

WPSE: FORTIFYING WEB PROTOCOLS VIA BROWSER-SIDE SECURITY MONITORING

Stefano Calzavara

Riccardo Focardi

Mauro Tempesta

Marco Squarcina

Matteo Maffei

Clara Schneidewind

August 17, 2018 - 27th Usenix Security Symposium

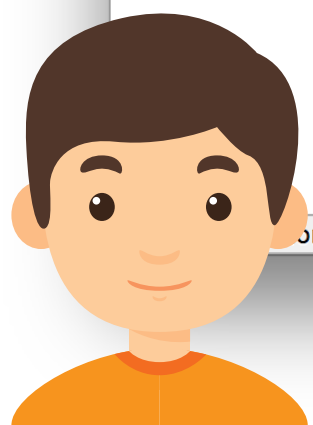
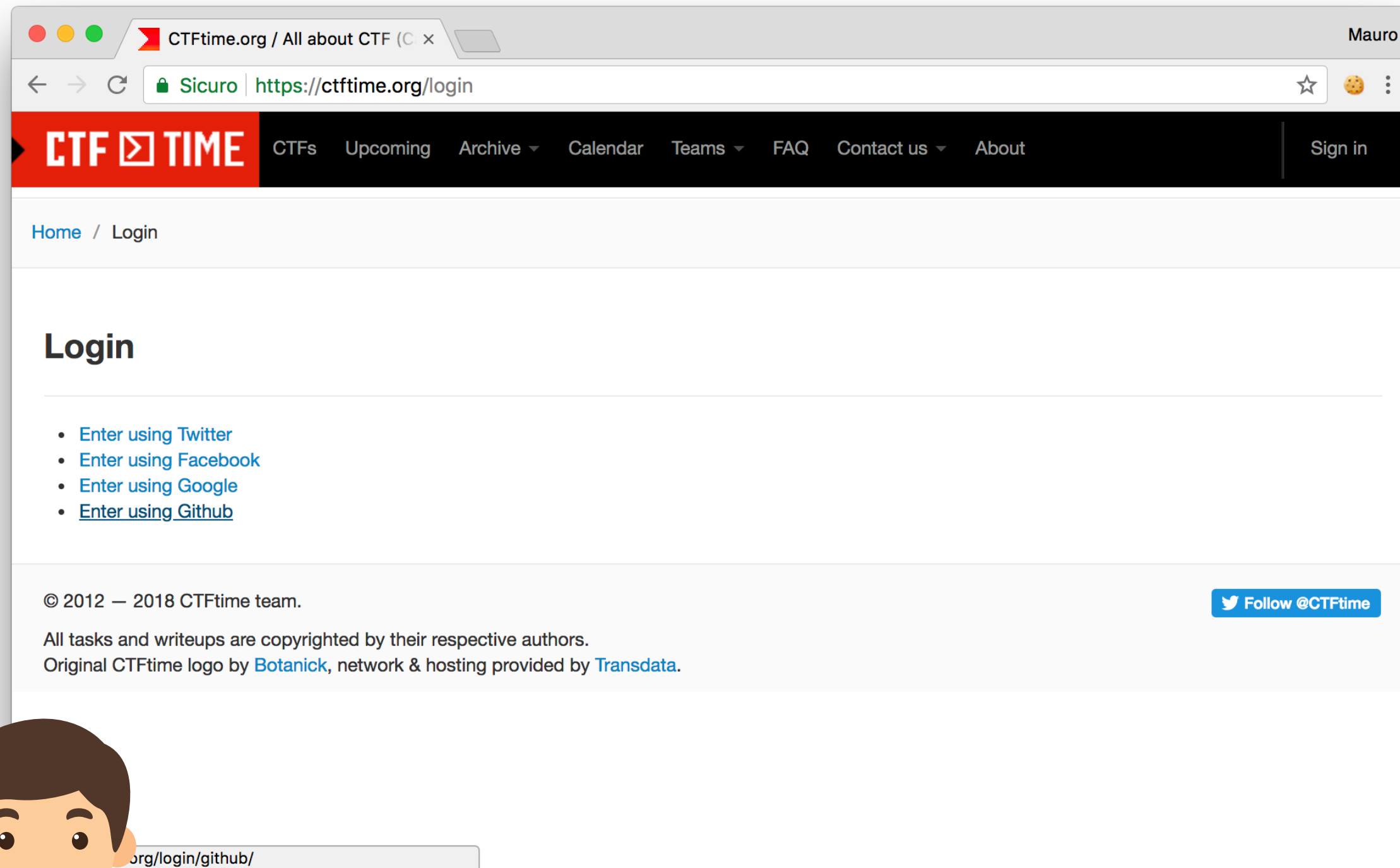


Università
Ca' Foscari
Venezia



TECHNISCHE
UNIVERSITÄT
WIEN

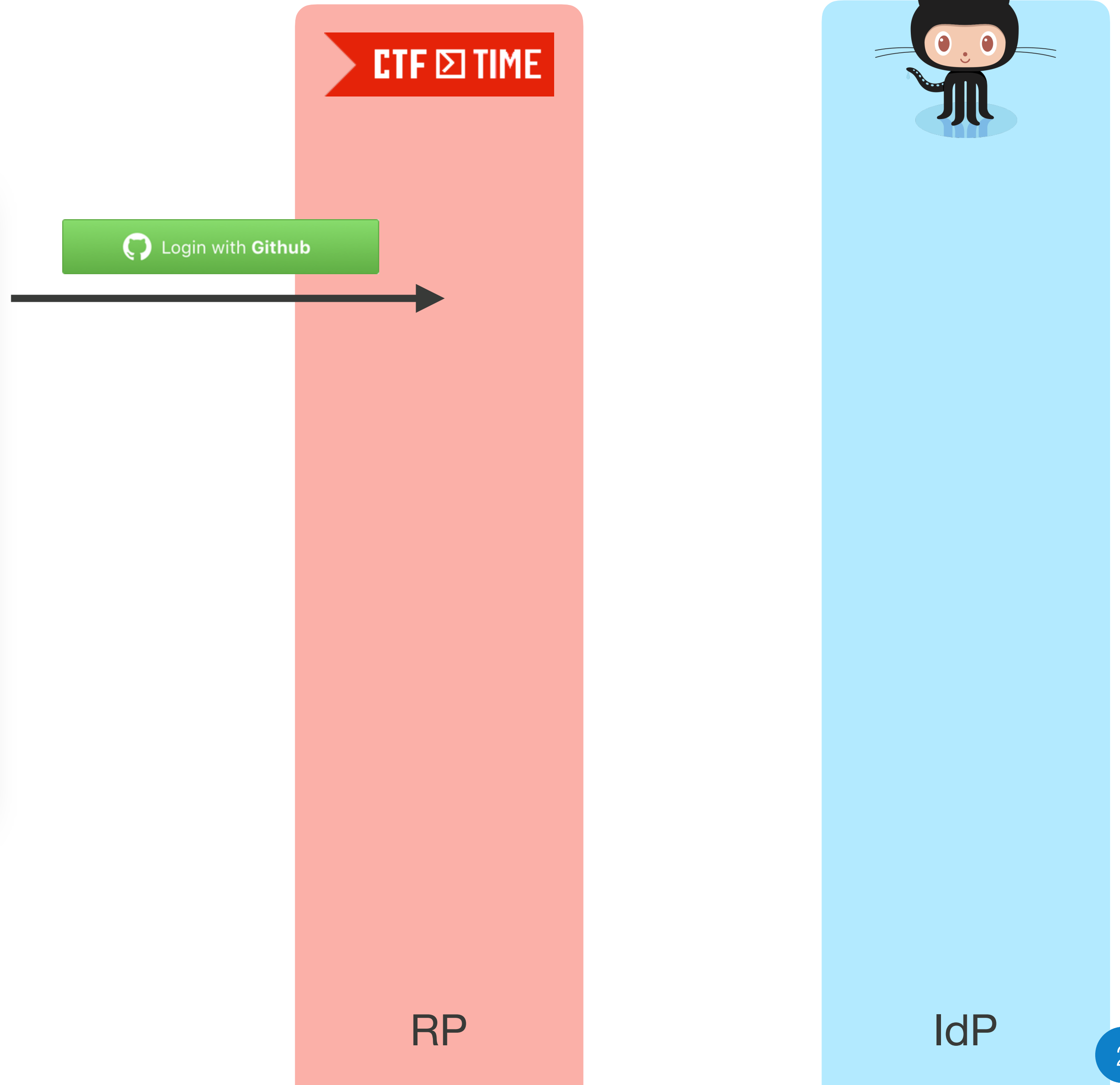
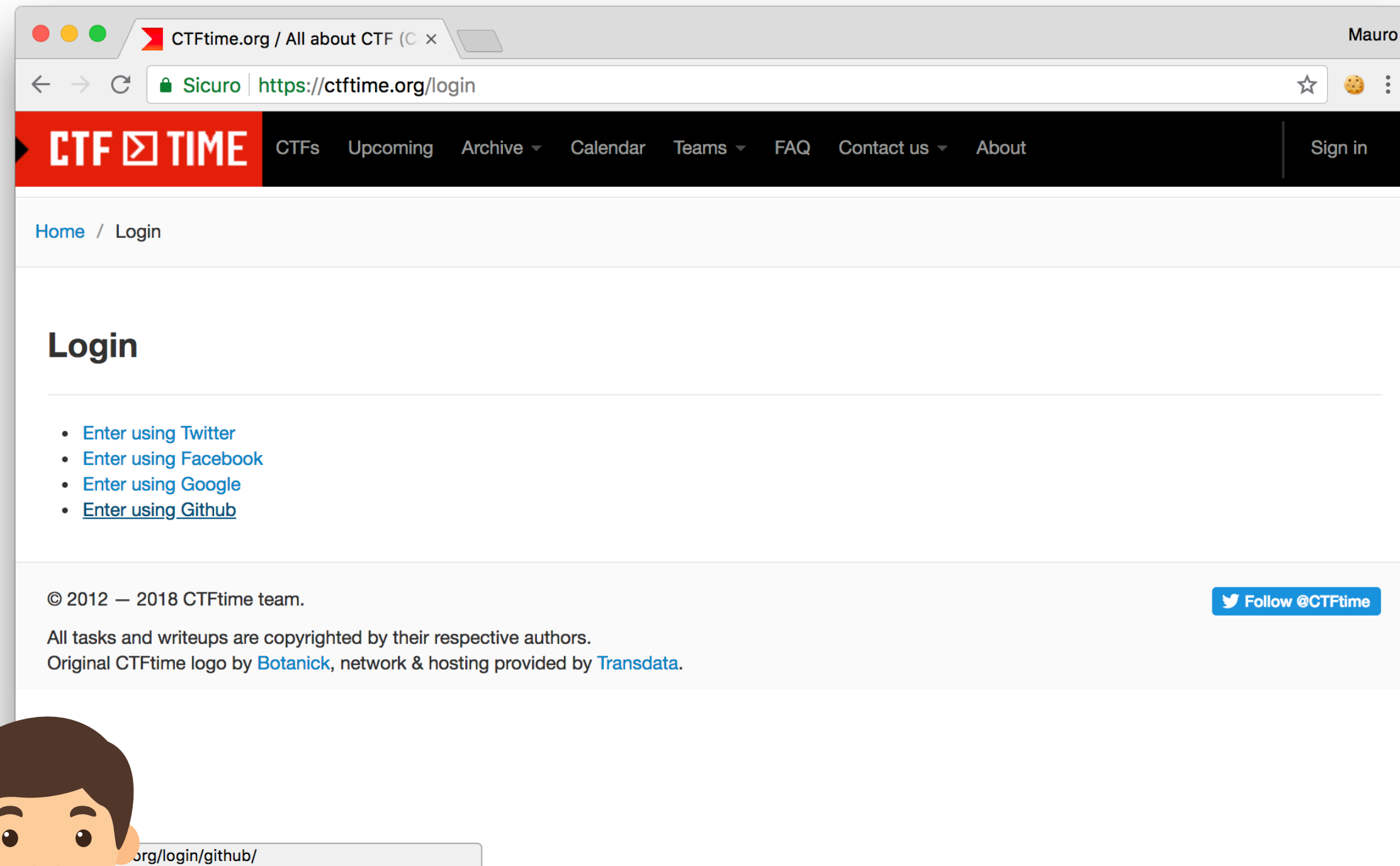
OVERVIEW OF A WEB PROTOCOL



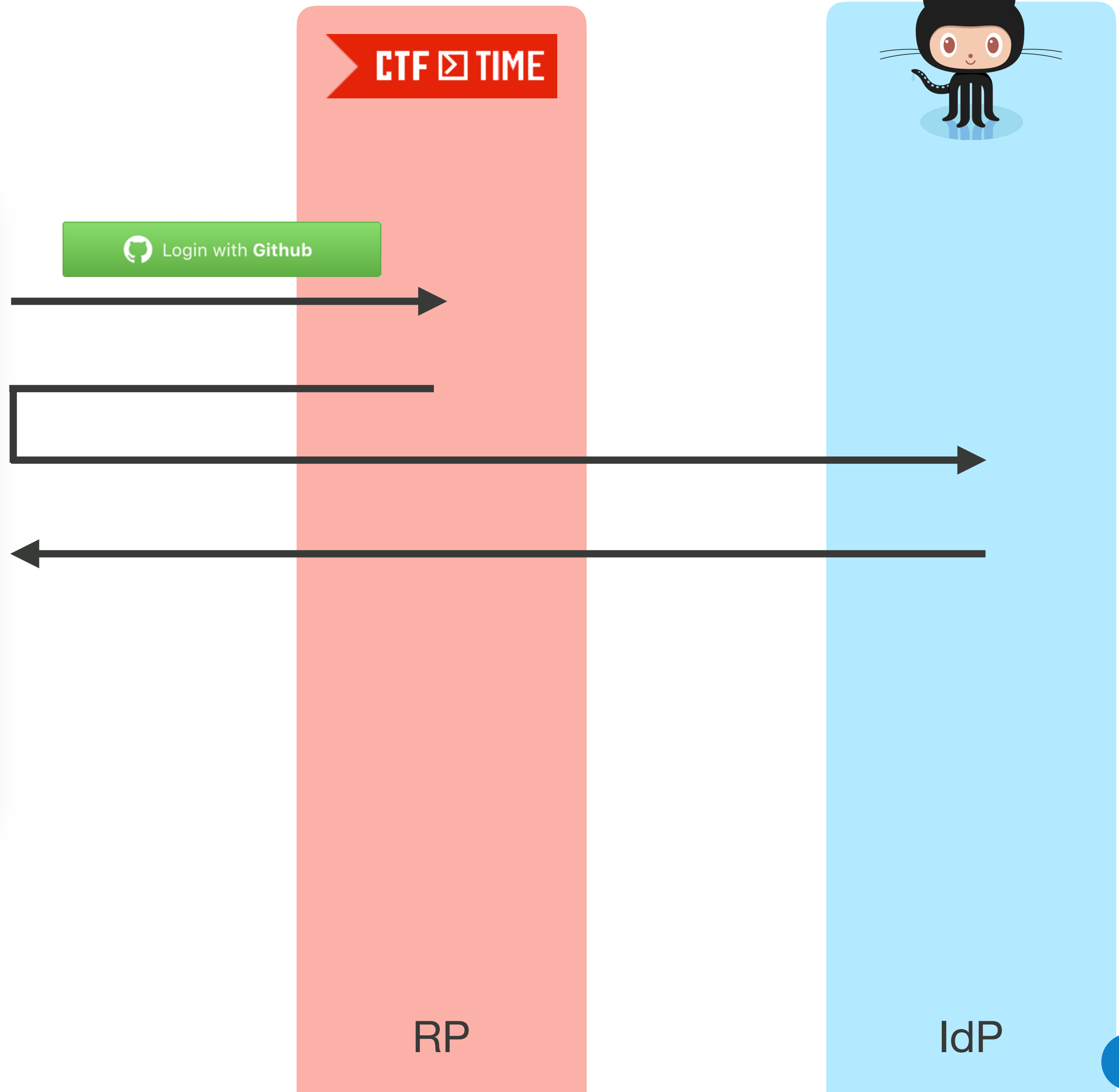
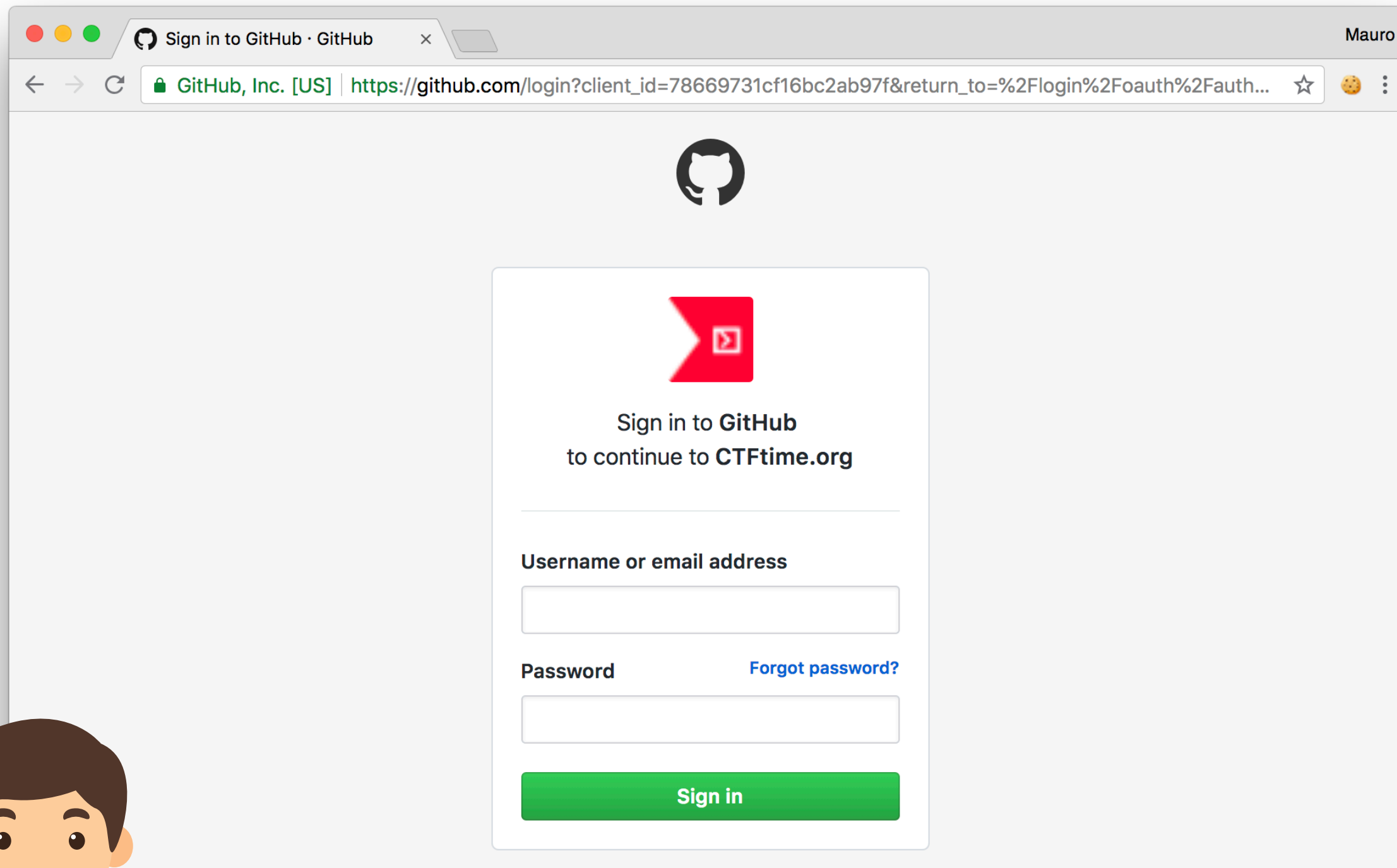
A vertical red bar with a white banner at the top containing the 'CTF TIME' logo. At the bottom of the bar, the letters 'RP' are displayed in a dark font.

A vertical blue bar with a GitHub Octocat logo at the top. At the bottom of the bar, the letters 'IdP' are displayed in a dark font.

OVERVIEW OF A WEB PROTOCOL



OVERVIEW OF A WEB PROTOCOL



OVERVIEW OF A WEB PROTOCOL

The screenshot shows the CTFtime.org website. The top navigation bar includes 'CTF TIME', 'CTFs', 'Upcoming', 'Archive', 'Calendar', 'Teams', 'FAQ', 'Contact us', 'About', 'Timezone: UTC', and 'MrStorm'. The main content is divided into three sections: 'Team rating' with a table of top teams, 'Now running' with details for 'TJCTF 2018', and 'Past events' with details for 'Real World CTF 2018 Quals'.

Place	Team	Country	Rating
1	Dragon Sector		704,084
2	Plaid Parliament of Pwning		583,211
3	dcua		510,212
4	TokyoWesterns		485,811
5	p4		461,567
6	LC&BC		435,130

Place	Team	Country	Points
1	Plaid Parliament of Pwning		49,280
2	Eat, Sleep, Pwn, Repeat		36,730



CTF TIME

Login with Github



user = MrStorm, pwd = ●●●●●●

RP

IdP

MOTIVATIONS

Designing and implementing web protocols is **HARD!**

- *Bansal et al.* - Discovering Concrete Attacks on Website Authorization by Formal Analysis (**S&P '12**)
- *Wang et al.* - Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services (**S&P '12**)
- *Sun and Beznosov* - The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems (**CCS '12**)
- *Fett et al.* - A Comprehensive Formal Security Analysis of OAuth 2.0 (**CCS '16**)
- ...

MOTIVATIONS

Designing and implementing web protocols is **HARD!**

- *Bansal et al.* - Discovering Concrete Attacks on Website Authorization by Formal Analysis (**S&P '12**)
- *Wang et al.* - Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services (**S&P '12**)
- *Sun and Beznosov* - The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems (**CCS '12**)
- *Fett et al.* - A Comprehensive Formal Security Analysis of OAuth 2.0 (**CCS '16**)
- ...

WHY?

MOTIVATIONS

Designing and implementing web protocols is **HARD!**

- *Bansal et al.* - Discovering Concrete Attacks on Website Authorization by Formal Analysis (**S&P '12**)
- *Wang et al.* - Signing Me onto Your Accounts through Facebook and Google: A Traffic-Guided Security Study of Commercially Deployed Single-Sign-On Web Services (**S&P '12**)
- *Sun and Beznosov* - The Devil is in the (Implementation) Details: An Empirical Analysis of OAuth SSO Systems (**CCS '12**)
- *Fett et al.* - A Comprehensive Formal Security Analysis of OAuth 2.0 (**CCS '16**)
- ...

The browser is not aware of the existence of web protocols and of their semantics!

OUR PROPOSAL - WPSE

Extend the browser with a lightweight security monitor that enforces the compliance of the browser behaviors with respect to the web protocol specifications

OUR PROPOSAL - WPSE

Extend the browser with a lightweight security module that enforces the compliance of the browser behaviors with respect to the web protocol specifications



Implemented as a
Google Chrome **extension**

OUR PROPOSAL - WPSE

Extend the browser with a lightweight security mechanism that enforces the compliance of the browser behaviors with respect to the web protocol specifications

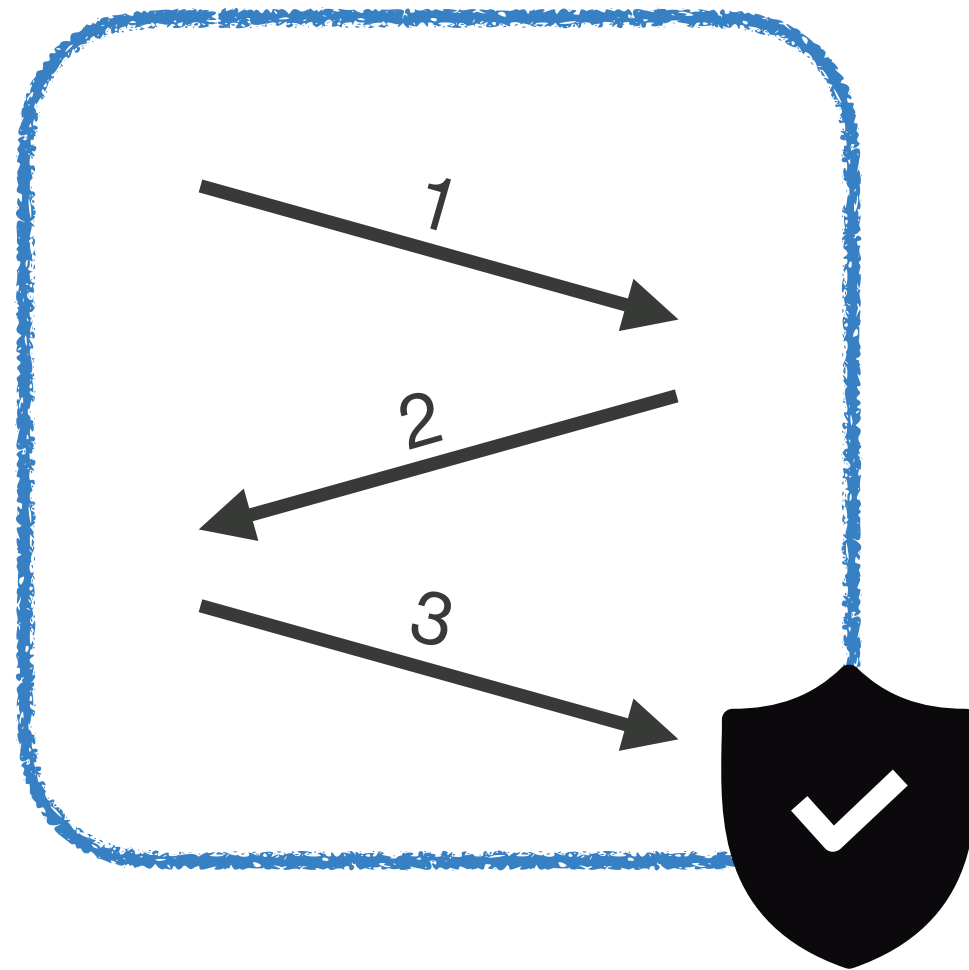


Implemented as a
Google Chrome **extension**

Advantages:

1. users of vulnerable websites are **automatically protected** against a large class of attacks
2. specifications can be written **once** and enforced on **several** sites

SECURITY CHALLENGES IN WEB PROTOCOLS



Compliance with the protocol flow

- Preserve the intended sequence of messages exchanged by honest participants
- Perform integrity checks on the contents of protocol messages



Secrecy of message components

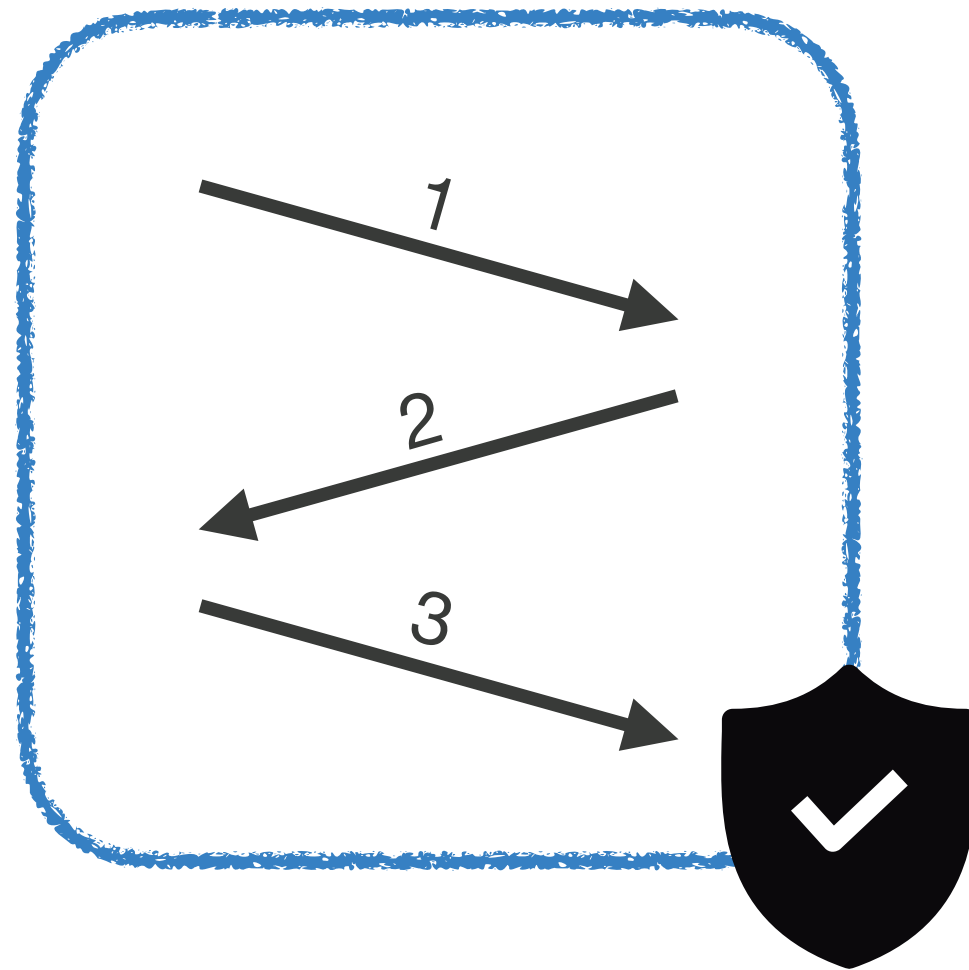
- Enforce the confidentiality of protocol secrets like tokens and credentials

TACKLING THE CHALLENGES IN WPSE

WPSE protocol specification:

- Structure and order of messages
- Desired security policies (confidentiality and integrity)

TACKLING THE CHALLENGES IN WPSE

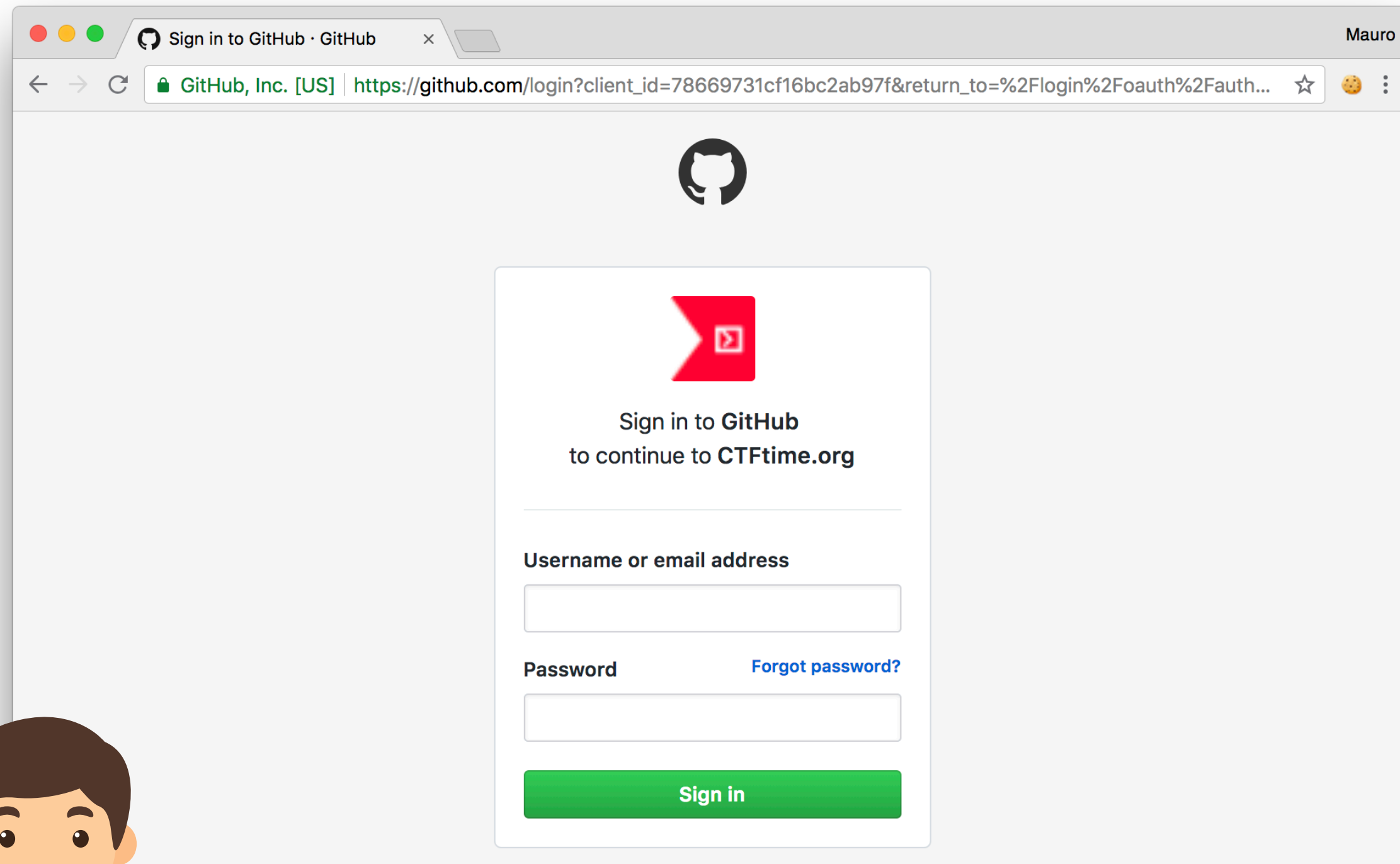


- Protocol messages are **blocked** if
 - not in the correct order
 - integrity constraints on messages are not satisfied
- Always **allow** protocol unrelated messages

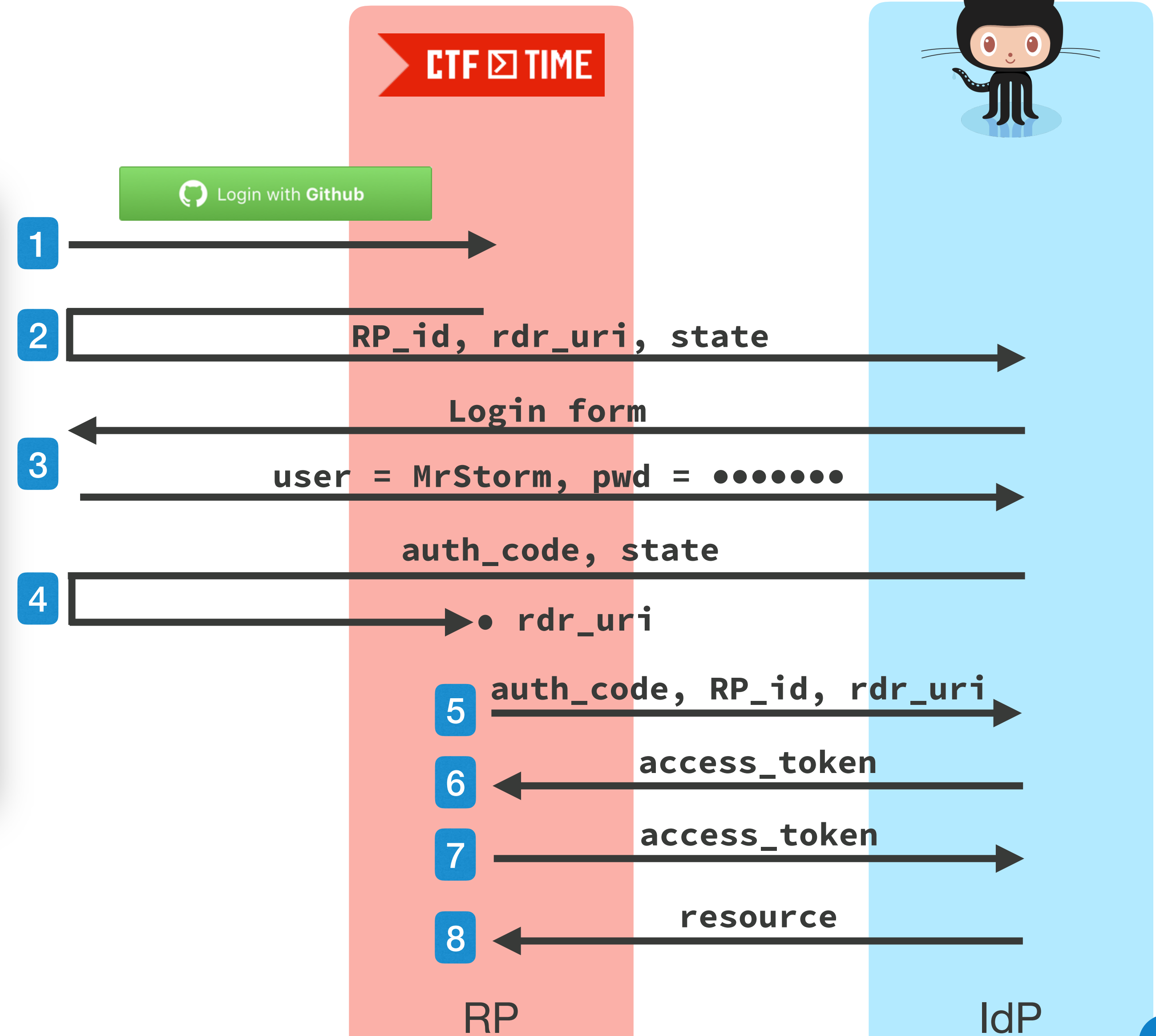


- Secrets in incoming messages are substituted with **random placeholders** before they enter the DOM
- Placeholders in outgoing requests are replaced with secrets **only** if sent to origins entitled to learn them

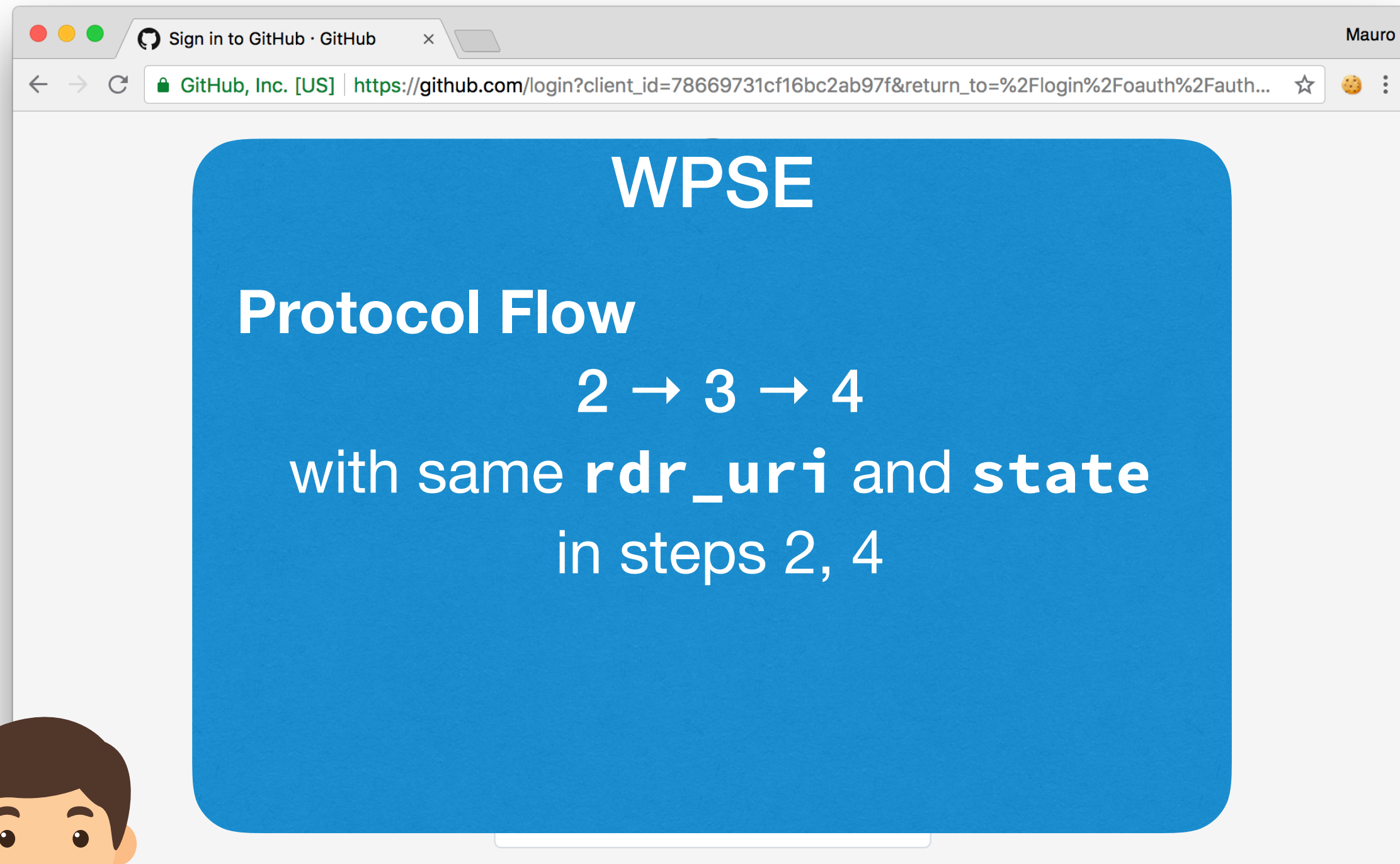
FORTIFYING OAUTH 2.0



U



FORTIFYING OAUTH 2.0



Sign in to GitHub · GitHub

GitHub, Inc. [US] | https://github.com/login?client_id=78669731cf16bc2ab97f&return_to=%2Flogin%2Foauth%2Fauth...

WPSE

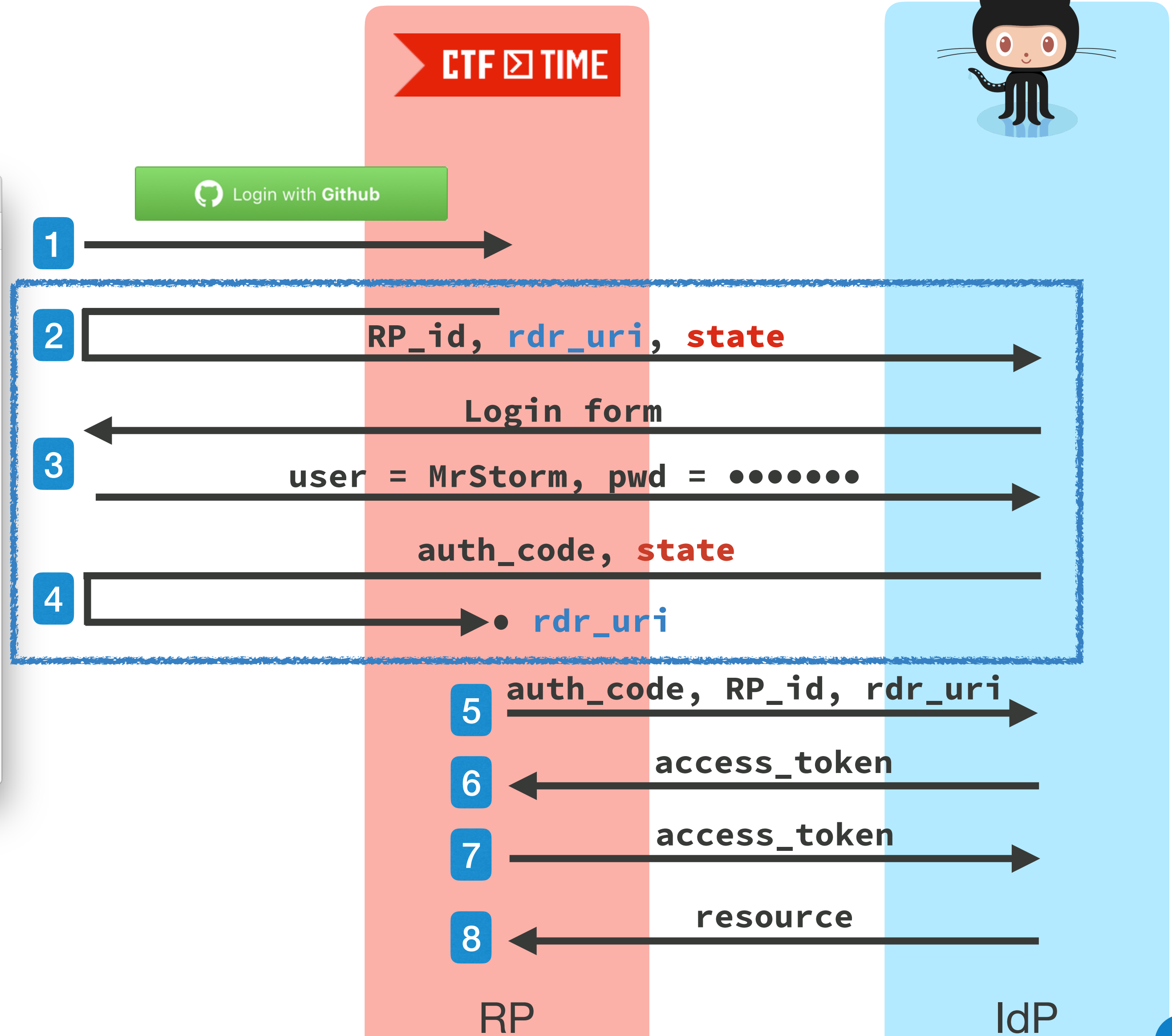
Protocol Flow

2 → 3 → 4

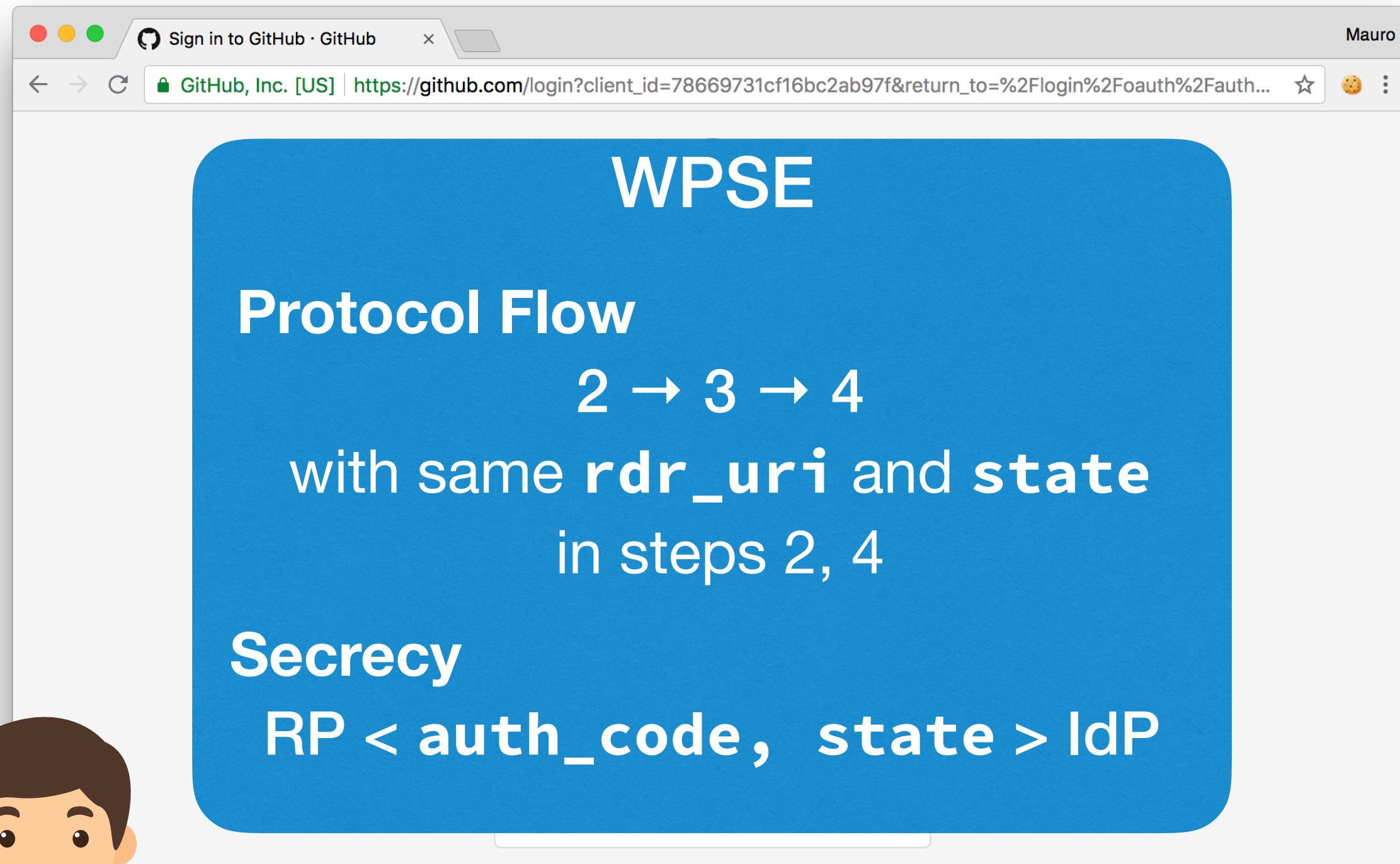
with same `rdr_uri` and `state` in steps 2, 4



U



FORTIFYING OAUTH 2.0



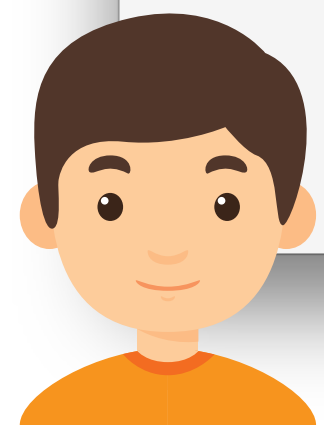
Sign in to GitHub · GitHub

GitHub, Inc. [US] | https://github.com/login?client_id=78669731cf16bc2ab97f&return_to=%2Flogin%2Foauth%2Fauth...

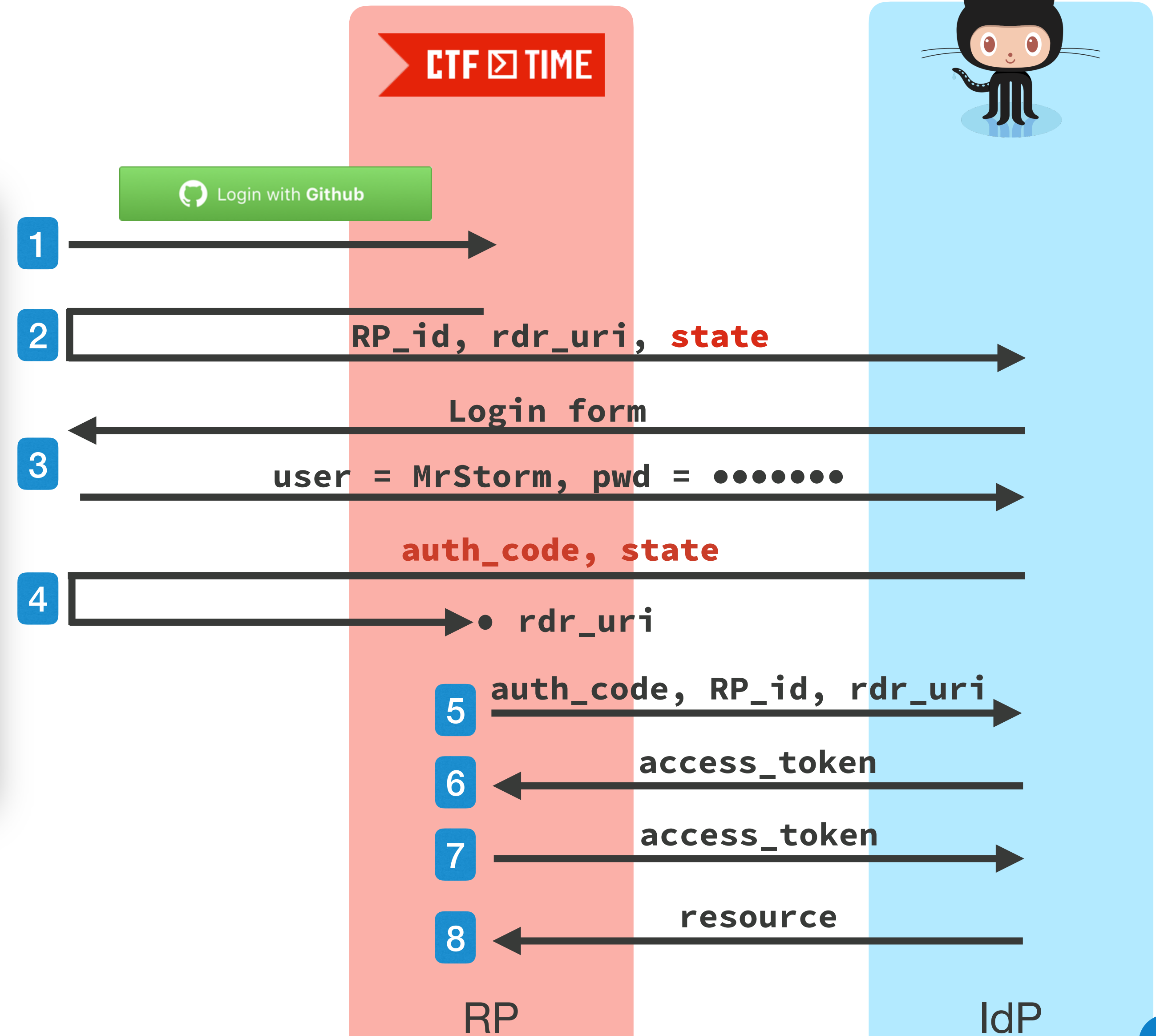
WPSE

Protocol Flow
2 → 3 → 4
with same `rdr_uri` and `state`
in steps 2, 4

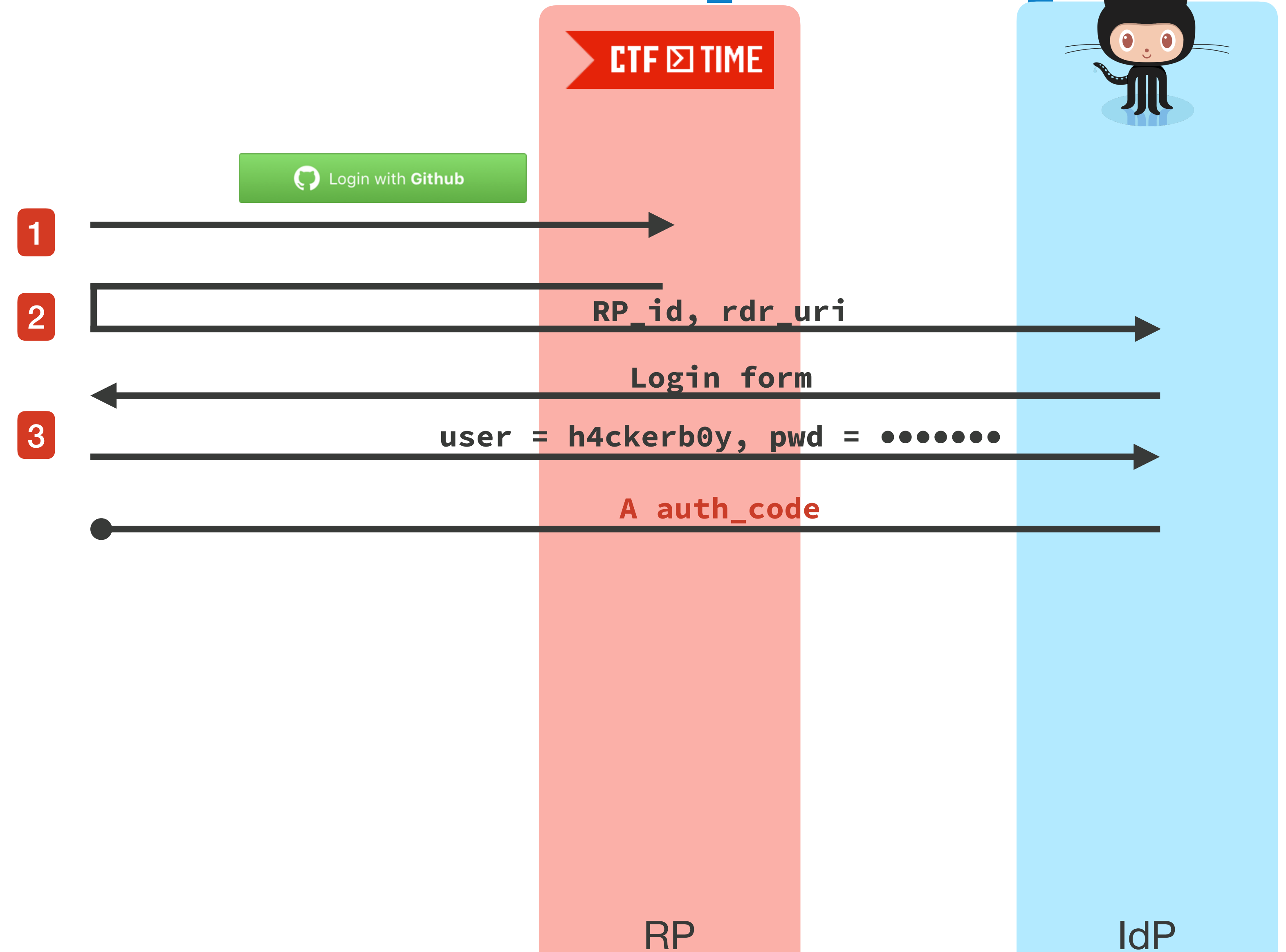
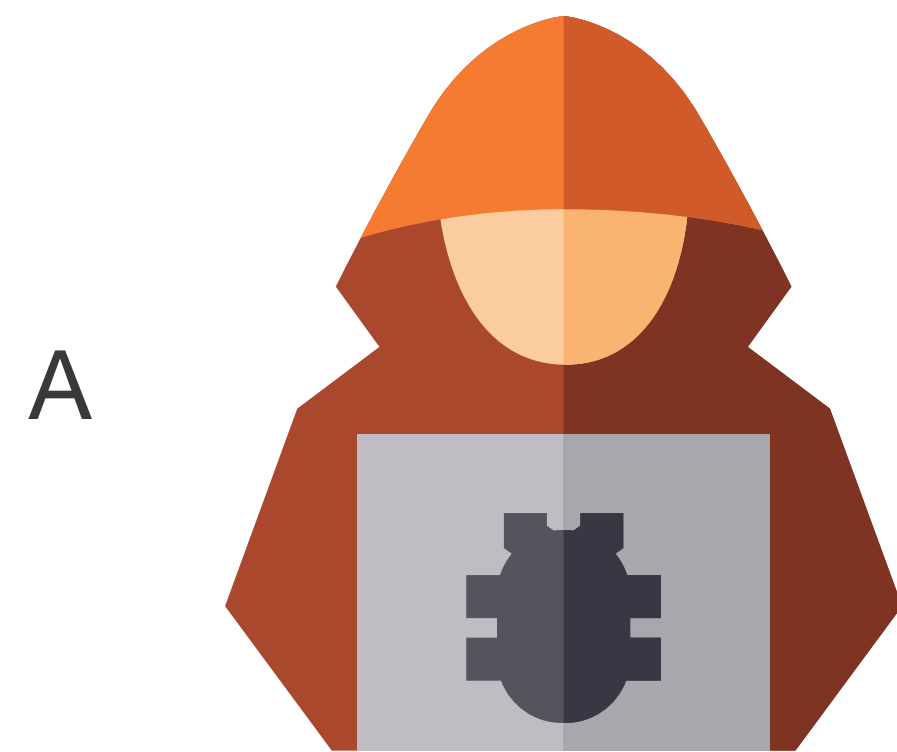
Secrecy
RP < `auth_code`, `state` > IdP



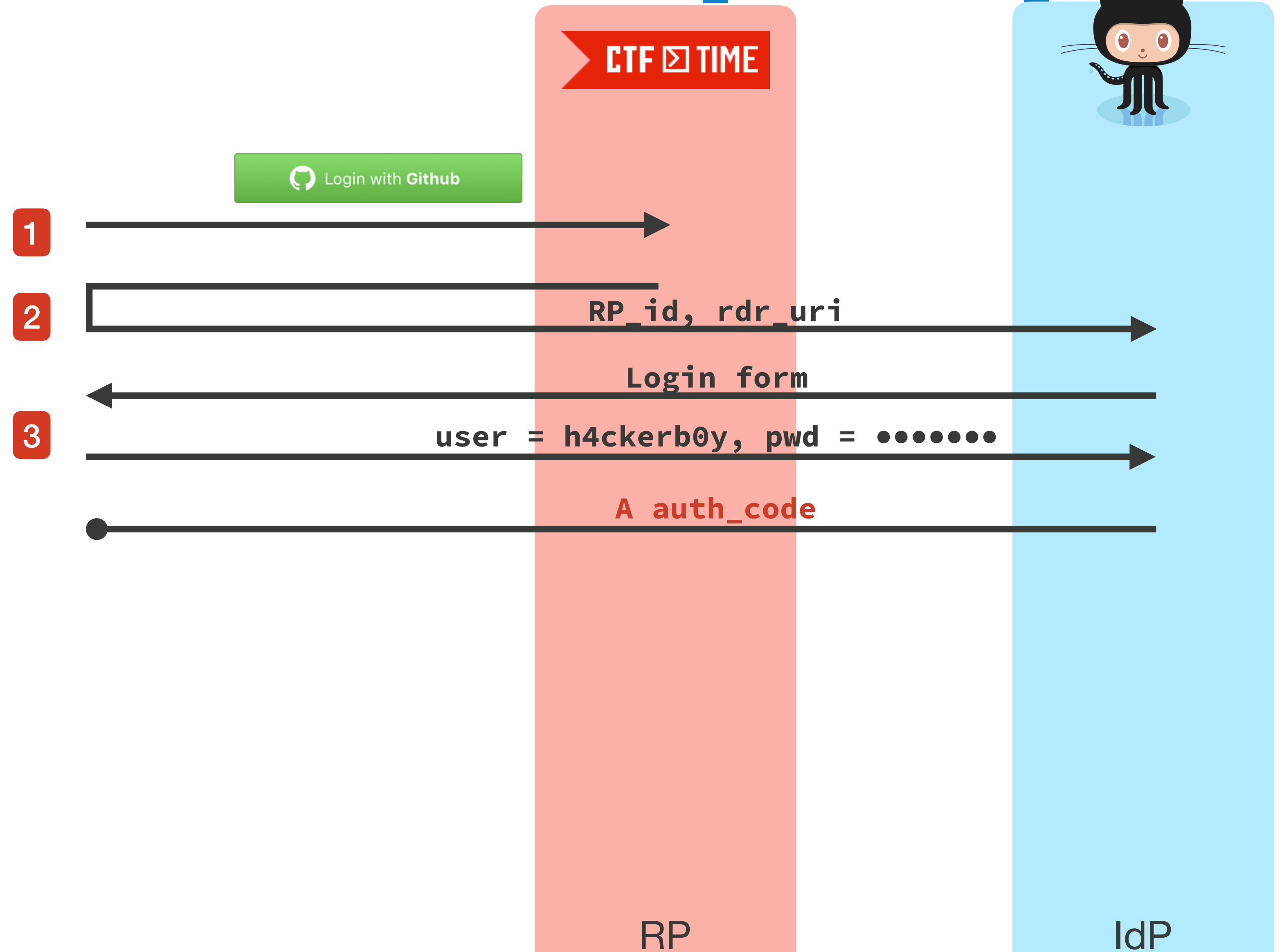
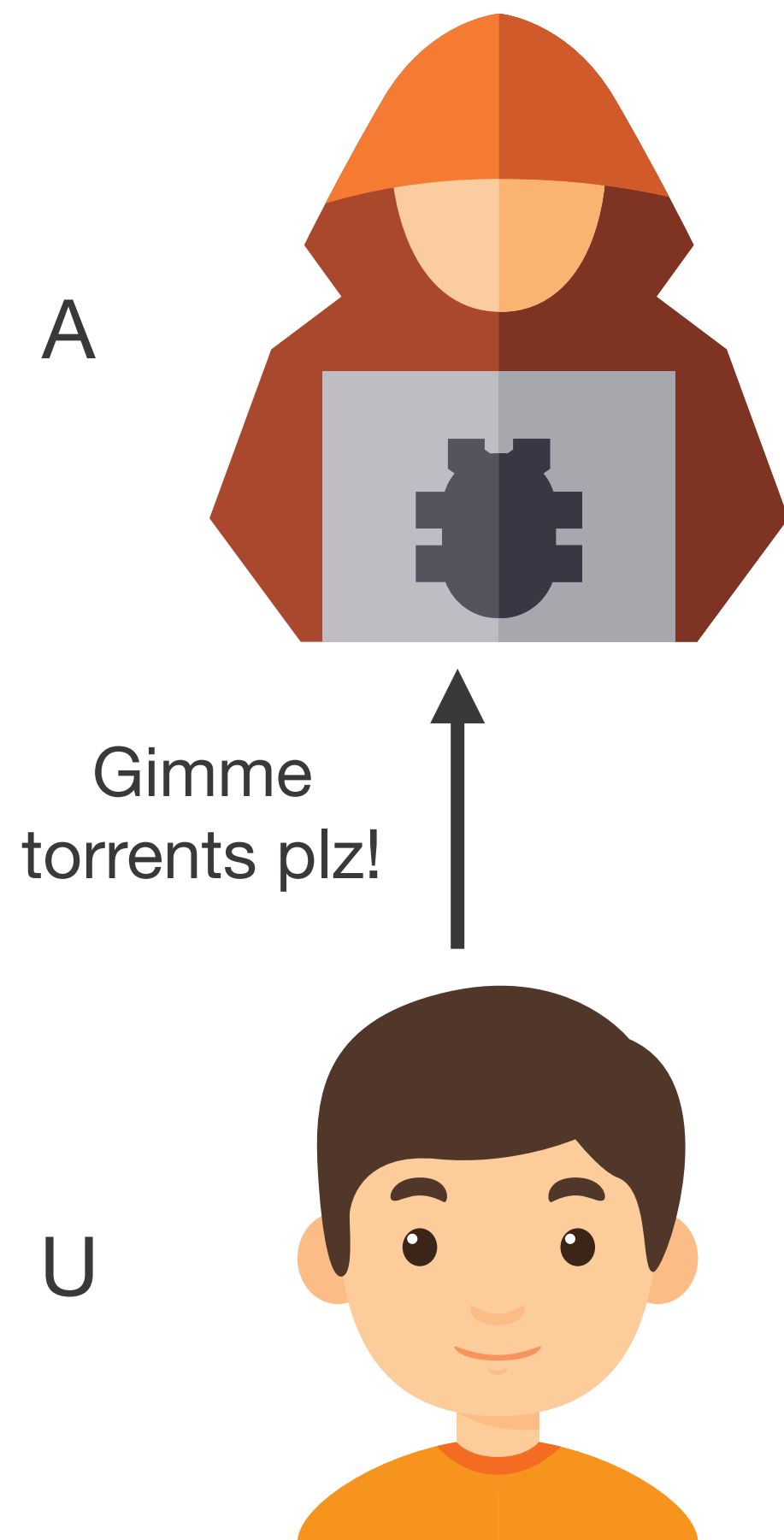
U



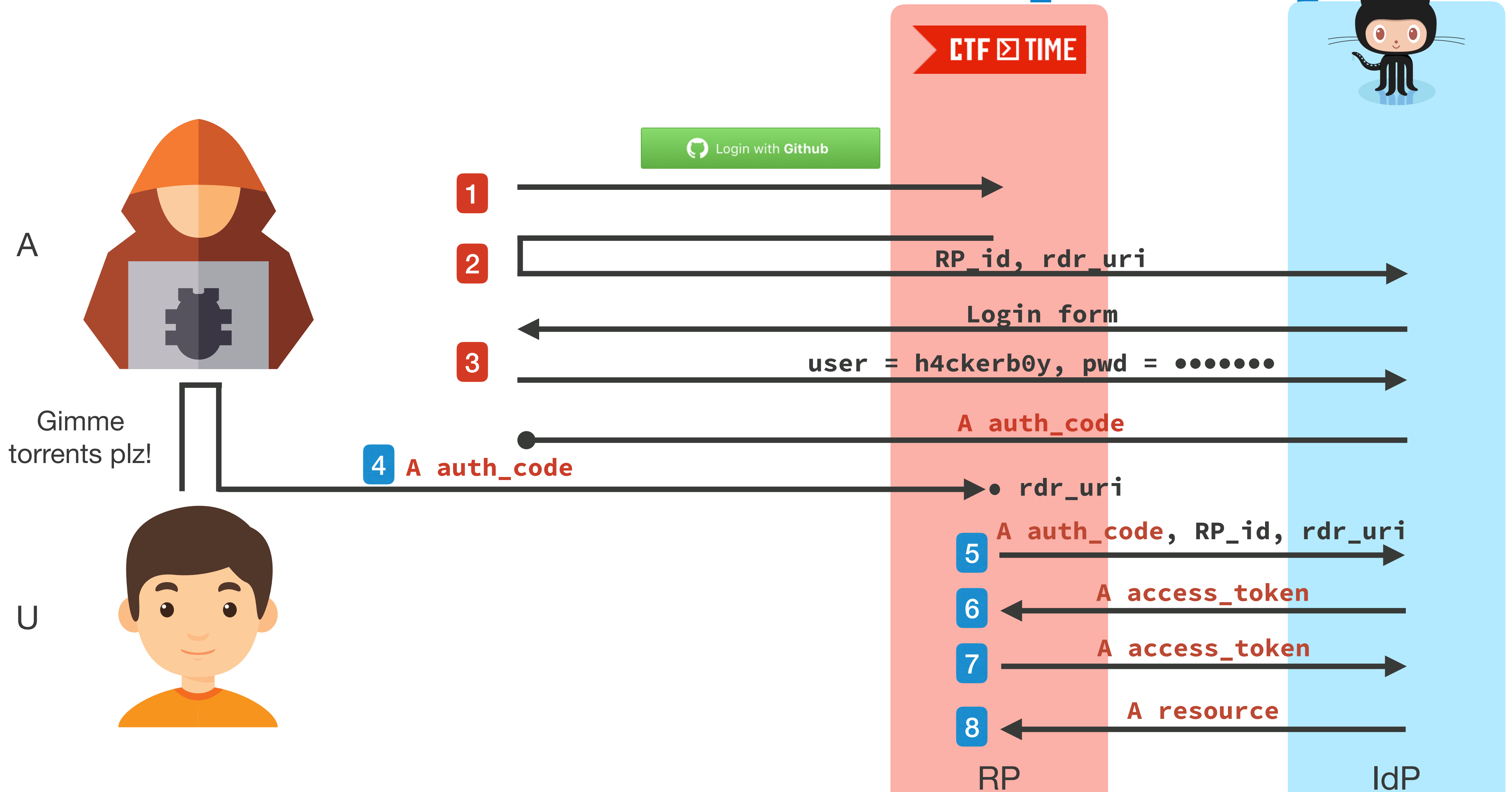
SESSION SWAPPING [SB12]



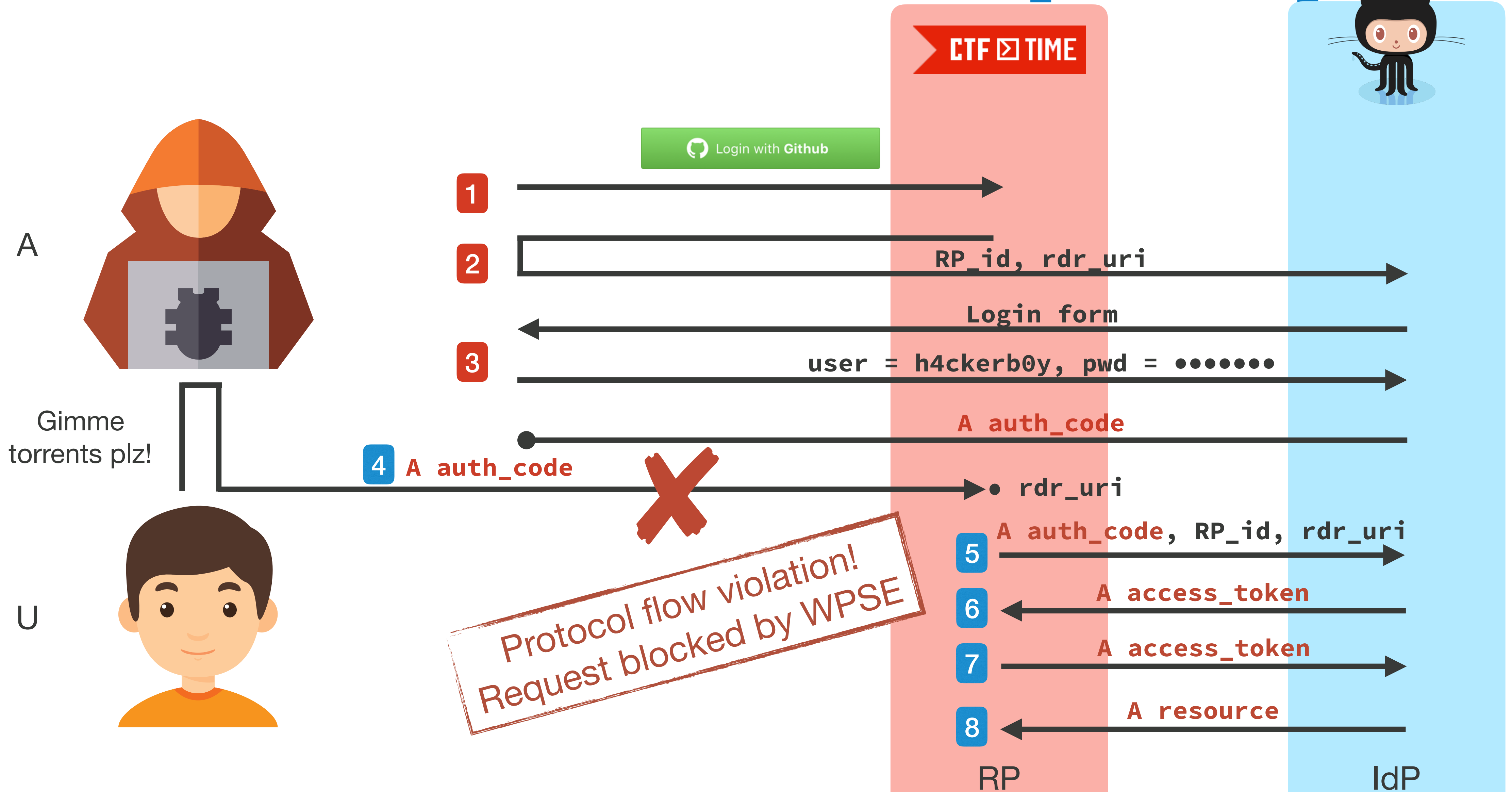
SESSION SWAPPING [SB12]



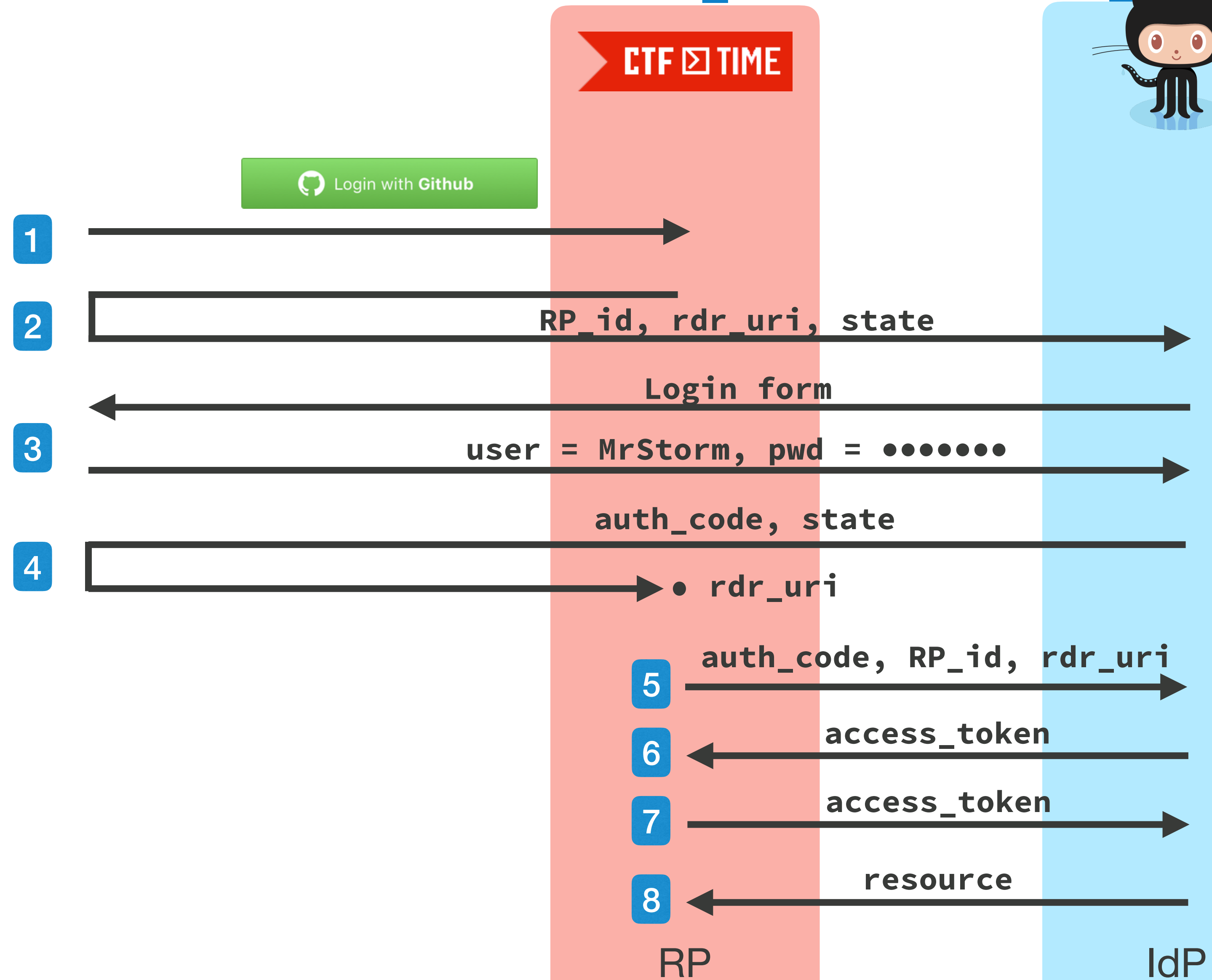
SESSION SWAPPING [SB12]



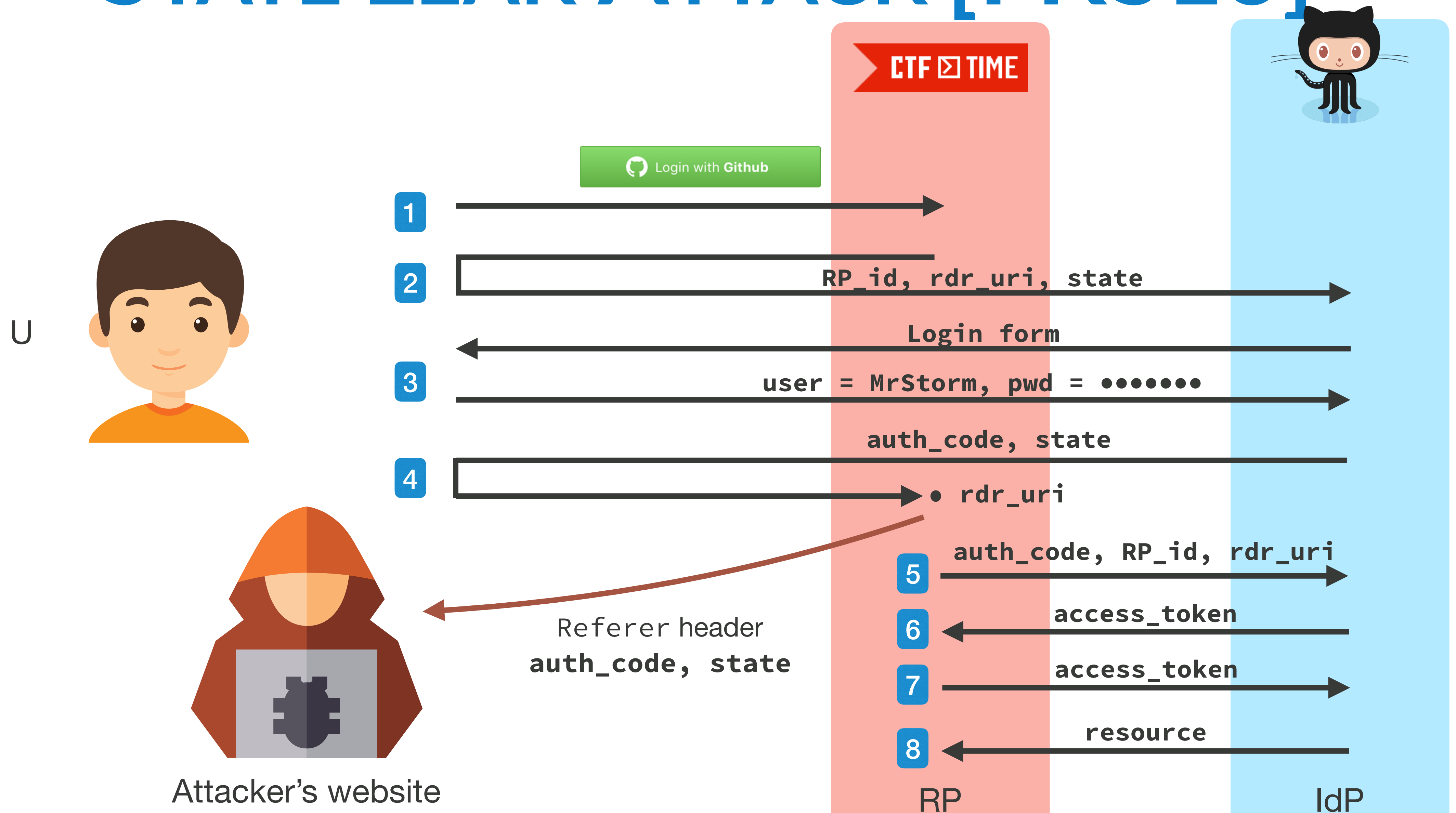
SESSION SWAPPING [SB12]



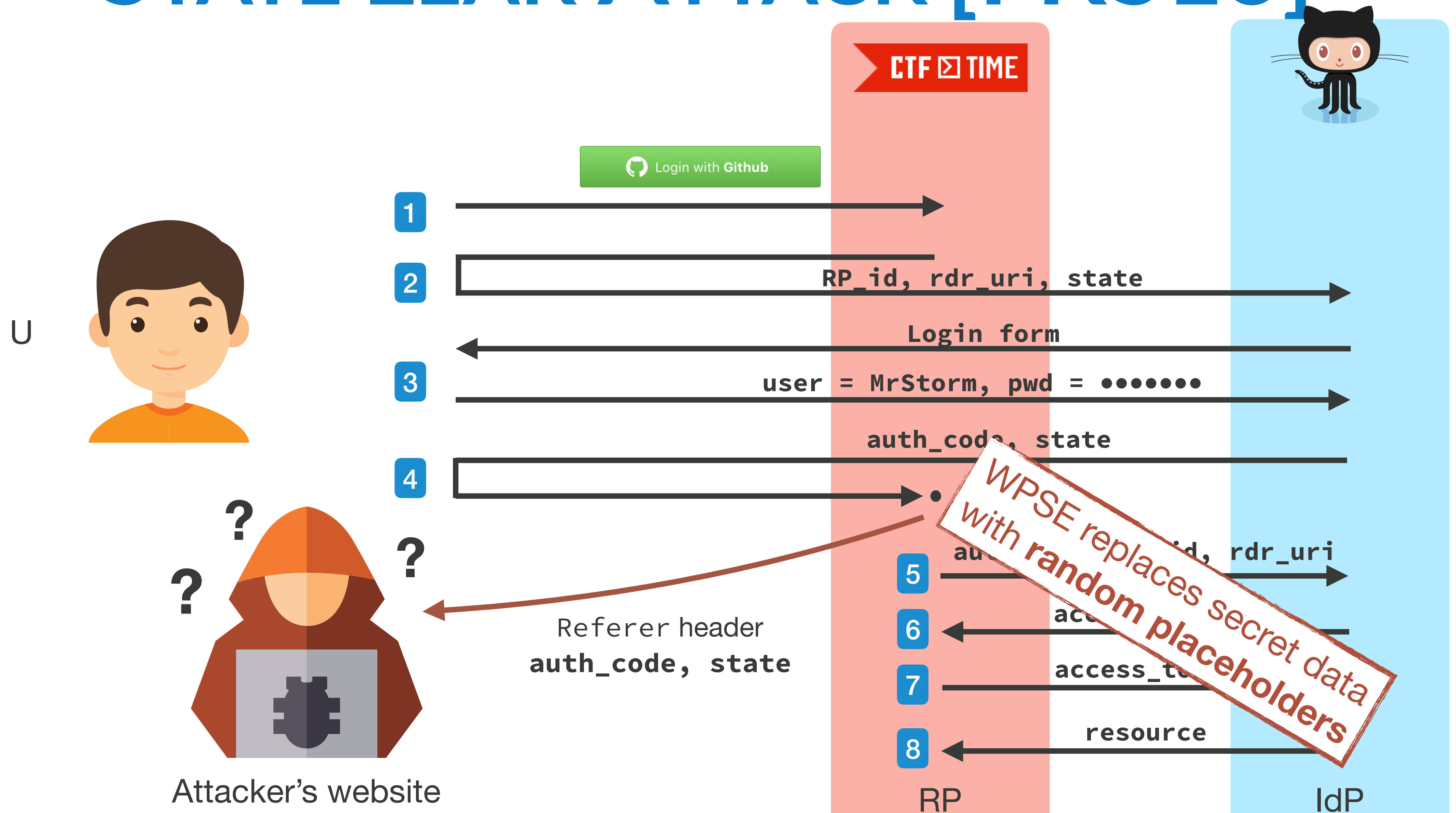
STATE LEAK ATTACK [FKS16]



STATE LEAK ATTACK [FKS16]



STATE LEAK ATTACK [FKS16]

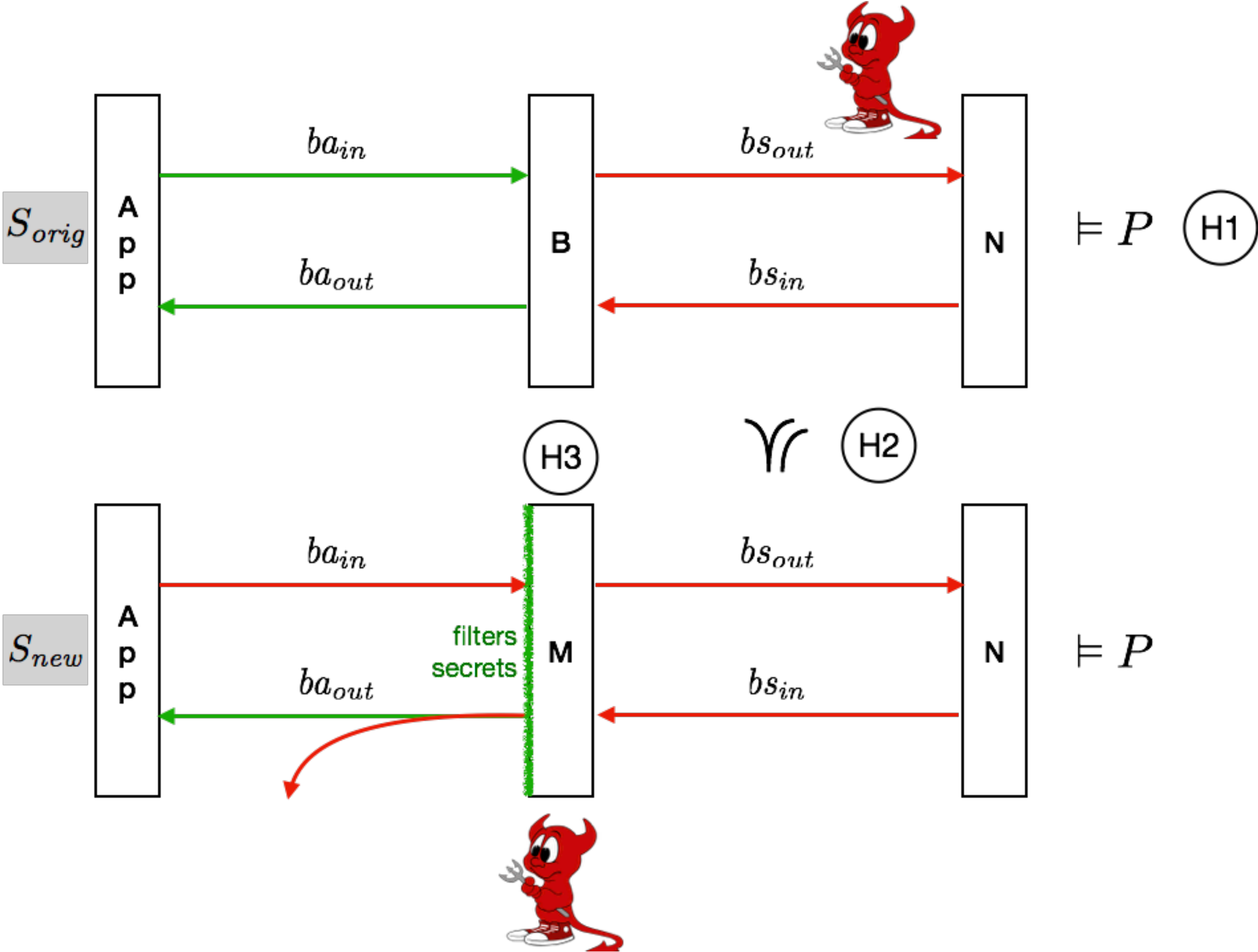


FORMAL RESULTS

(H1) The protocol fulfills safety property P with a benign webpage

(H2) WPSE allows only a subset of the I/O sequences performed by the browser in a honest protocol run

(H3) Secrets are not leaked and securely stored by the browser

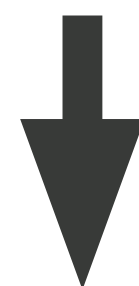


FORMAL RESULTS

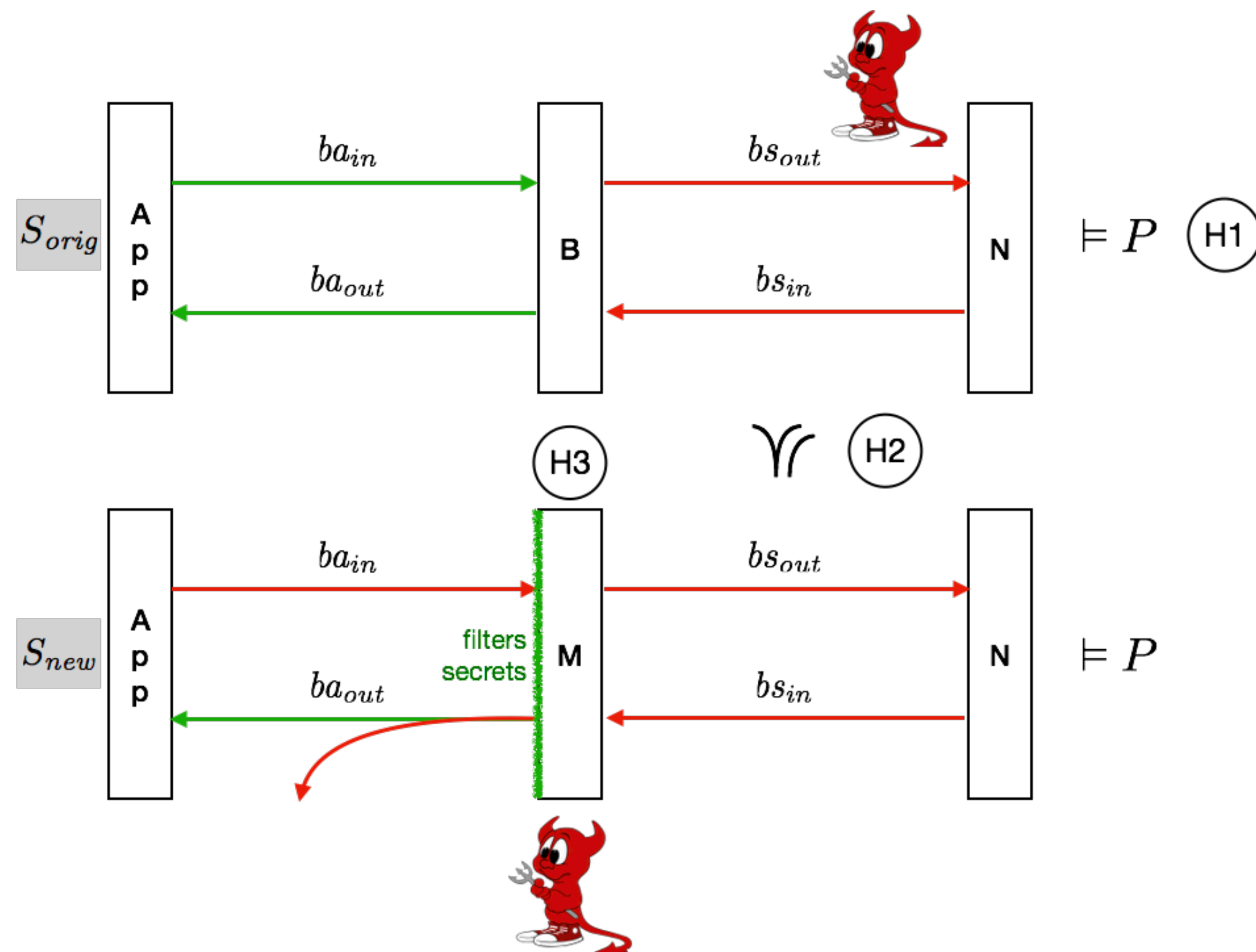
(H1) The protocol fulfills safety property P with a benign webpage

(H2) WPSE allows only a subset of the I/O sequences performed by the browser in a honest protocol run

(H3) Secrets are not leaked and securely stored by the browser



The protocol fulfills P with a **compromised** browser monitored by WPSE



EXPERIMENTAL EVALUATION



- Manual investigation of 30 RPs for each IdP from Alexa top 100K
- Analyzed both **authorization code mode** and **implicit mode** of OAuth 2.0

Security

- Leakage of sensitive data due to **advertisement libraries** (4 RPs)
- Lack or misuse of the **state parameter** (55 RPs)

Compatibility

Problems due to **security critical deviations** in the **protocol flow** (7 RPs), e.g. auth code is sent twice, second time over HTTP

A NEW ATTACK AGAINST GOOGLE IMPLEMENTATION OF SAML 2.0

- Similar to the **session swapping** attack presented before
- **Login CSRF** against Google Suite applications (Google Drive, GMail, ...)

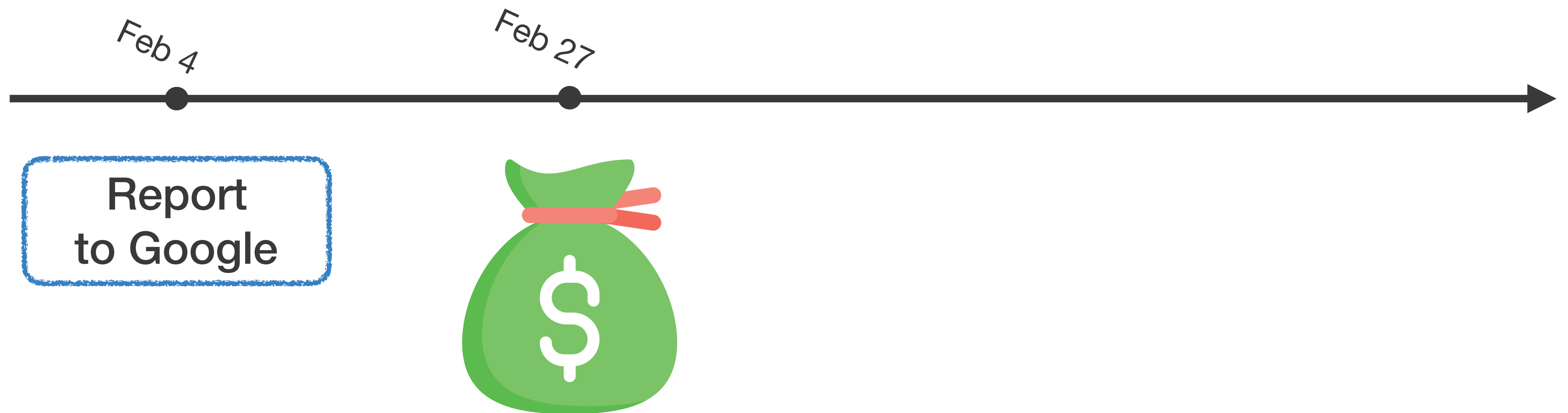
A NEW ATTACK AGAINST GOOGLE IMPLEMENTATION OF SAML 2.0

- Similar to the **session swapping** attack presented before
- **Login CSRF** against Google Suite applications (Google Drive, GMail, ...)



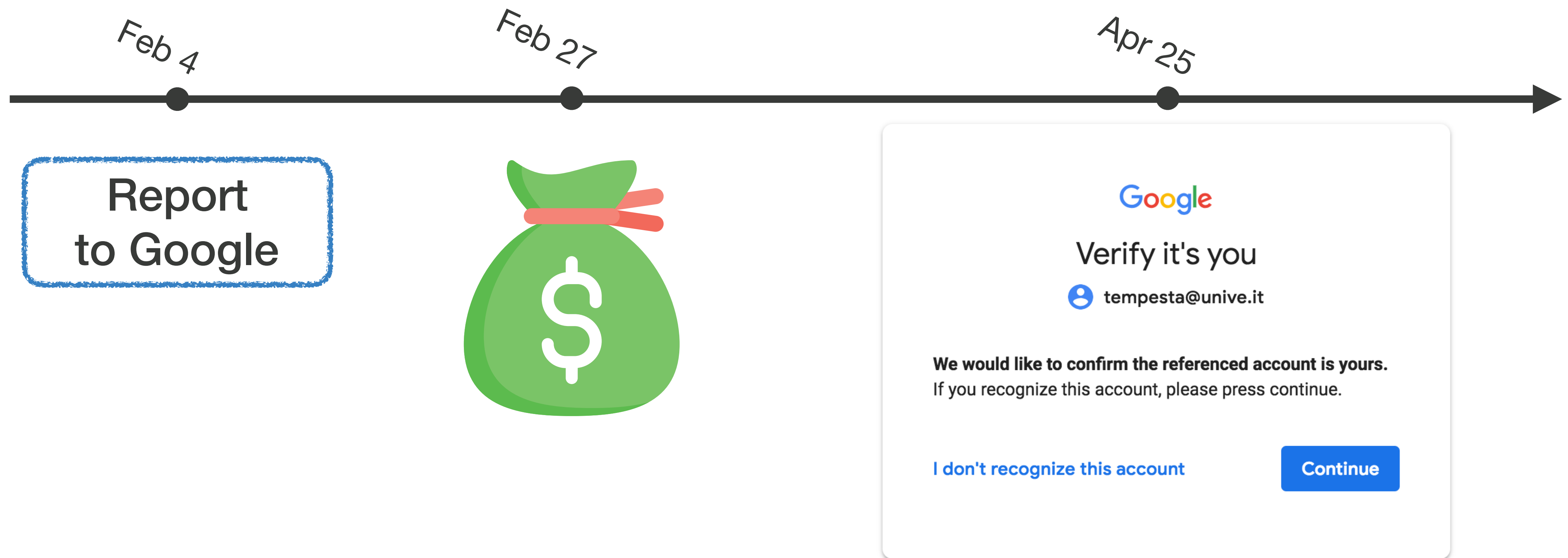
A NEW ATTACK AGAINST GOOGLE IMPLEMENTATION OF SAML 2.0

- Similar to the **session swapping** attack presented before
- **Login CSRF** against Google Suite applications (Google Drive, GMail, ...)



A NEW ATTACK AGAINST GOOGLE IMPLEMENTATION OF SAML 2.0

- Similar to the **session swapping** attack presented before
- **Login CSRF** against Google Suite applications (Google Drive, GMail, ...)



SUMMING UP

Lightweight policies on the client-side suffice to enforce provable security guarantees in web protocols

SUMMING UP

Lightweight policies on the client-side suffice to enforce provable security guarantees in web protocols



- Support for additional protocols e.g., **e-payments**
- **Automatic** techniques to **synthesize WPSE policies** from protocol specifications / browser traffic
- Embed WPSE into real **browsers**

THANK YOU!

QUESTIONS?



tempesta@unive.it



<https://sites.google.com/site/wpseproject/>