OAuth 2.0 の概要とセキュリティ

2013/12/27

OAuth 2.0 概要

OAuthとは

- あるサービス(SP：Service Provider)が提供するAPIへのアクセス権を、他のサービス(Consumer)に与えるためのプロトコル
- **アクセス権限の付与のためのプロトコル**であって、認証のためのプロトコルではない

- アイデンティティ関連の各種技術が依って立つプロトコル(OAuth 2.0)
  - OpenID Connect：認証、フェデレーション
  - SCIM：プロビジョニング
  - UMA：アクセス権制御

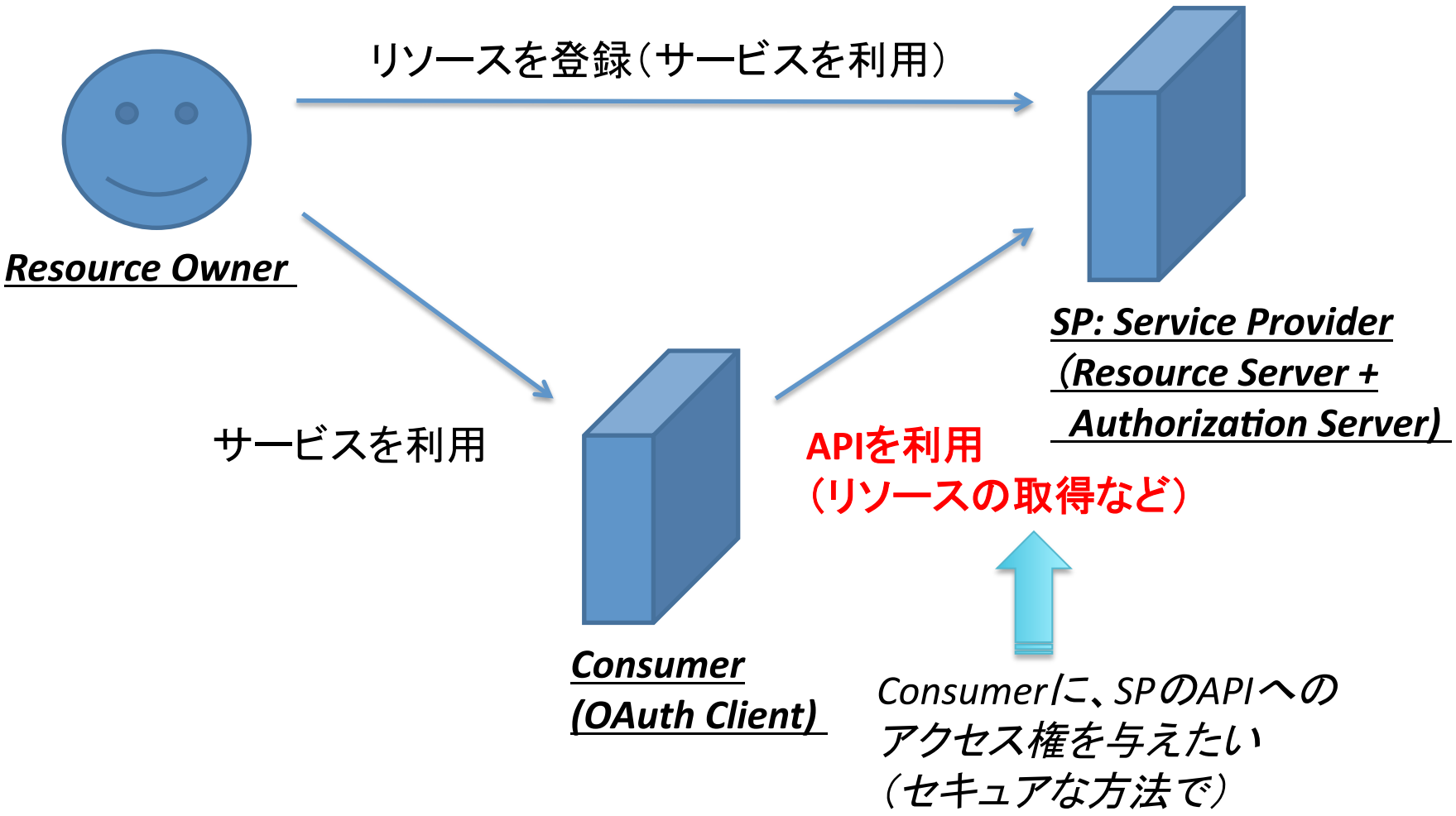
※ SCIM: System for Cross-domain Identity Management
※ UMA: User Managed Access

OAuthとは

- OAuth 2.0 は 2012年10月にRFCに
  - RFC 6749 - The OAuth 2.0 Authorization Framework
  - RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
  - RFC 6819 - The OAuth 2.0 Threat Model and Security Considerations

- 数々のメジャーなサービスで実装されている（2013年7月時点）
  - OAuth 2.0
    - Facebook
    - Google
    - GitHub
    - LinkedIn
  - OAuth 1.0
    - Twitter

登場人物


リソースを登録（サービスを利用）
Resource Owner
サービスを利用
Consumer
(OAuth Client)
APIを利用
（リソースの取得など）
SP: Service Provider
（Resource Server +
Authorization Server）
Consumerに、SPのAPIへの
アクセス権を与えたい
（セキュアな方法で）

UX

Consumerにログイン

SPを選択

UX

SPにログイン

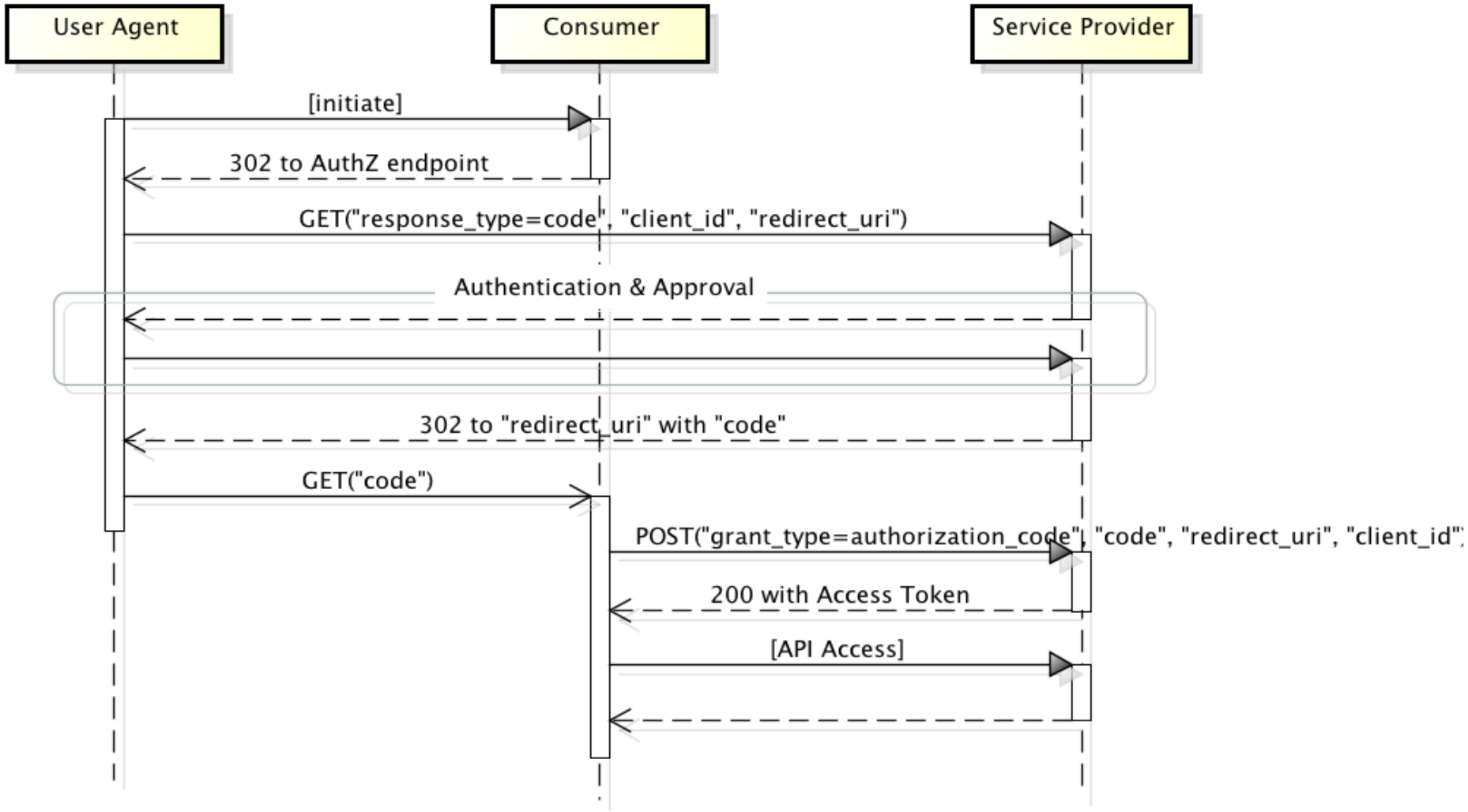
同意画面で、
ConsumerにSPへのアクセス権限を与える

この例では別の機能に関する同意画面も出る

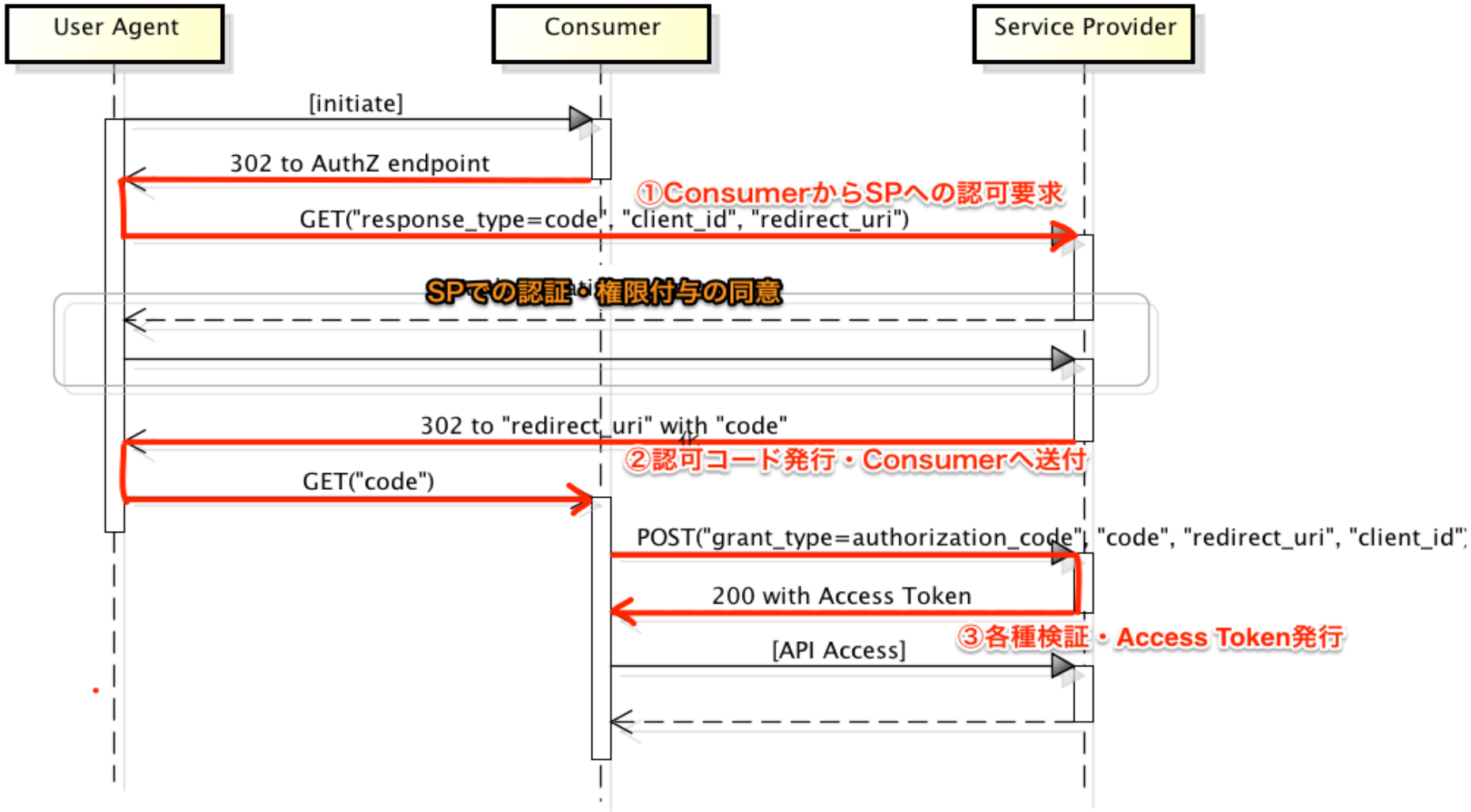
OAuth 2.0のフロー

- Authorization Code Grant（認可コードグラント）
  - client_secretを秘密にできる（＝リバースエンジニアリングされる危険がない）Consumerの場合に利用する
    - つまりWebアプリケーションのこと
- Implicit Grant（インプリシットグラント）
  - client_secretを秘密にできないクライアント（＝リバースエンジニアリングの危険がある）Consumerの場合に利用する
    - ブラウザ上で実行される（JavaScript、Flash...）アプリケーション
    - モバイルアプリ
  - SCIMの文脈では、Consumerがファイアウォール内に存在する場合にも利用される

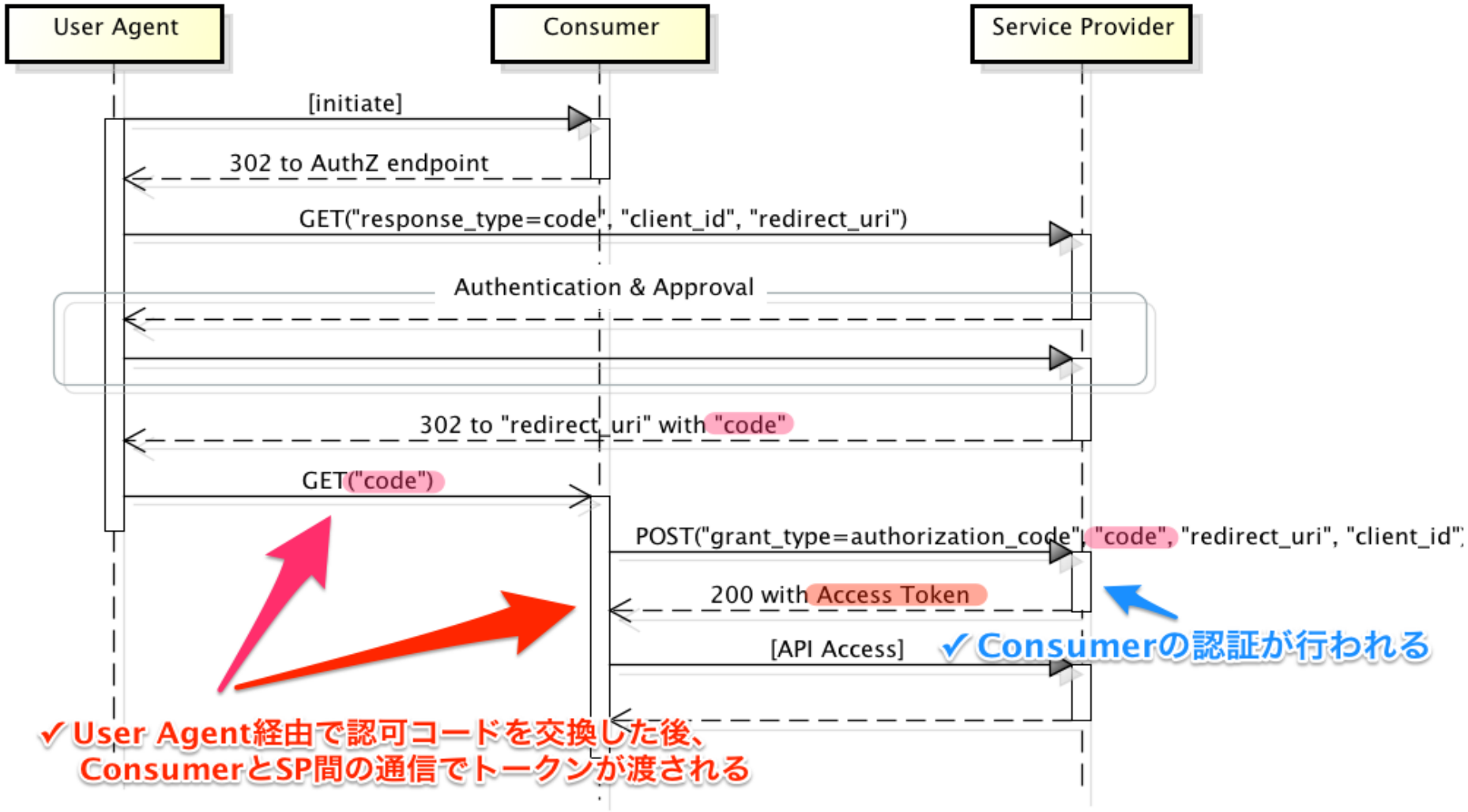
Authorization Code Grant
（認可コードグラント）



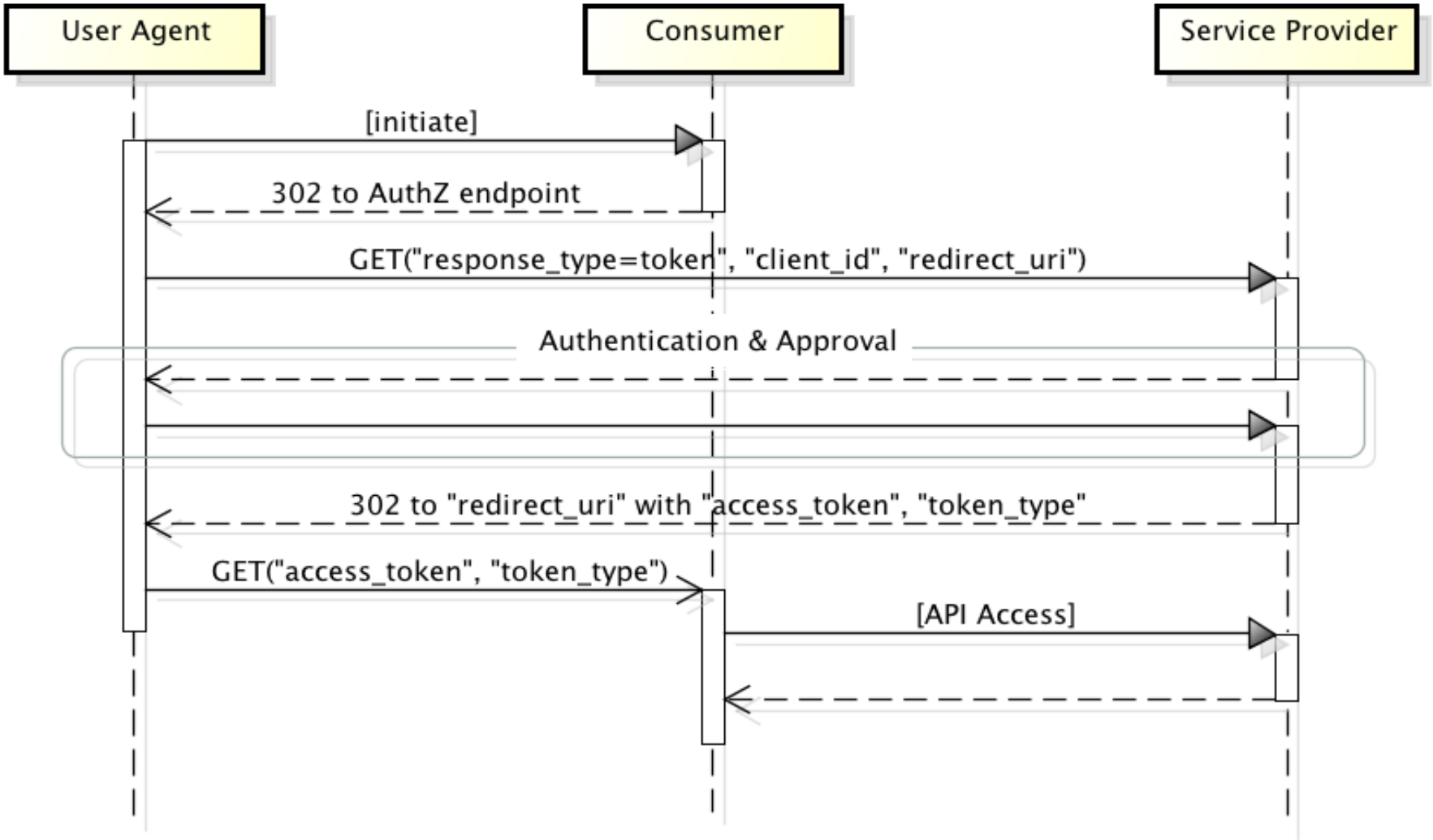
Authorization Code Grant
（認可コードグラント）



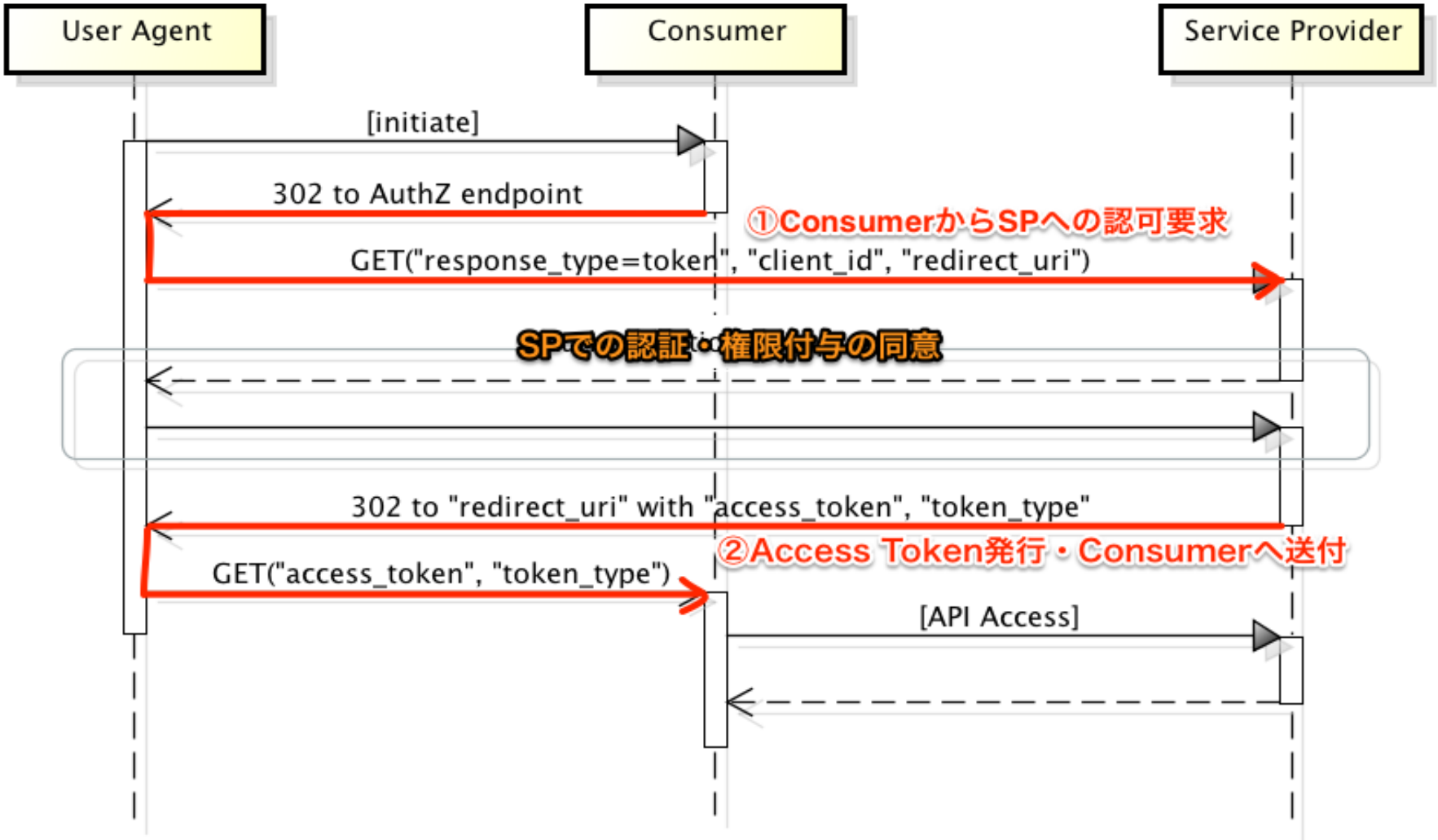
Authorization Code Grant
（認可コードグラント）



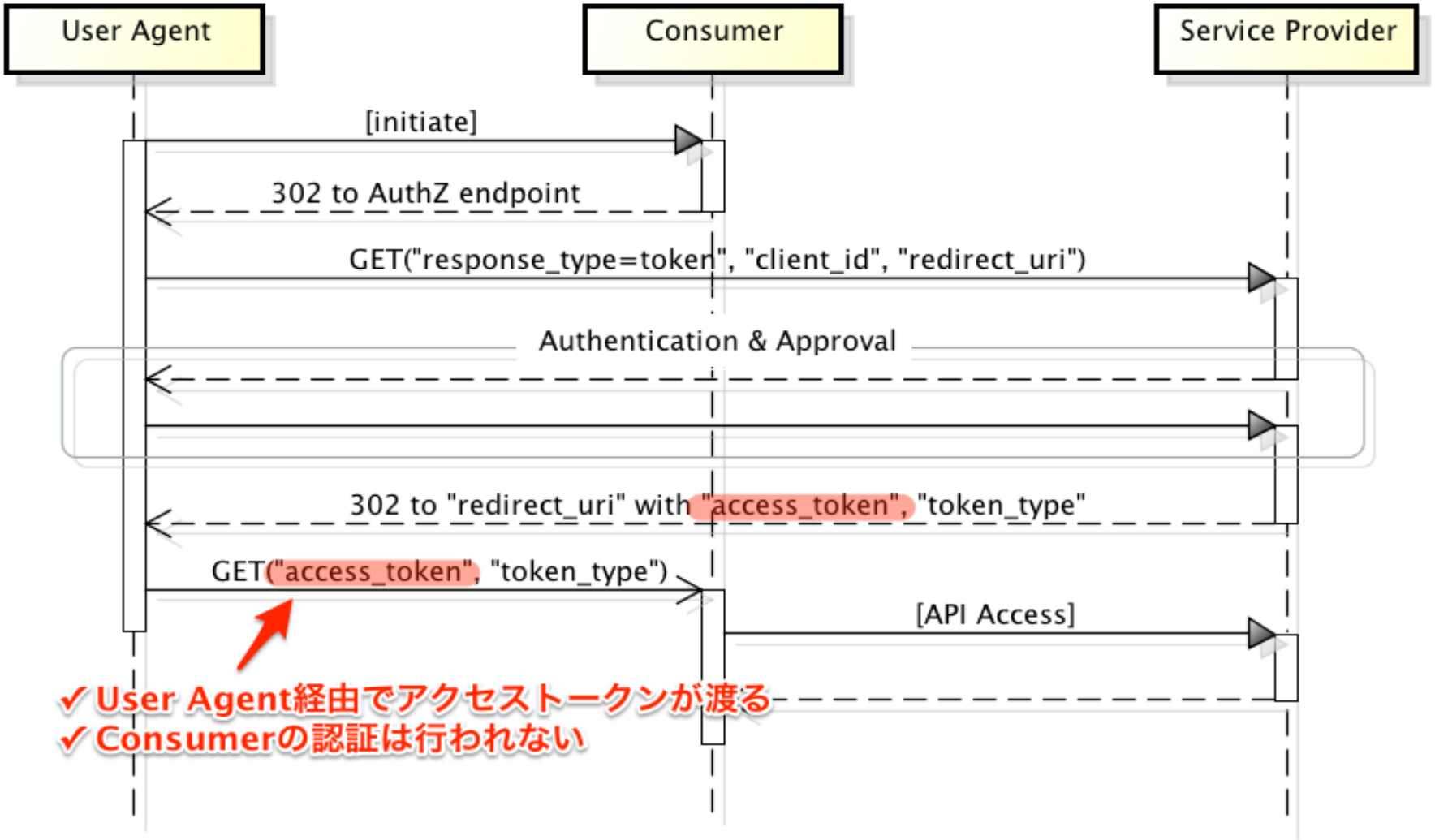
Implicit Grant
（インプリシットグラント）



Implicit Grant
（インプリシットグラント）



Implicit Grant
（インプリシットグラント）



OAuth 2.0 セキュリティ上の考慮事項

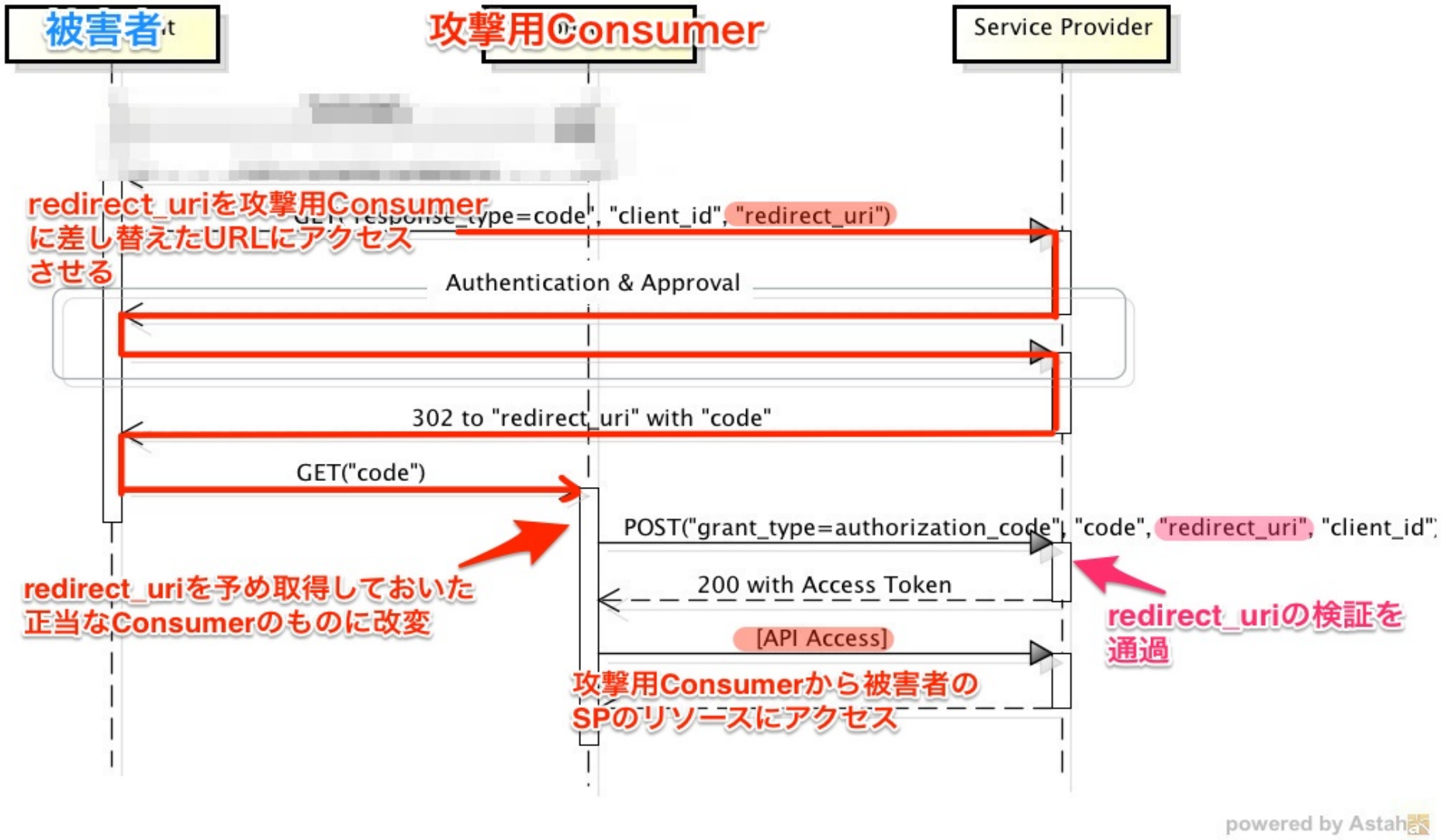
攻撃例(1) Consumerの偽装

- 攻撃手法
  - Consumerの検証の不備があるSPを狙う
  - 攻撃用のConsumerを立てて、正当なConsumerとしてSPに認可トークンを発行させる
- 対策
  - Authorization Code Grantの場合
    - 事前にConsumerに対しクレデンシャル（Client Secret)を発行し、アクセストークン発行前にConsumerを認証する
  - Implicit Grantの場合
    - アクセストークンリクエストで使用するredirect_uriを事前にSPに登録しておき、アクセストークン発行前にredirect_uriが正当であることを確認する(MUST)
    - Resource OwnerによるConsumerの確認を行うなどのステップを設けても良い

攻撃例(2) redirect_uriの改ざん

- 攻撃手法
  - 対象の環境
    - redirect_uriの事前登録は行わない（認可リクエスト中のredirect_uriパラメータをつかって、redirect_uriを渡す）
    - ↑ つまり、フローはAuthorization Code Grant
      （Implicit Grantでは、redirect_uriの事前登録は必須）
  - 手順
    1) 攻撃者は、正当なConsumerを使って認可リクエストを発行し、そのURIを取得する
    2) 1)のURIのredirect_uriパラメータを罠サイトのURIに差し替えたものを用意する
    3) 1)のURIのredirect_uriパラメータを保存しておく
    4) 攻撃者は、被害者を騙して 2)のURIのリンクをクリックさせる
       →被害者はSPと認証、認可操作を行い、罠サイトにリダイレクトされる
    5) 罠サイトは、3)で保持した正規のredirect_uriを使って、アクセストークンリクエストを発行する
       →罠サイトに対してアクセストークンが発行される

攻撃例(2) redirect_uriの改ざん



攻撃例(2) redirect_uriの改ざん

- 対策
  - SPで、認可リクエスト中のredirect_uriと、アクセストークンリクエスト中のredirect_uriが一致することを検証する(MUST)
  - redirect_uriがSPに事前登録される場合(SHOULD)、各リクエスト中のredirect_uriと、登録されたredirect_uriの一致を確認することも有効
- RFC上の該当記述
  - 10.6. Authorization Code Redirection URI Manipulation

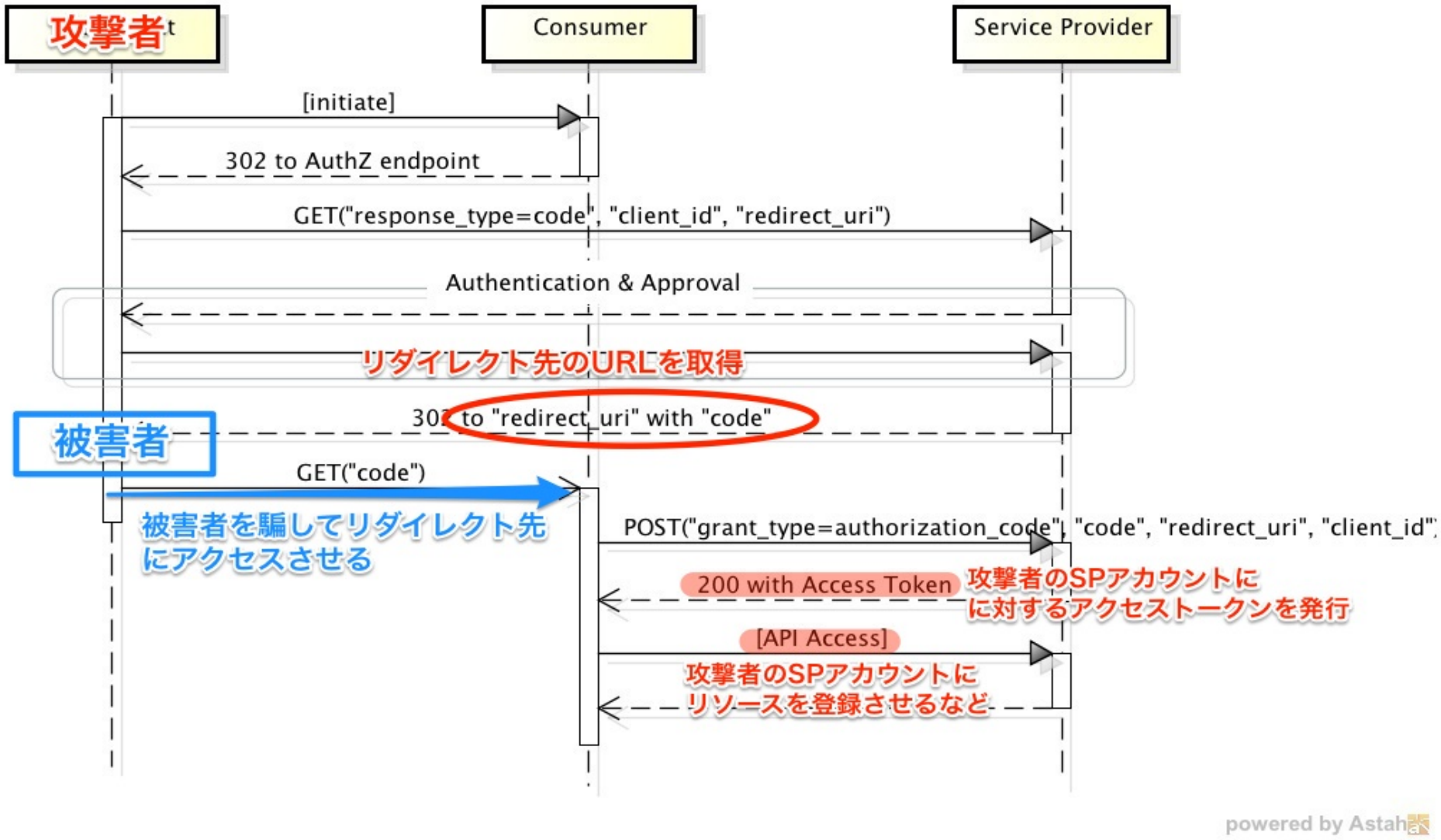
攻撃例(3) CSRF

- 攻撃手法
  - 手順
    1) 攻撃者は、通常の手順で、SPでの認可までを行う
    2) 認可後のリダイレクト先のURLを取得して、被害者にアクセスさせる

→ 攻撃者の認可トークンを被害者に使わせた状態で、Consumerにアクセスさせる

→ Consumerを操作すると、攻撃者としてSPのAPIが実行される

攻撃例(3) CSRF


攻撃者
Consumer
Service Provider
[initiate]
302 to AuthZ endpoint
GET("response_type=code", "client_id", "redirect_uri")
Authentication & Approval
リダイレクト先のURLを取得
302 to "redirect_uri" with "code"
被害者
GET("code")
被害者を騙してリダイレクト先
にアクセスさせる
POST("grant_type=authorization_code", "code", "redirect_uri", "client_id"
200 with Access Token
攻撃者のSPアカウントに
に対するアクセストークンを発行
[API Access]
攻撃者のSPアカウントに
リソースを登録させるなど
powered by Astah

攻撃例(3) CSRF

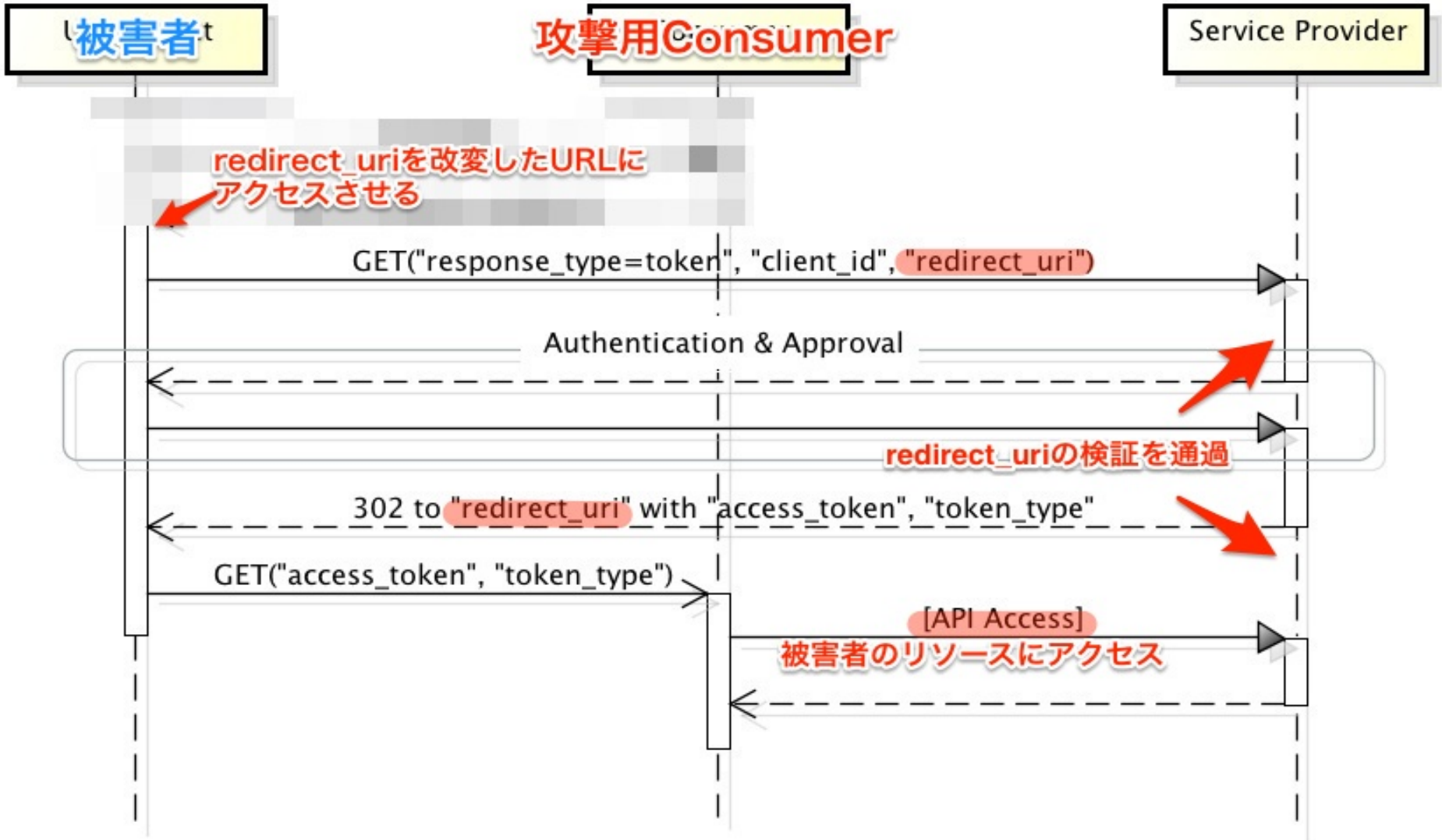
- 対策
  - “state”パラメータにより、認可リクエストとアクセストークンリクエストが同じセッションで行われていることをチェックする
- RFC上の該当記述
  - 10.12. Cross-Site Request Forgery
- 参考
  - 「OAuthのセキュリティ強化を目的とする拡張仕様を導入しました」http://alpha.mixi.co.jp/2013/12020/

攻撃例(4) オープンリダイレクタ

- 攻撃手法
  - 対象の環境
    - Consumerの検証をredirect_uriの事前登録に依存している
    - redirect_uriの登録をURIの一部のみで許可している
  - 手順
    1) 攻撃者は、redirect_uriの一部を、正当なConsumerと共有する（正当性のチェックを通過できる）ようなConsumerを作成する
    2) redirect_uriを攻撃用のConsumerのURIに差し替えた、認可リクエストURIを作成する
    3) 攻撃者は、被害者を騙して 2)のURIのリンクをクリックさせる
    4) 被害者は認証、認可操作を行う

    → 認可コード or アクセストークンが攻撃用のConsumerに送信される

攻撃例(4) オープンリダイレクタ



攻撃例(4) オープンリダイレクタ

- 対策
  - Authorization Code Grantの場合
    - client_id / client_secretを用いた認証など、redirect_uriに依存しないConsumer検証を行う
  - Implicit Grantの場合
    - 事前登録するredirect_uriとして、URI全体を用いる

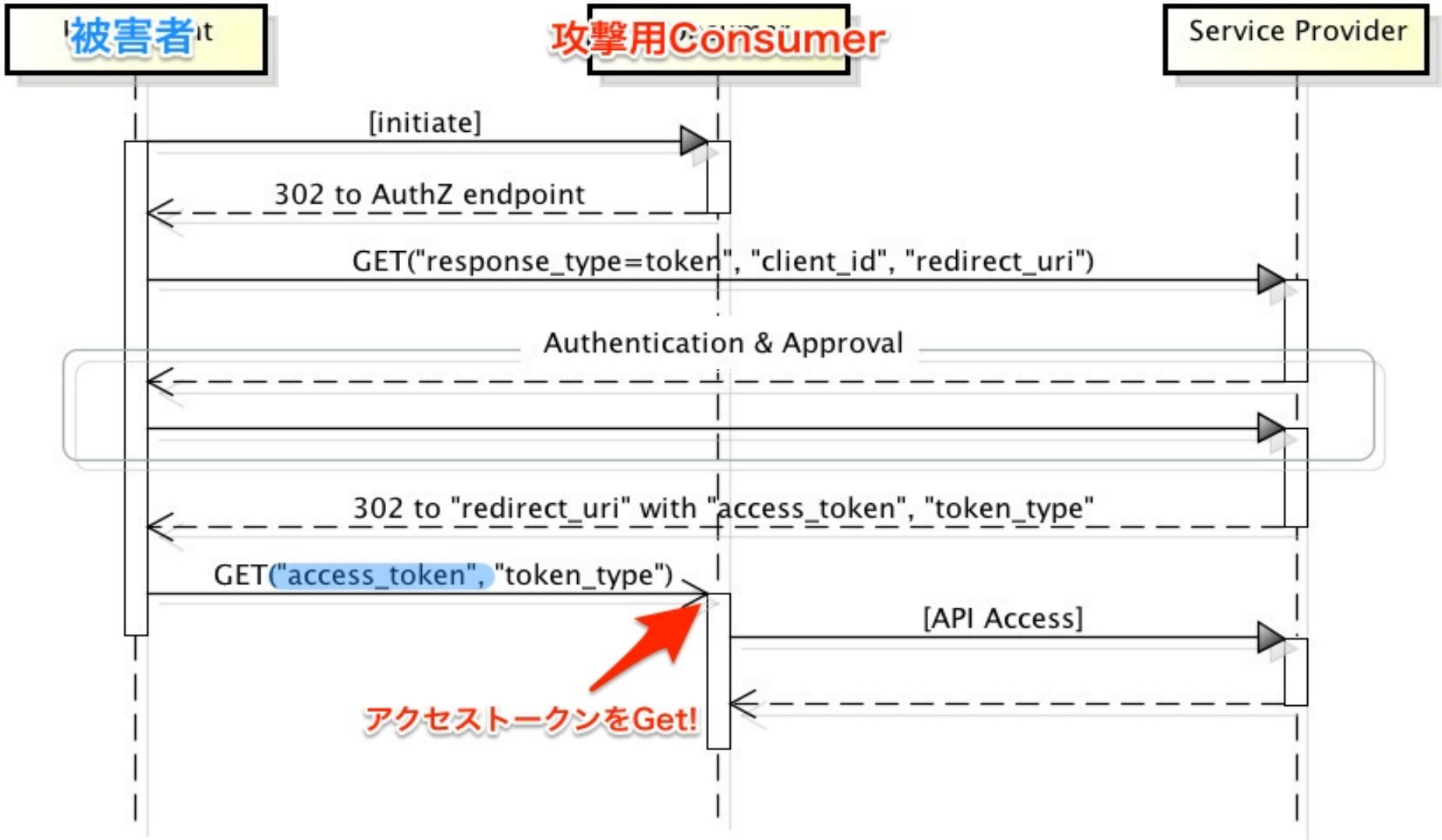
- RFC上の該当記述
  - 10.15. オープンリダイレクタ

攻撃例(5)
Implicit Grantにおけるトークン不正利用

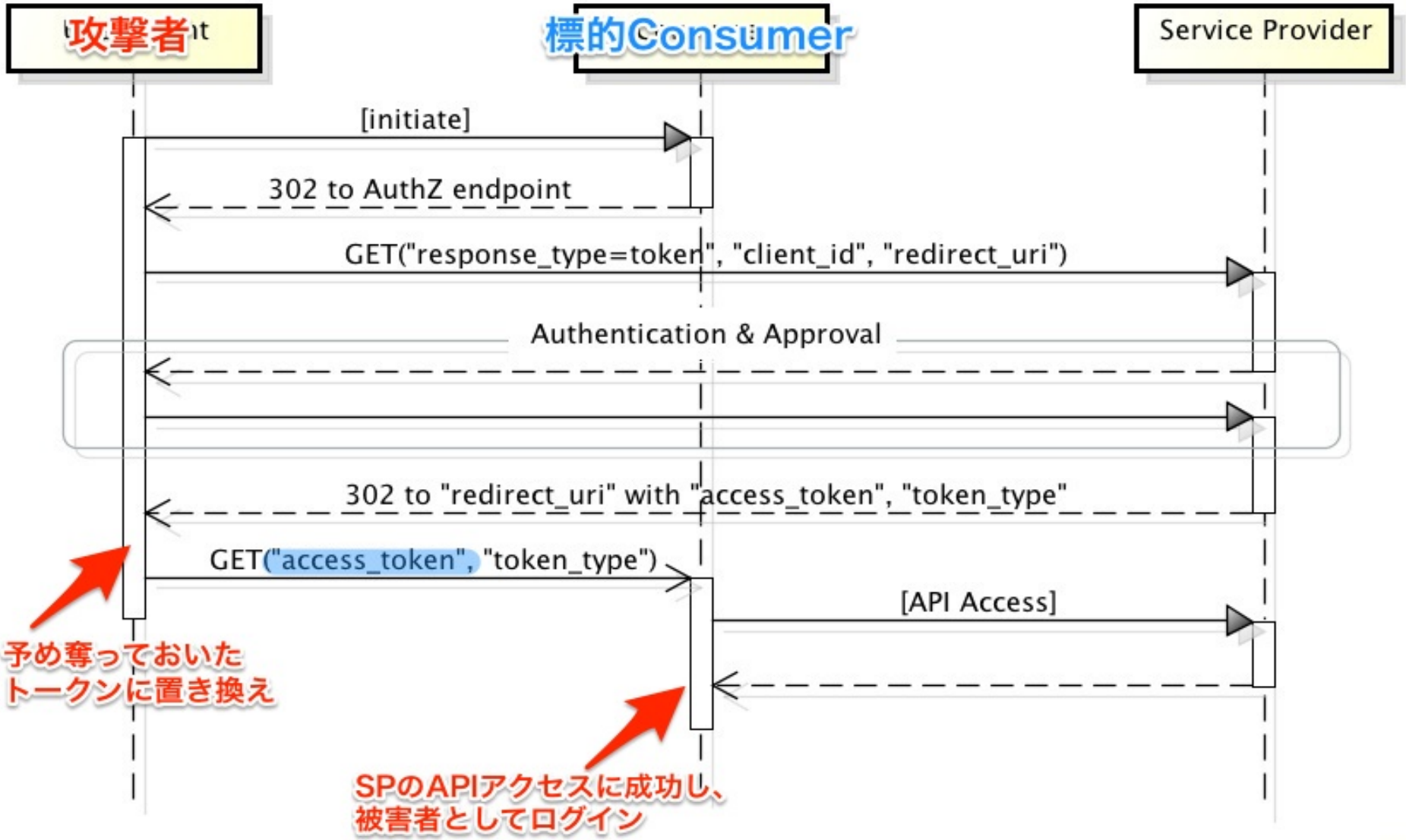
- 攻撃手法
  - 対象の環境
    - Implicit Grantを利用しているConsumerで、OAuth 2.0のアクセストークンをリソースオーナーの認証に使用しているもの
  - 手順
    1) 攻撃者は、攻撃対象のConsumerと同じSPを利用するConsumer（不正Consumer）を作成する
    2) 被害者が不正Consumerにアクセスし、これに対するアクセストークンを発行する
    3) 攻撃者は、2)のアクセストークンを取得する
    4) 攻撃者は、攻撃対象のConsumerにアクセスし、SPでの認証・認可操作を行う
    5) 攻撃対象のConsumerにリダイレクトされる際に、リクエスト中のアクセストークンを2)で取得したものに差し替える

    → 攻撃者は、被害者として、攻撃対象のConsumerにログインする

攻撃例(5)
Implicit Grantにおけるトークン不正利用



攻撃例(5)
Implicit Grantにおけるトークン不正利用



攻撃例(5)
Implicit Grantにおけるトークン不正利用

- 対策
  - OAuth 2.0のアクセストークンを、認証に使わない
  → OAuthは認可のプロトコル。認証にはOpenIDを使用する

- RFC上の該当記述
  - 10.16. インプリシットフローにおけるリソースオーナーなりすましのためのアクセストークン不正利用
- 参考
  - 「単なる OAuth 2.0 を認証に使うと、車が通れるほどのどでかいセキュリティー・ホールができる」 http://www.sakimura.org/2012/02/1487/

他

- ここまでに示したもの以外にも、セキュリティ上の考慮事項は大小多岐にわたる

- 実装者はRFCを参照すべし
  - 「The OAuth 2.0 Authorization Protocol - RFC 6749 10. Security Considerations」
    → Oauth 2.0仕様が定める考慮事項
  - 「OAuth 2.0 Threat Model and Security Considerations - RFC 6819」
    → Oauth 2.0仕様の範囲を超え、さらなるセキュリティ上の検討項目を提示

参考資料一覧

OAuth 2.0 参考資料

- 入門編
  - @IT「デジタル・アイデンティティ技術最新動向」シリーズ
    - 「第1回 「OAuth」の基本動作を知る」 http://www.atmarkit.co.jp/fsecurity/rensai/digid01/01.html
    - 「第2回 RFCとなった「OAuth 2.0」――その要点は？」 http://www.atmarkit.co.jp/ait/articles/1209/10/news105.html

OAuth 2.0 参考資料

- 発展編①
  - OAuth 2.0 のRFCの日本語翻訳
    - 「The OAuth 2.0 Authorization Protocol - RFC 6749」 http://openid-foundation-japan.github.com/rfc6749.ja.html
    - 「The OAuth 2.0 Authorization Framework: Bearer Token Usage - RFC 6750」 http://openid-foundation-japan.github.com/rfc6750.ja.html
    - 「OAuth 2.0 Threat Model and Security Considerations - RFC 6819」 http://openid-foundation-japan.github.com/rfc6819.ja.html
  - 書籍
    - 「Getting Started With OAuth 2.0 / Ryan Boyd（著）」 http://www.amazon.co.jp/dp/1449311601/

OAuth 2.0 参考資料

- 発展編②
  - 主要サービスのAPIドキュメント
    - Facebook
      https://developers.facebook.com/docs/authentication/
    - Google
      https://developers.google.com/accounts/docs/OAuth2
    - GitHub
      http://developer.github.com/v3/oauth/
    - LinedIn
      https://developer.linkedin.com/documents/authentication
    - Twitter(OAuth 1.0)
      https://dev.twitter.com/docs/auth/oauth

OAuth 2.0

- マニアック編
  - 仕様策定や、実装に関わっている人たちのブログ
    - 「OpenID ファウンデーション・ジャパン」（OpenID ファウンデーション・ジャパンの公式ブログ）
      http://blog.openid.or.jp/
    - 「r-weblife」(mixiのエンジニア）
      http://d.hatena.ne.jp/ritou/
    - 「.Nat Zone」（米国OpenID Foundation理事長）
      http://www.sakimura.org/