formalized central security properties
  * Authorization
  * Authentication
  * Session Integrity
- Tried to prove these properties, but found severe attacks (also on OpenID Connect)
- Proposed fixes
- Proved security for fixed standard

All details: TR available at infsec.uni-trier.de

OAuth 2.0: Security Proof

Model closely follows OAuth 2.0 Standard and current best practices.

- Unbounded number of IdPs, RPs, Browsers
- IdPs, RPs, Browsers can be corrupted
- Authorization: Network Attacker
  Authentication: Network Attacker
  Session Integrity: Web Attackers
- All modes and options run concurrently.

OAuth 2.0: Security Proof

Theorem:

OAuth 2.0 satisfies Authorization, Authentication and Session Integrity Properties.

Our Contributions

OAuth 2.0:

- Developed formal model of the standard
  - * Based on most comprehensive model of the web to date (extension of S&P 2014).
- Formalized central security properties
  - * Authorization
  - * Authentication
  - * Session Integrity
- Tried to prove these properties, but found severe attacks (also on OpenID Connect)
- Proposed fixes
- Proved security for fixed standard

Thank you!

All details: TR available at infsec.uni-trier.de