Vetting Single Sign-On SDK Implementations via Symbolic Reasoning

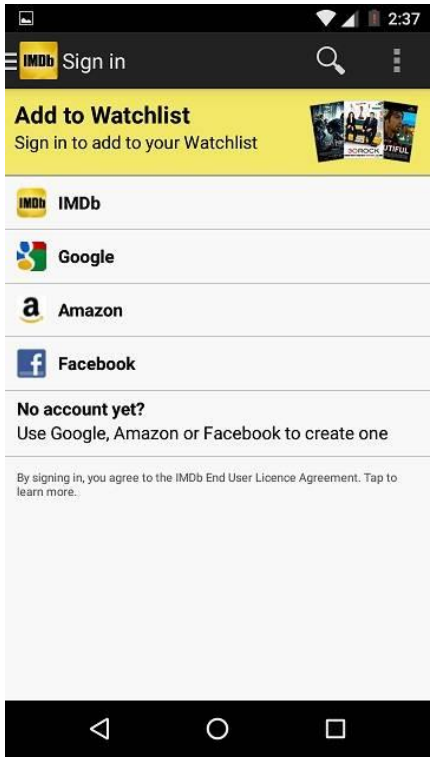
Ronghai Yang[1,2], Wing Cheong Lau[1], Jiongyi Chen[1], Kehuan Zhang[1]

[1]The Chinese University of Hong Kong and [2]Sangfor Technologies Inc.

香港中文大學
The Chinese University of Hong Kong

SANGFOR

Strong Adoption of Single Sign-On Services

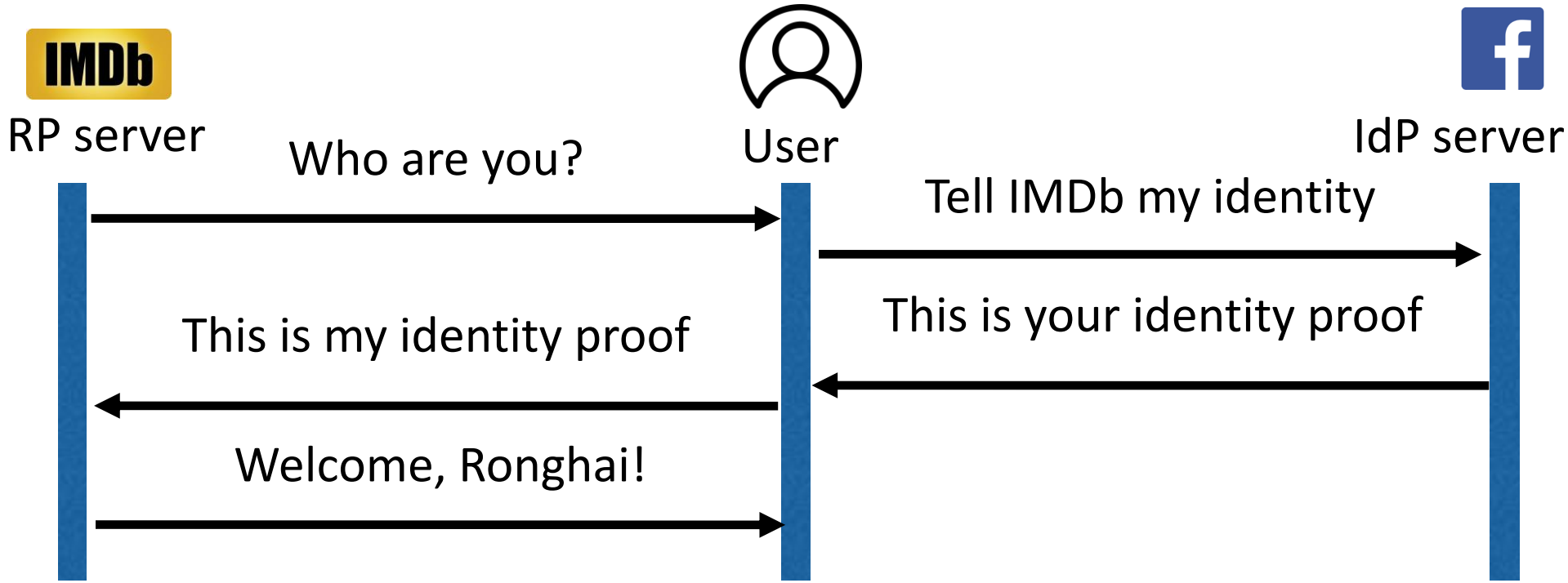


- Janrain Report: 75% of users prefer SSO and 91% of them are satisfied
- 405 out of Top-1000 web applications support SSO services [1]
- 1372 out of 4151 Android apps support SSO services [2]

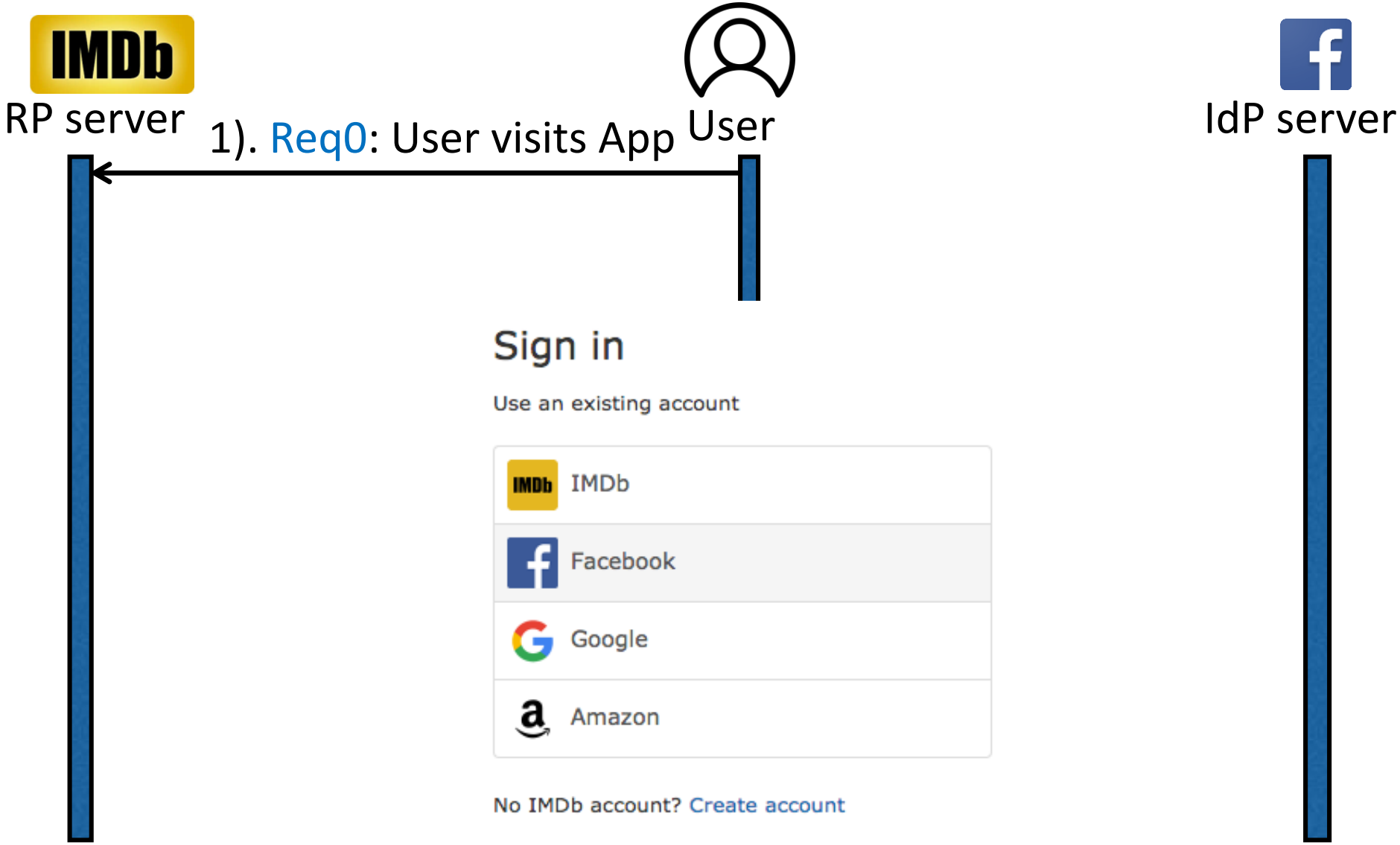
[1] Yang, R., Li, G., Lau, W. C., Zhang, K., and Hu, P. Model-based security testing: An empirical study on OAuth2.0 implementations. In AsiaCCS (2016)
[2] Wang, H., Zhang, Y., Li, J., Liu, H., Yang, W., Li, B., and Gu, D. Vulnerability assessment of OAuth implementations in Android applications. In ACSAC (2015)

Basic Interactions of Single Sign-On (SSO)

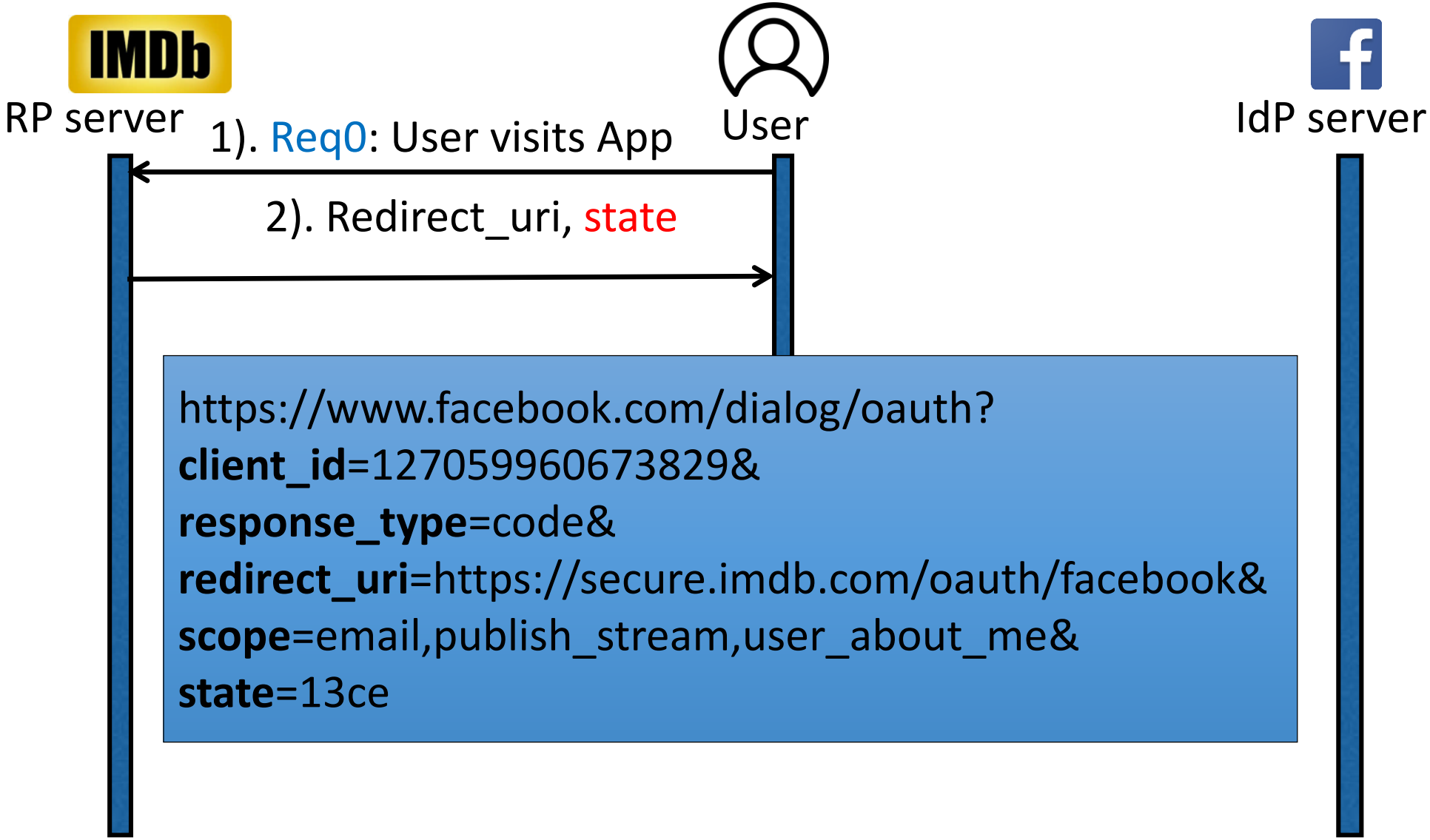
- Three Parties: the third-party application (Relying Party, RP), the Identity Provider (IdP) and the client device



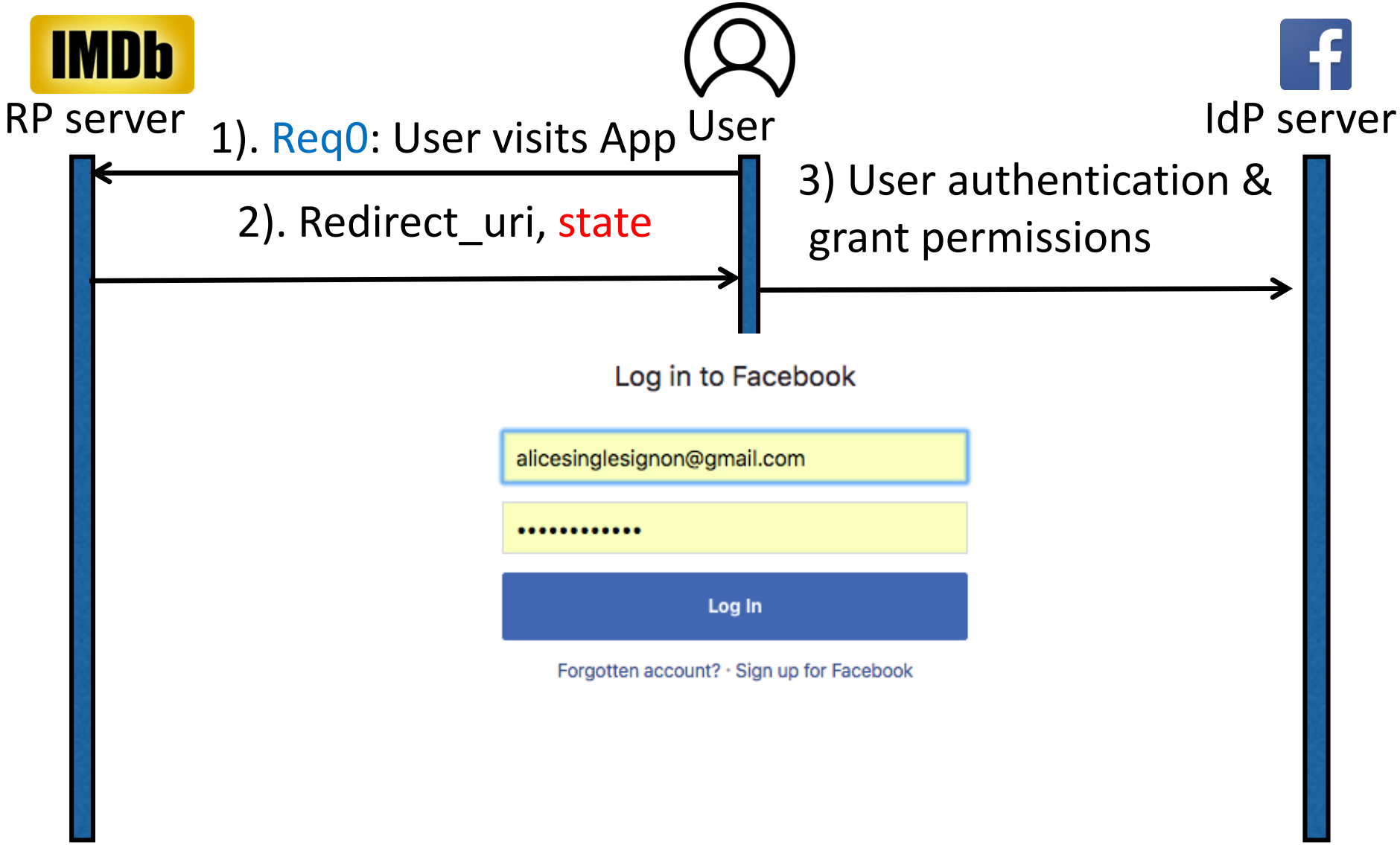
Single Sign-On Protocol Flow



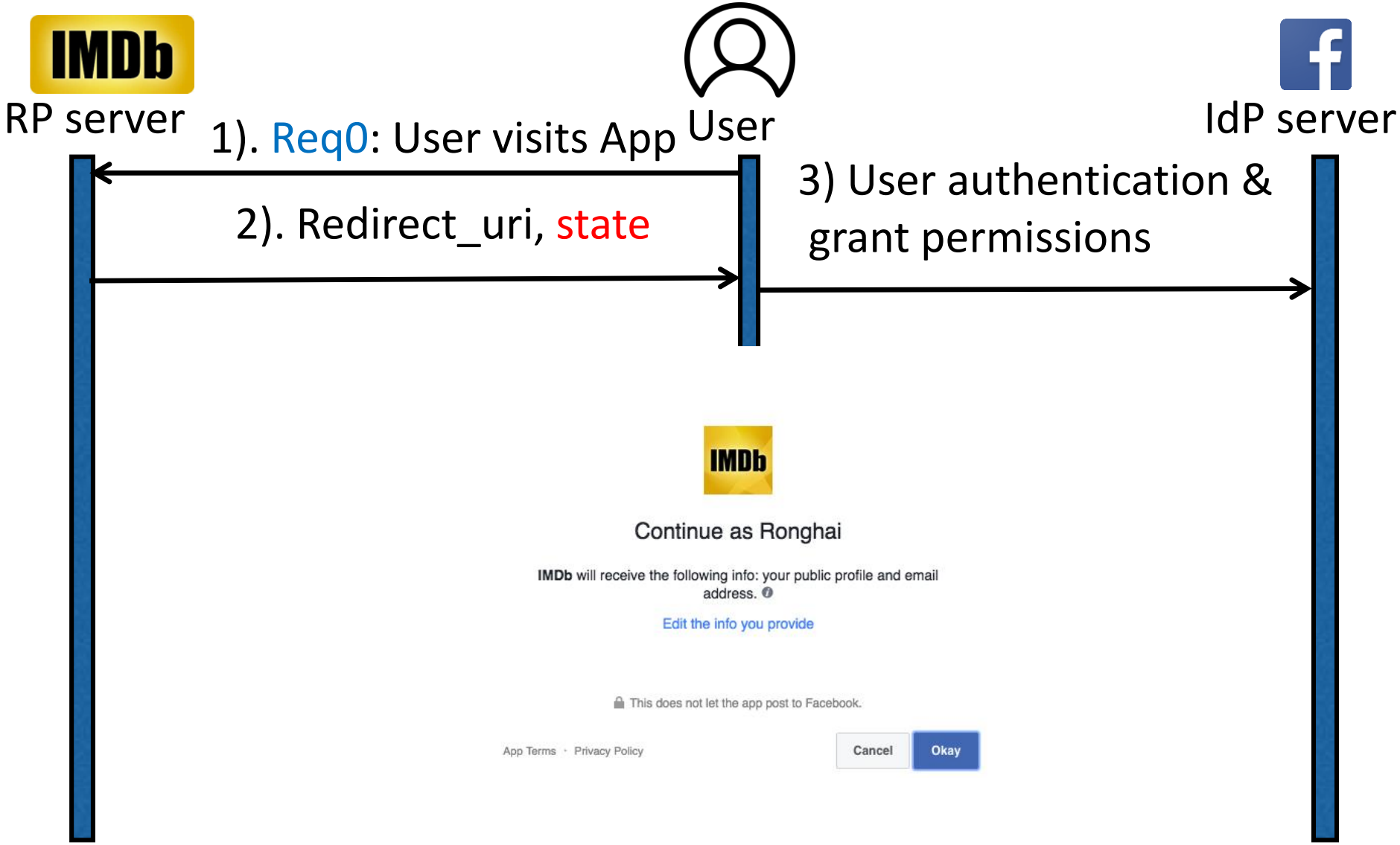
Single Sign-On Protocol Flow



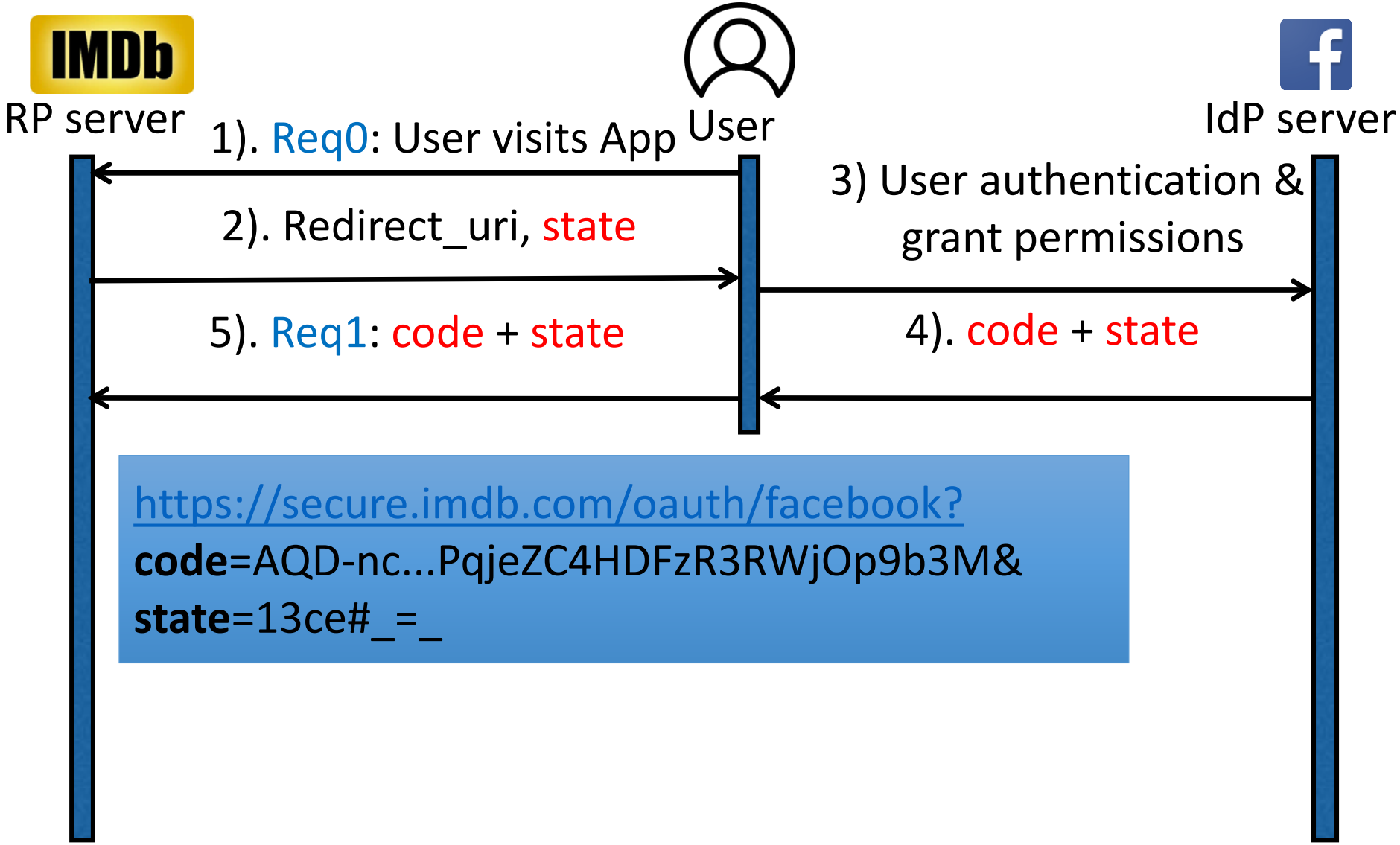
Single Sign-On Protocol Flow



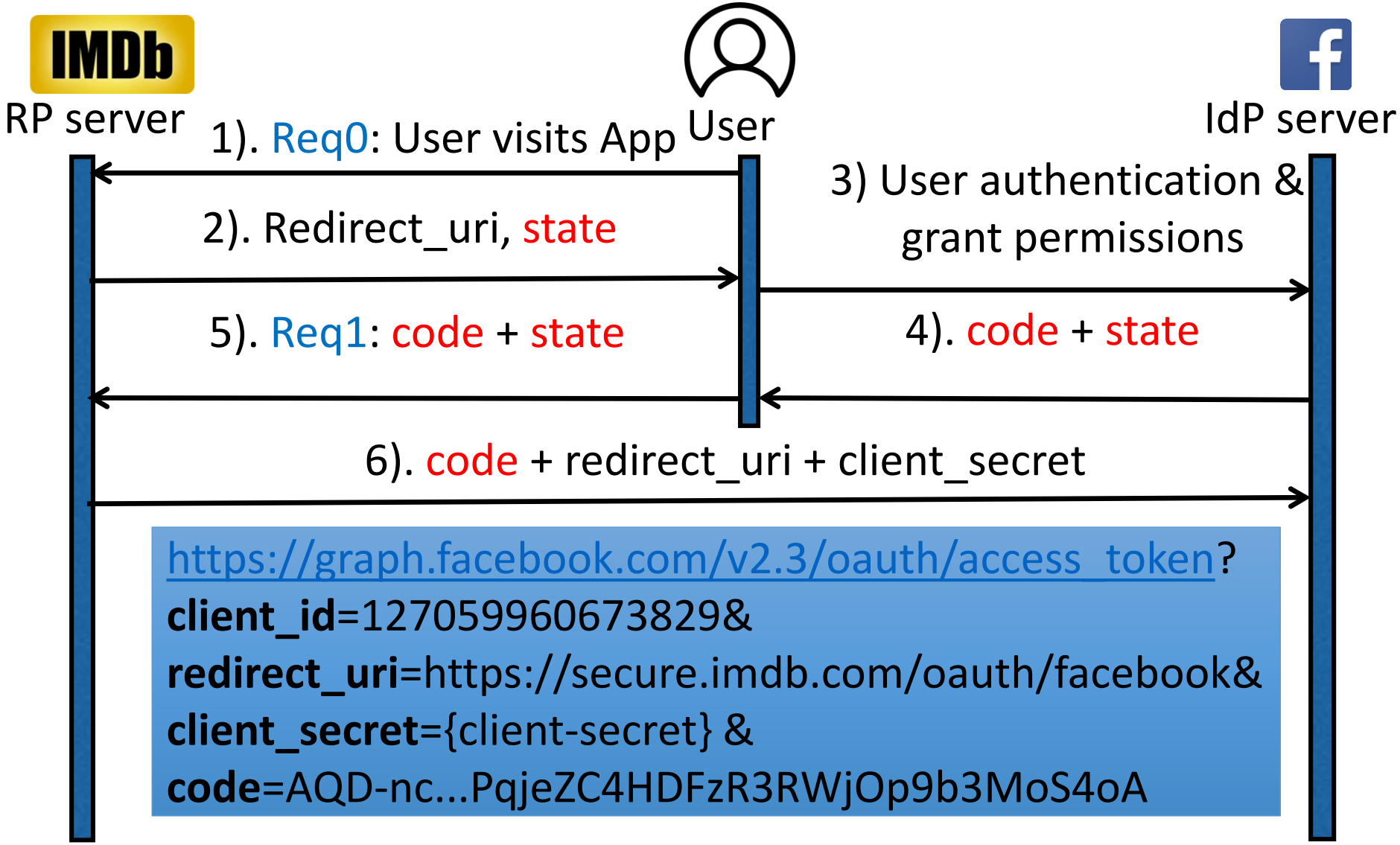
Single Sign-On Protocol Flow



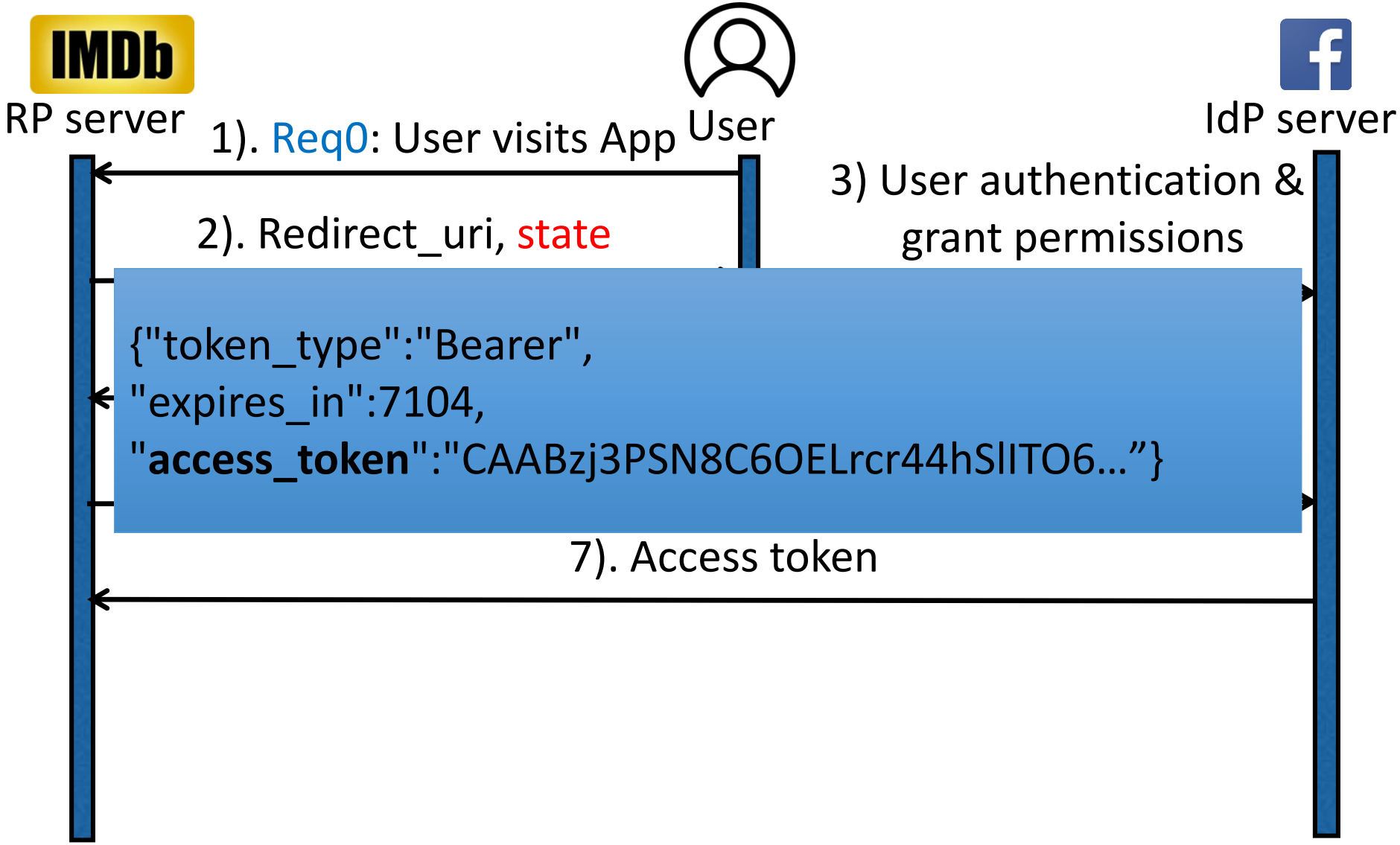
Single Sign-On Protocol Flow



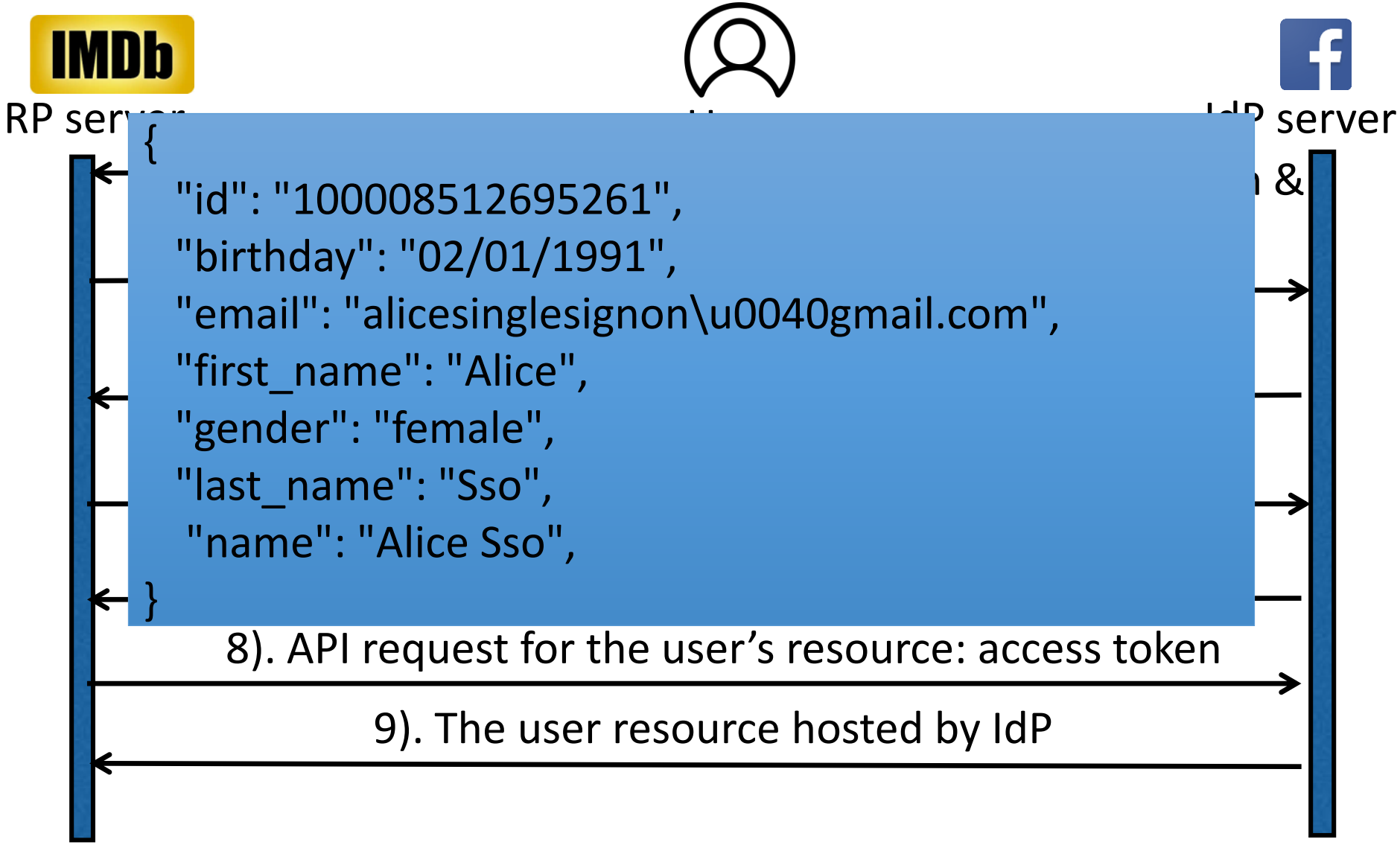
Single Sign-On Protocol Flow



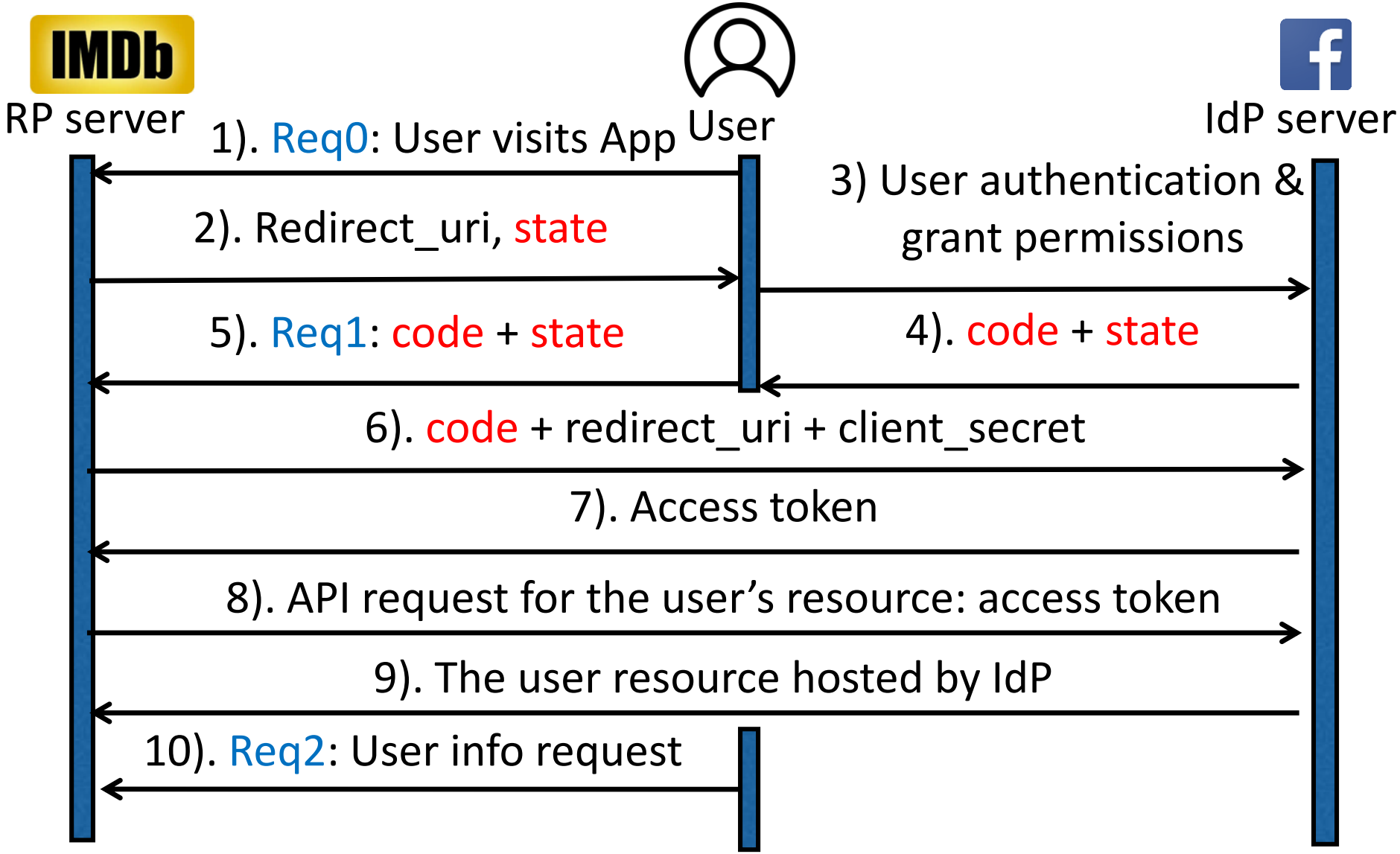
Single Sign-On Protocol Flow



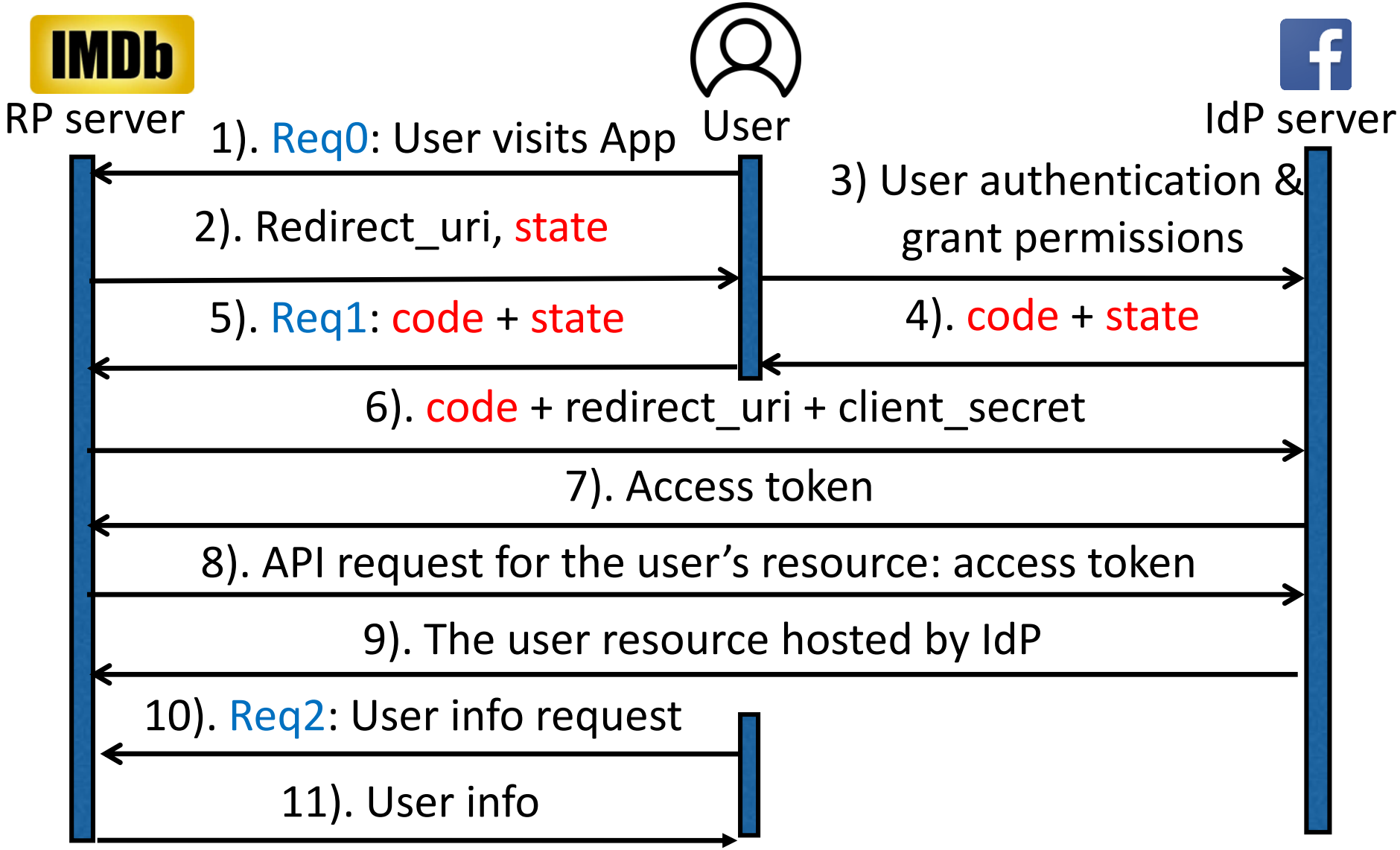
Single Sign-On Protocol Flow



Single Sign-On Protocol Flow

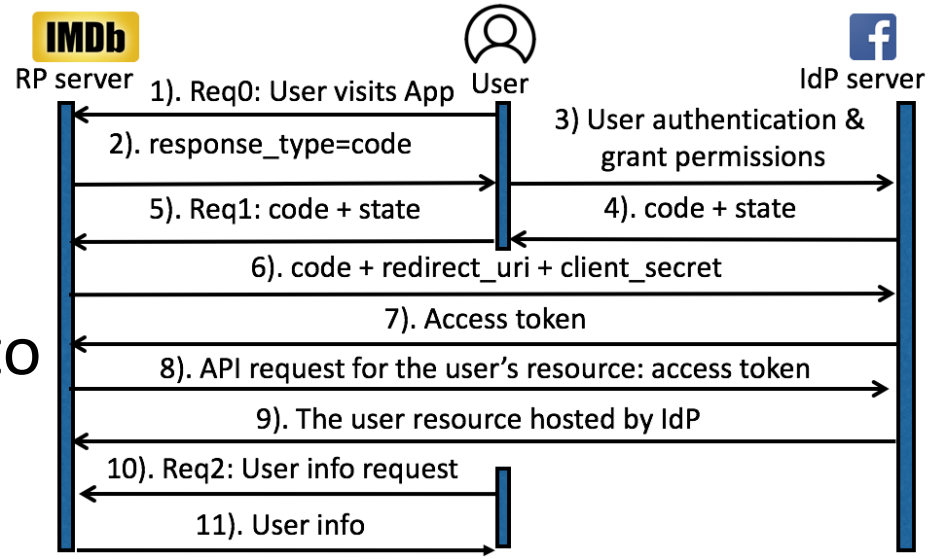


Single Sign-On Protocol Flow



Complication of Single Sign-On

- Multi-party systems
- Multi-step operations
- SDKs are provided to help RP developers to implement SSO services



SDK Usages

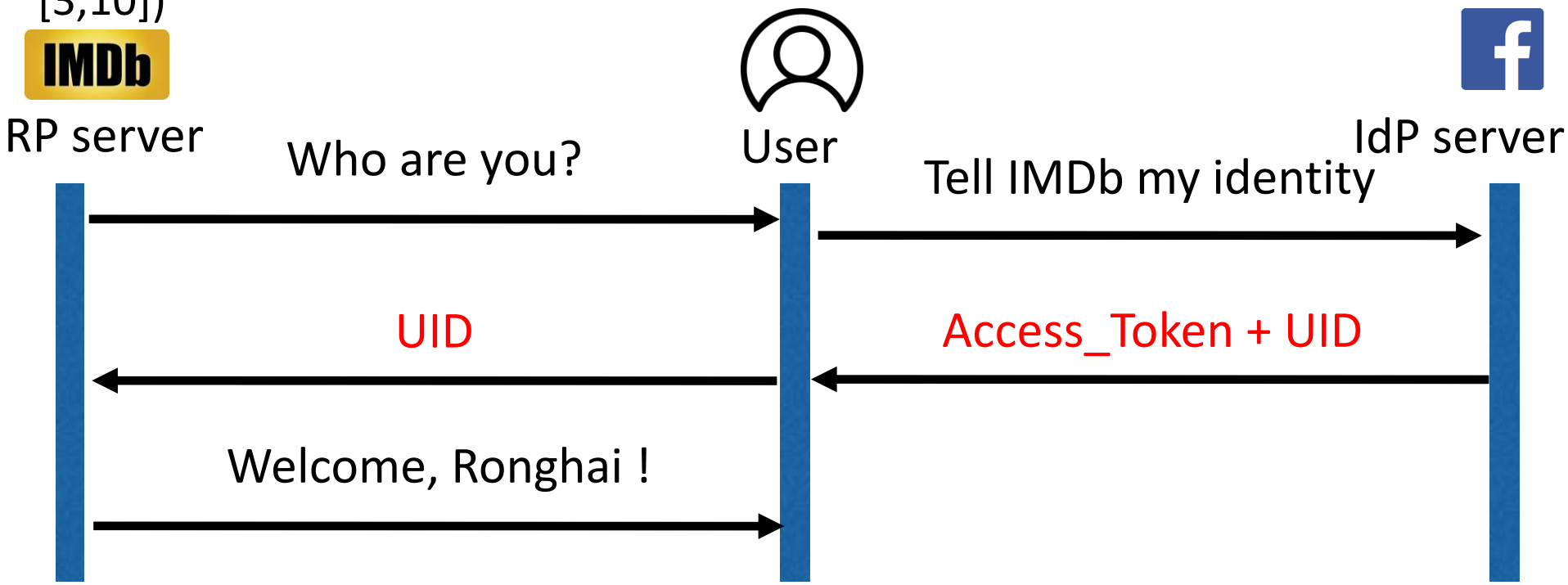
- SDKs are provided not only by IdPs but often by 4th-party SDK providers (for cross-IdP support of an application)
- If the SDK is not secure, any application using the SDK will be insecure!

SDK Names	# of Downloads
Facebook SDK	602,297
Request-OAuthLib	4,785,778
OAuthLib	6,476,894
Sinaweibopy	28,019
OAuth2Lib	Not found
Rauth	487,275
Python-weixin	1,404
BoxSDK	77,074
Renrenpy	10,387
Douban-client	30,601

*: The number is retrieved from PyPI statistics and is a conservative estimate. The installed number for many IdPs (*e.g.*, Facebook, Wechat, Renren, Douban), may not be included in the statistics.

Possible Attacks due to Vulnerabilities in SDKs

- Many attacks are due to the incorrect implementations of SDKs
  - For example, the SDK does not check the existence of access token (profile vulnerability [3,10])



[3] Yang, R., Lau W.C., Breaking and fixing mobile app authentication with OAuth2.0-based protocols In ACNS (2017)
[10] Yang, R., Lau, W. C., and Liu, T. Signing into one billion mobile app accounts effortlessly with Oauth 2.0. In BlackHat Europe (2016)

Prior Work on SSO Security

- Formal analysis of SSO protocol standards, including model checking [4,5,6] and cryptographic proof [7]

[4] Bai, G., Lei, J., Meng, G., Venkatraman, S. S., Saxena, P., Sun, J., Liu, Y., and Dong, J. S. AUTHSCAN: automatic extraction of web authentication protocols from implementations. In NDSS (2013)

[5] D. Fett, R. Kusters, and G. Schmitz. An expressive model for the web infrastructure: Defining and application to the Brower ID SSO system. In S&P (2014)

[6] Fett, D., Kusters, R., and Schmitz, G. A comprehensive formal security analysis of Oauth 2.0 In CCS (2016)

[7] Chari, S., Jutla, C. S., and Roy, A. Universally composable security analysis of Oauth v2.0

Prior Work on SSO Security (cont'd)

- Real-world vulnerability discovery using network traffic analysis [8,9,10] or Model-based automated testing [11,12]
- Discovery of hidden assumptions required for the proper use of SDK [13]

[8] Wang, R., Chen, S., and Wang, X., Signing Me into your Accounts through Facebook and Google: a Traffic-Guided Security Study on Commercially Deployed Single-Sign-On Web Services, IEEE S&P (2012)

[9] Wang, H., Zhang, Y., Li, J., Liu, H., Yang, W., Li, B., and Gu, D. Vulnerability assessment of OAuth implementations in Android applications. ACSAC (2015)

[10] Yang, R., Lau, W. C., and Liu, T. Signing into one billion mobile app accounts effortlessly with Oauth 2.0., BlackHat Europe (2016)

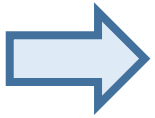
[11] Ferry, E., O'Raw, J., and Curran, K. Security evaluation of the Oauth 2.0 framework, Inf. & Comput. Security (2015)

[12] Yang, R., Li, G., Lau, W. C., Zhang, K., Hu, P. Model-based security testing: An empirical study on OAuth2.0 implementations, AsiaCCS (2016)

[13] Wang, R., Zhou, Y., Chen, S., Qadeer, S., Evans, D., and Gurevich, Y. Explicating SDKs: Uncovering assumptions underlying secure authentication and authorization, USENIX Security (2013)

Prior Work on SSO Security (summary)

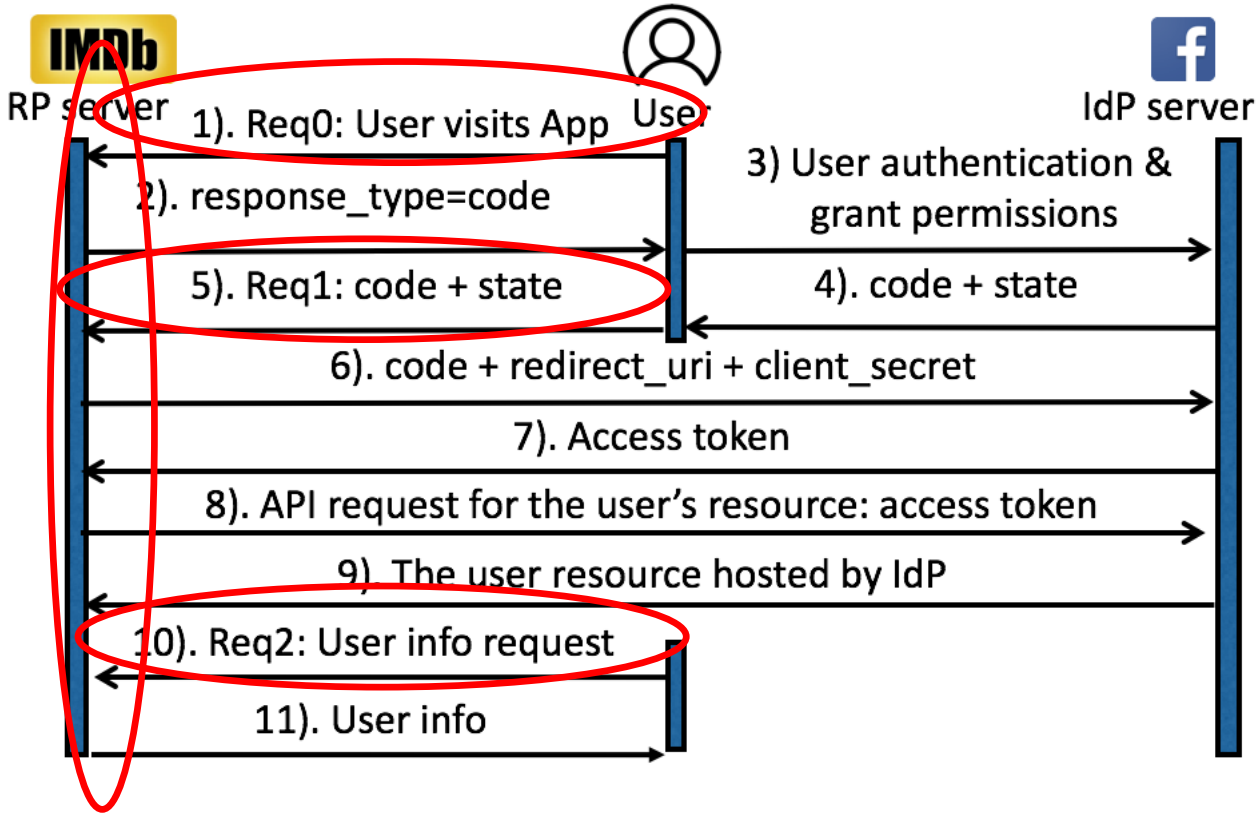
- Formal analysis of SSO protocol standards, including model checking [4,5,6] and cryptographic proof [7]
- Real-world vulnerability discovery using network traffic analysis [8,9,10] or model-based automated testing [11,12]
- Discovery of hidden assumptions required for the proper use of SDK [13]



Little effort has been devoted to a systematic testing of implementation flaws in SSO SDK internals

Goal & Scopes

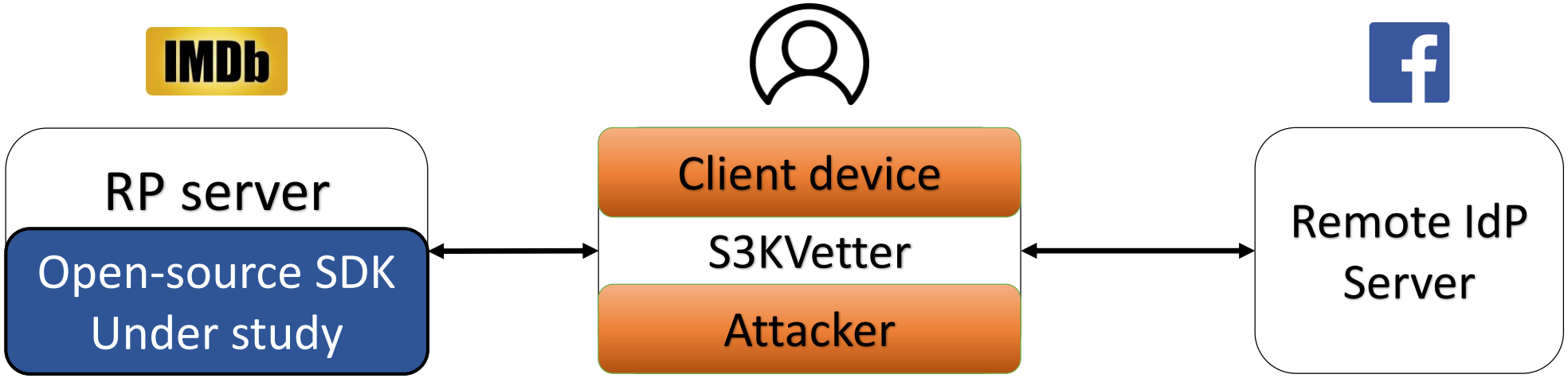
- Is an SSO SDK vulnerable by itself ?
  - Work properly under whatever inputs from the attacker?
- Focus on logic vulnerabilities of the RP server SDK internals



Threat Model

- The attacker can lure the victim to visit a malicious RP (mRP) server (to obtain a valid access token of the victim's IdP account, but binding only to the mRP)
- The attacker can setup an external machine and use his/her own account to freely communicate with the client, the IdP server, and the RP server
- When HTTPS is NOT used, the attacker can eavesdrop victim's communication

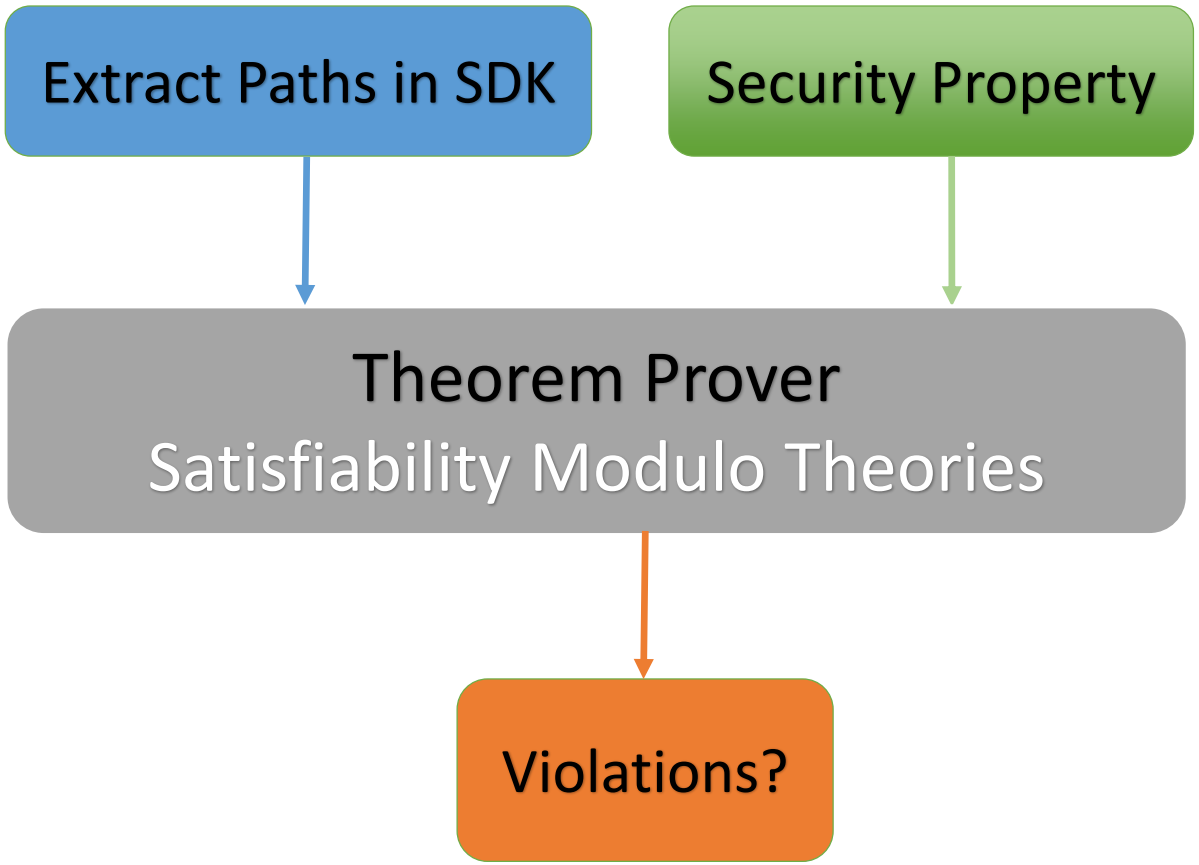
Roles of S3KVetter



- Single Sign-on SDK Vetter (S3KVetter)
  - Interact with the RP server and the IdP server as if it is the client device
  - Communicate with the RP server on behalf of the attacker

Overview

- Extract all the program paths from the SDK (via concolic execution)
- Define the security properties (i.e., expected behaviors) for SSO SDKs
- Check whether the security properties hold on every program path (via theorem prover)



Technical Challenges

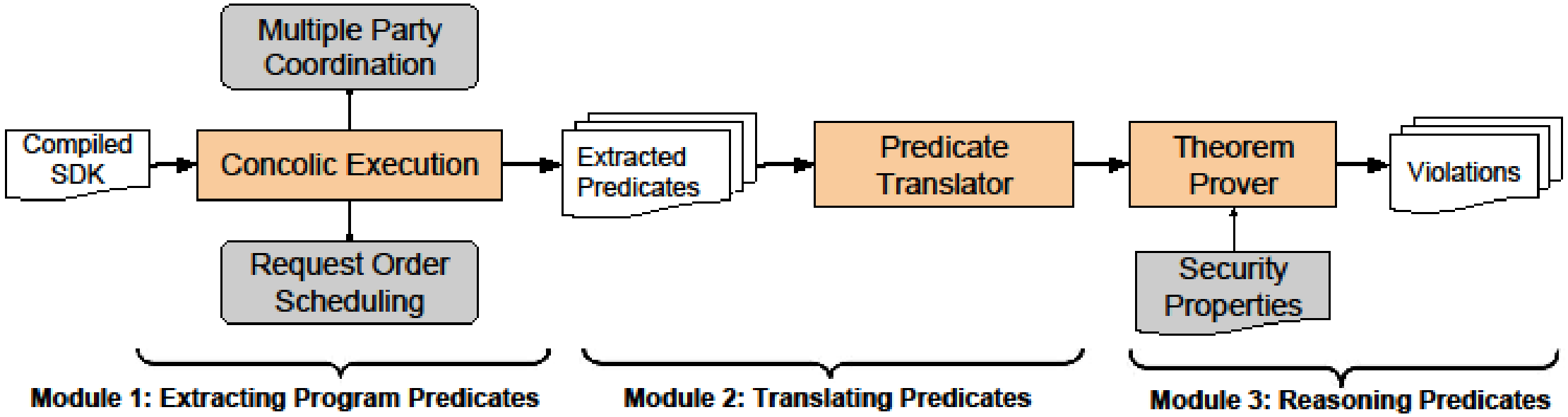
- Multiple-party communication and multi-lock-step operations
  - Some nonce parameters (e.g. code, state) can only be used for once
  - Some parameters are remotely generated and consumed by the remote server

⇨ Extra effort required to create/manage inputs to properly feed SSO SDKs

Issues of Existing Approaches for performing Symbolic Execution on Multi-party Systems

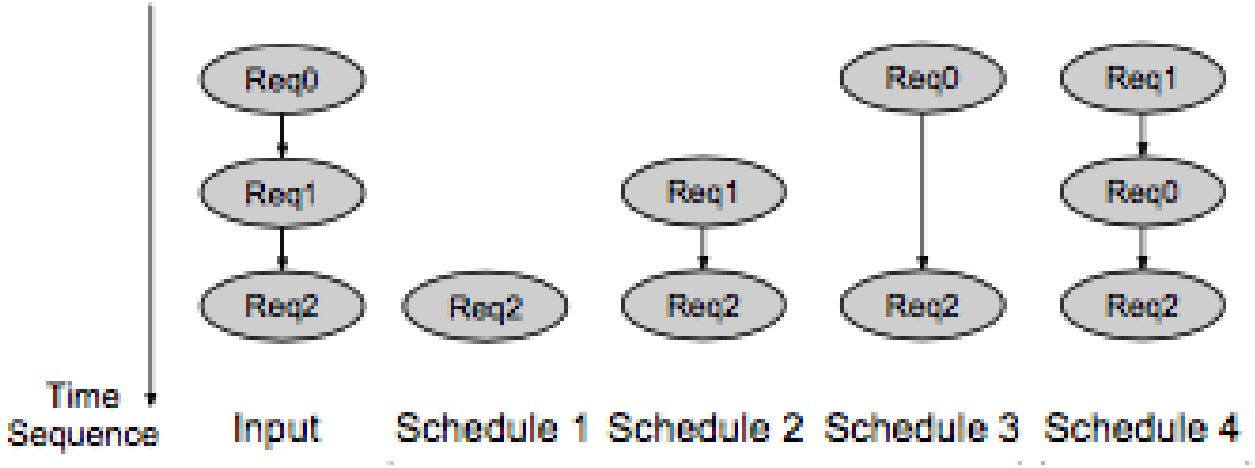
- Run the external functions (of SDK) concretely
  - Remote IdP API imposes limit on API access rate
- Return a random value of the same return type of external functions without execution, e.g., DART
  - This causes false positives to the testing results
- Symbolically explore the external functions, e.g., KLEENet
  - We do not have access to the source code of the IdP server

S3KVetter System Architecture



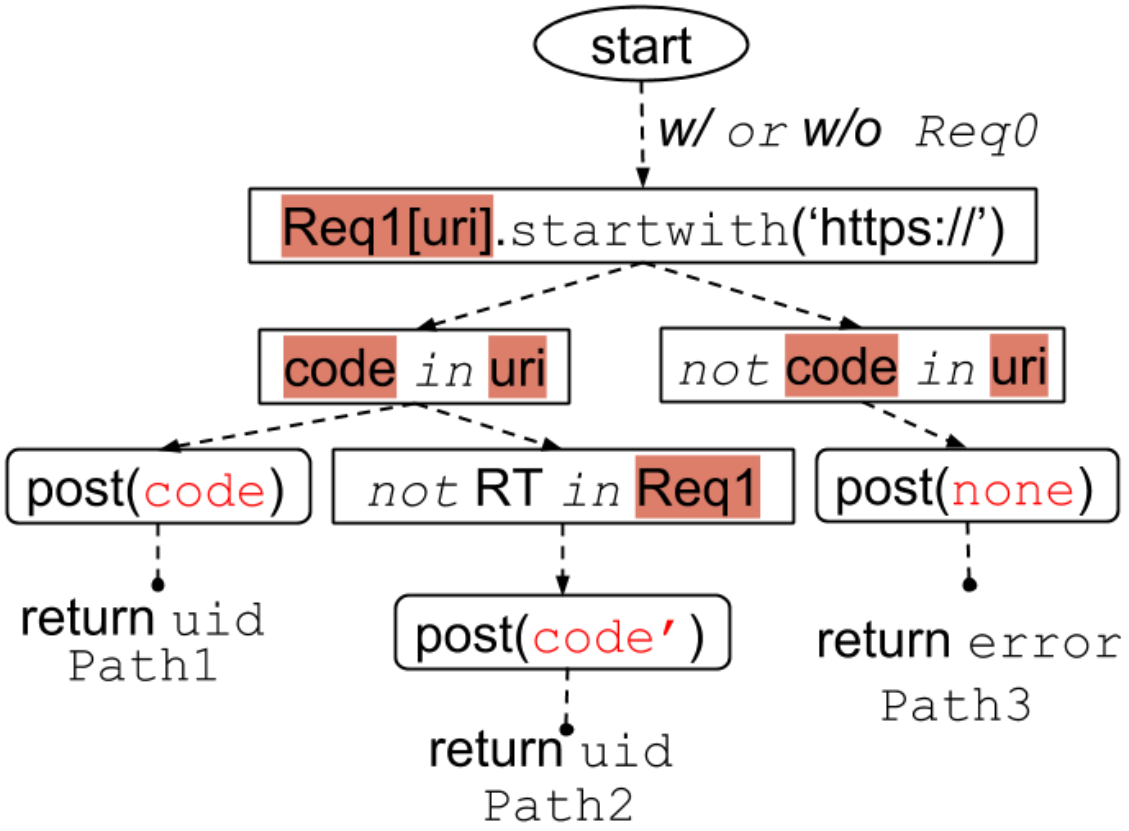
Scheduling Request Orders

- Use a general and simple scheduling algorithm
- Interest in authentication property
  - Completed by the last request only



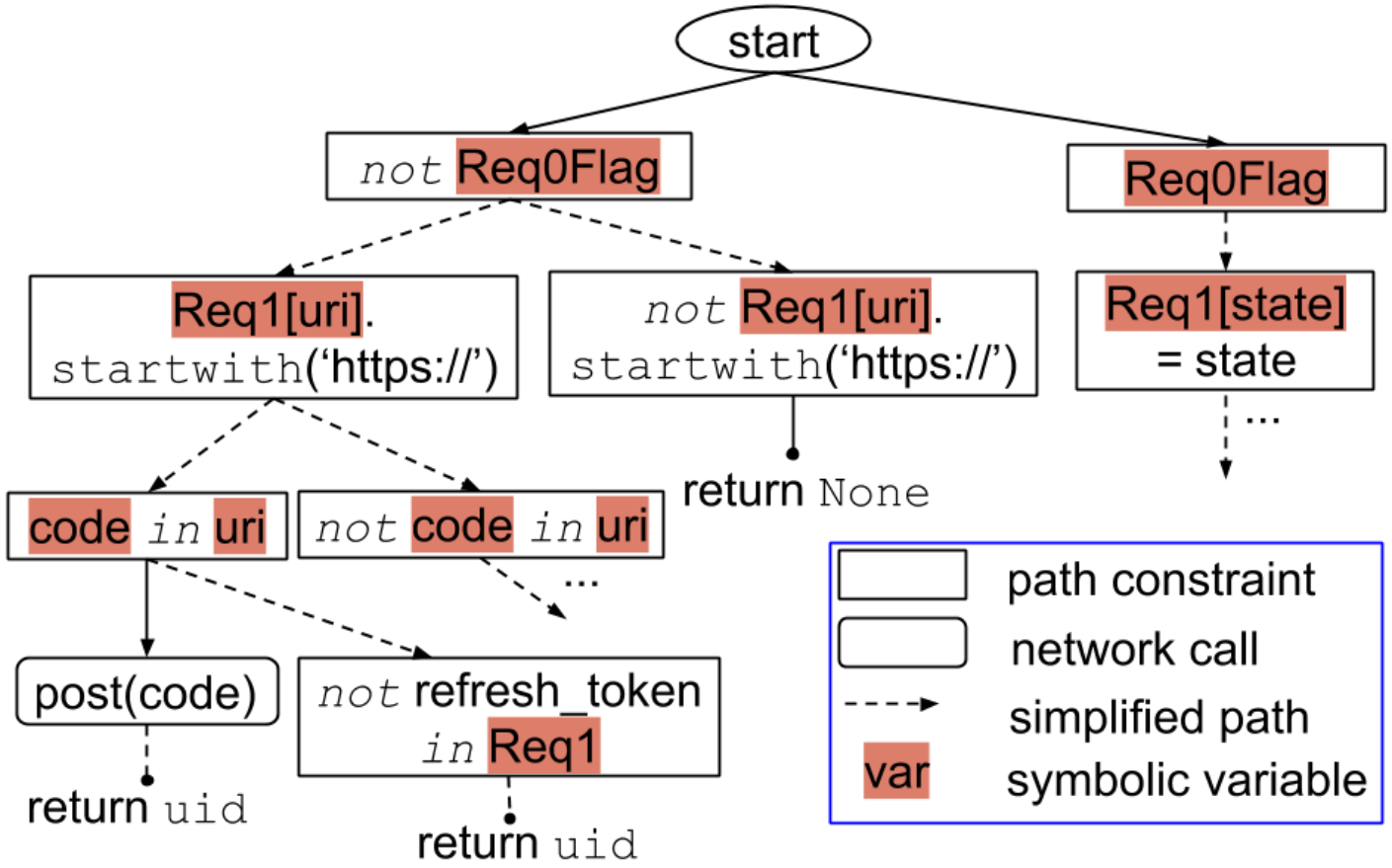
Coordinating Multi-party Communication

- S3KVetter simulates and modifies the external world for the SDK
  - S3KVetter generates a nonce parameter internally, on behalf of IdP
  - Use this nonce parameter, if it satisfies the conditions of the path to be explored
  - Locally solves the condition, if the nonce parameters does not satisfy the condition



Example Symbolic Predicate Tree of SDK

```
1  oauth = OAuth2Session(client_id,redirect_uri)
2
3  @app.route("/")
4  def init():
5    auth_url, state = oauth.authorization_url(
         base_url)
6    return redirect(auth_url)
7
8  @app.route("/callback", methods=["GET"])
9  def callback():
10   token = oauth.fetch_token(token_url,secret,
         auth_response=request.url)
11   session['oauth_token'] = token
12   return redirect(url_for('.profile'))
13
14 @app.route("/profile", methods=["GET"])
15 def profile():
16   return oauth.get('https://idp.com/user')
17
18 def fetch_token(token_url, secret,
       auth_response):
19   ...
20   if state and params.get("state", None)!=
         state:
21     raise MismatchingStateError()
```



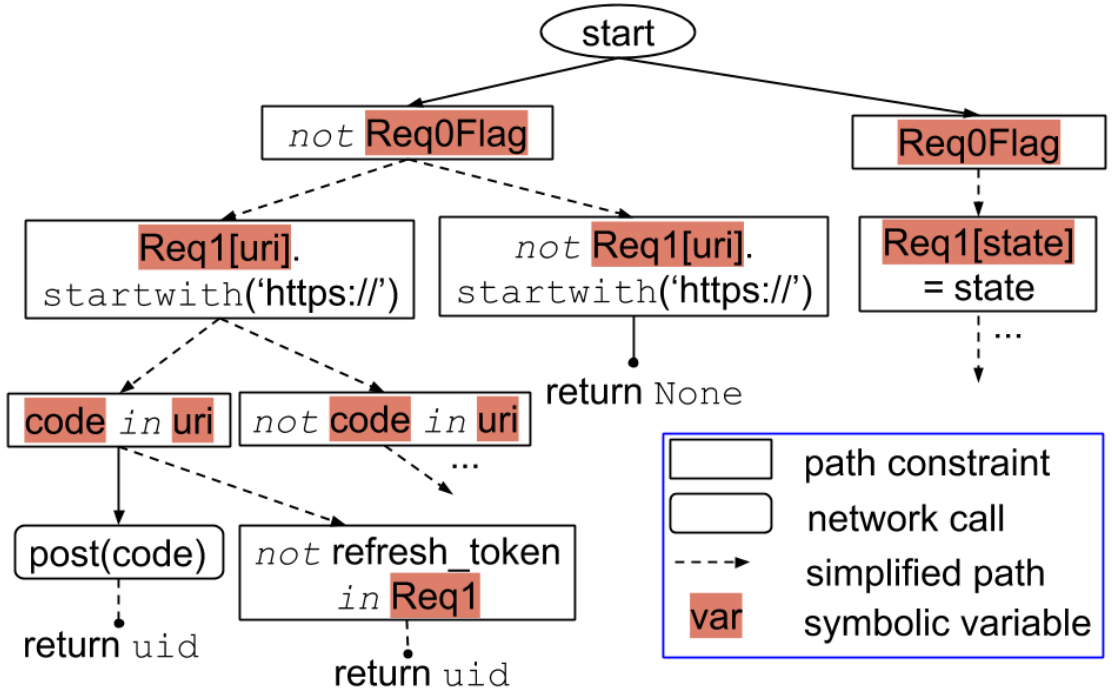
Translate the Predicate Tree

- Represent the predicate tree with SMT-Lib v2.0

not (= Req0Flag 0) and (**str.prefixof** ”https” Req1[uri]) and str.contains **uri code and …**

or not (= Req0Flag 0) and not (**str.prefixof** ”https” Req1[uri]) and …

or …



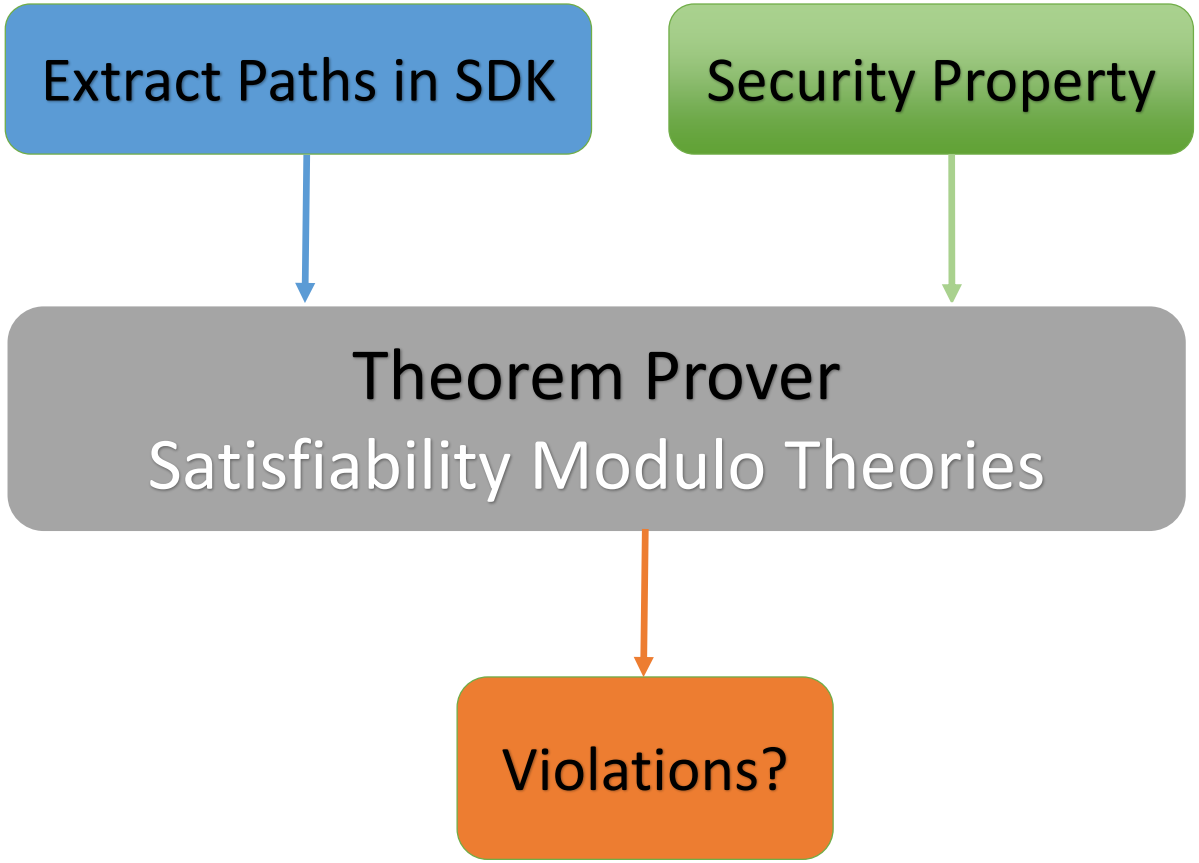
Formulation of Desired Security Properties

- Key observation: An RP server should login the user **if and only if** the **exact** user has actually authorized this **specific** RP

```
1 RPsession.uid == TokenRecordsOnIdP[RPsession.
      access_token].uid and
2 RPsession.uid == CodeRecordsOnIdP[RPsession.
      code].uid and
3 RPsession.uid == TokenRecordsOnIdP[RPsession.
      refresh_token].uid and
4 appid ==  TokenRecordsOnIdP[RPsession.
      access_token].appid and
5 appid ==  TokenRecordsOnIdP[RPsession.
      refresh_token].appid and
6 RPsession.uid == IdPsession.uid
```

Results Overview

- Discover 7 types of vulnerabilities on 10 popular SDKs
  - Four types are previously unknown
- Consequences:
  - Hijack user's account in the RP, e.g. due to Access Token injection
  - Sniff user's activities in the RP due to Use-before-Assignment of *state*



Statistics of SDK under Study

SDK Names	Lines of code	# of downloads	Grant flow under study	Baseline with unmodified PyExZ3				Improved result with S3KVetter			
				# of path discovered	statement coverage	branch coverage	# of bugs discovered	# of path discovered	statement coverage	branch coverage	# of bugs discovered
Facebook SDK	976	602,291	implicit	8	45%	37%	2	40	58%	56%	2
Request-OAuthLib	15432	4,785,778	code	322	37%	31%	0	649	42%	35%	2
OAuthLib	17917	6,476,894	code	640	41%	33%	1	1282	46%	39%	5
Sinaweibopy	800	28,019	code	2	43%	39%	2	6	47%	44%	2
OAuth2Lib	971	not found	code	2	73%	68%	0	4	83%	77%	1
Rauth	9241	487,275	code	2	41%	34%	2	14	43%	36%	2
Python-weixin	2736	1,404	code	2	32%	29%	2	6	38%	35%	2
Boxsdk	15277	77,074	code	2	44%	37%	2	12	55%	47%	2
Renrenpy	251	10,387	code	2	54%	46%	1	12	56%	50%	1
Douban-client	2092	30,601	implicit	1	49%	52%	2	2	62%	60%	3

- Compared to the Baseline, S3KVetter can achieve 2%-13% higher statement coverage and 2%-19% higher branch coverage for the SSO SDKs under test
- Discover 8 additional vulnerabilities in ten SSO SDKs

Summary of Discovered Vulnerabilities

SDK	Existing classes of vulnerabilities			New classes of vulnerabilities			
	Token substitution	no check of TLS	misuse or no use of state	use-before-assignment of state variable	Bypass MAC key protection	refresh_token injection	access_token injection
Facebook SDK	N	Y	Y	N.A	N.A	N	N
Request-OAuthLib	N	N	N	Y	N.A	Y	N
OAuthLib	Y	N	Y	N.A	Y	Y	Y
Sinaweibopy	N	Y	Y	N.A	N.A	N	N
OAuth2Lib	N	N	Y	N.A	N.A	N	N
Rauth	N	Y	Y	N.A	N.A	N	N
Python-weixin	N	Y	Y	N.A	N.A	N	N
Boxsdk	N	Y	Y	N	N.A	N	N
Renrenpy	N	N	Y	N.A	N.A	N	N
Douban-client	Y	Y	Y	N.A	N.A	N	N

- Use-before-Assignment of "State" variable => Allow sniffing of Victim activities via CSRF attacks
- Bypass MAC key, Refresh Token injection and Access Token Injection => Attacker can hijack Victim's RP account.

These vulnerabilities have been fixed after we informed the developers of the corresponding SDKs.

Example Vulnerability in OAuthLib: Access Token Injection

- Root cause of access token injection

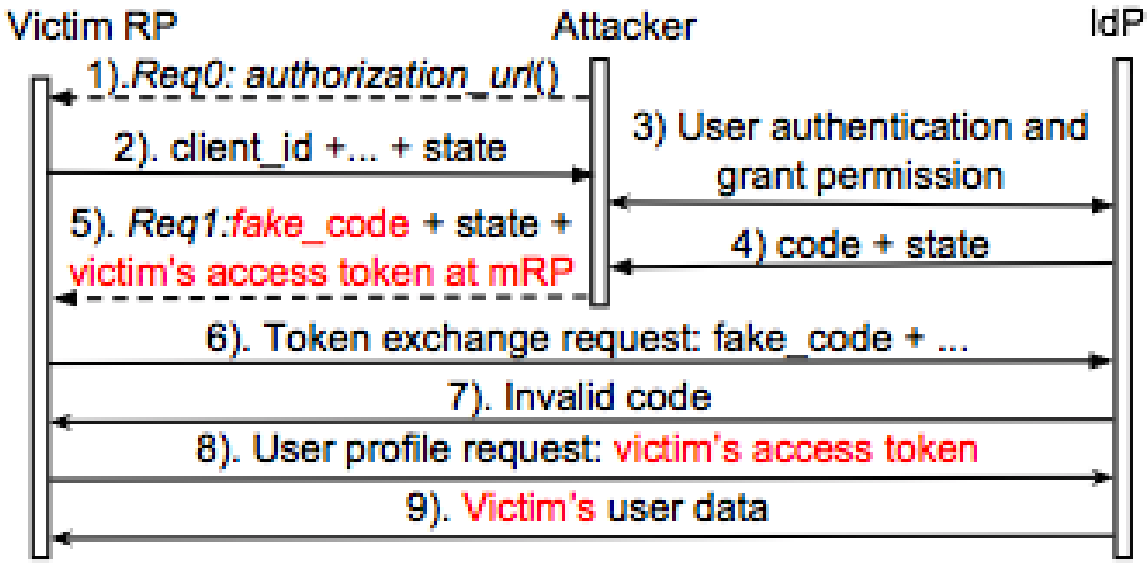
```
1  def _populate_attributes(self, resp):
2    if 'code' in resp:
3      self.code = resp.get('code')
4    if 'access_token' in resp:
5      self.access_token = resp.get('
           access_token')
6    if 'mac_key' in resp:
7      self.mac_key = resp.get('mac_key')
```

- An attacker can remotely inject any access token of her choice

https://RP.com?state=xxx&code=fake code
&access token=victim's access token at mRP

Example Exploit for Access Token Injection

- Assume the attacker has Alice's access token
  - Setup a malicious RP, and lure the victim to login
  - Inject victim's access token into the RP server
- The RP can obtain victim user's resource hosted in IdP
- The attacker can log into the RP as the victim user



Conclusion

- Measurement study and new findings
  - In-depth security analysis on ten popular SSO SDKs
  - 7 types of serious logic vulnerabilities, four are previously unknown
- Vulnerability detection for multi-party systems
  - Symbolizing request orders & multi-party coordination
  - Other usages: Payment system, etc.

Thank you!

Ronghai Yang[1,2], Wing Cheong Lau[1], Jiongyi Chen[1] and Kehuan Zhang[1]

[1]The Chinese University of Hong Kong and [2]Sangfor Technologies Inc.