平成 23 年度　情報システム技術分野　分野別研修　実施報告

総合技術センター
情報システム技術分野　齊原　啓夫（Hiroo Saihara）
情報システム技術分野　菊地　真美（Mami Kikuchi）
情報システム技術分野　井上　富夫（Tomio Inoue）
情報システム技術分野　石井　純也（Junya Ishii）
情報システム技術分野　木戸　崇博（Takahiro Kido）
分析・解析技術分野　岡山 恵美子（Emiko Okayama）
計測・制御技術分野　片岡　由樹（Yoshiki Kataoka）
計測・制御技術分野　山中　卓也（Takuya Yamanaka）
運営・管理支援分野　藤田　智弘（Tomohiro Fujita）

1 はじめに

平成 23 年度の分野別研修として「安全な Web アプリケーション演習および実習」を実施したので報告する。

2 分野別研修の概要

近年の情報セキュリティでは，Web アプリケーションの脆弱性を利用して不正なアクセスを行う攻撃が多く発生している。

本研修では，参考書「安全な Web アプリケーションの作り方」を利用して輪講形式で研修を行った。担当は以下の通りである。

1 回目　日時：9/1(木)　10:00- 担当：齊原
　ガイダンス,3 章
2 回目　日時：9/8(木)　 9:00- 担当：木戸
　4 章 4.1，4.2
3 回目　日時：9/15(木)　9:00- 担当：片岡
　4 章 4.3，4.5
4 回目　日時：9/22(木)　9:00- 担当：藤田
　4 章 4.4
5 回目　日時：9/29(木)　9:00- 担当：山中
　4 章 4.6
6 回目　日時：10/6(木)　9:00- 担当：菊地
　4 章 4.7，4.8
7 回目　日時：10/13(木) 9:00- 担当：石井
　4 章 4.9，4.10，4.11
8 回目　日時：10/20(木) 9:00- 担当：井上
　4 章 4.12，4.13
9 回目　日時：10/27(木) 9:00- 担当：全員
分野研修最終,情報システム統一研修報告

研修期間は 9 月からの 2 ヶ月間で，担当のページを各自でまとめて発表する輪講形式で行った。報告は各自が工夫して行いスライドや資料を作成した。また脆弱性を確認するためにブラウザ上で動作するデモを取り入れて実習を行った。また最終日には情報セキュリティの小テストや独立行政法人情報処理推進機構（IPA）の 2011 年版 10 大脅威を確認した。

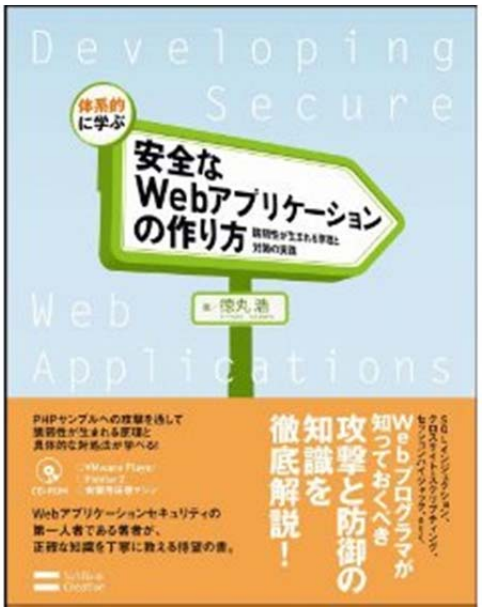

図 1 参考書

参考書「安全な Web アプリケーションの作り方」を輪講形式で輪読・議論を行うことにより、Web アプリケーションの脆弱性だけではなく、ネットワーク全般の攻撃と防御の知識を得る。また脆弱性の確認や、それに対する防御の仕組みを具体的に学ぶ。

3 Web セキュリティの基礎（担当：齊原）

3.1 HTTP とセッション管理

Web アプリケーションにおいて，どの情報が漏洩しやすいのか，どの情報が書き換えられるのか，安全に情報を保持するにはどのようにすればよいのか考える。また情報伝達に利用される HTTP とセッション管理についての考え方を説明する。

○HTTP

Web ページを閲覧する際にはブラウザを用いて行う。サーバへの要求として HTTP リクエストを送信して，サーバはブラウザへの返信として HTTP レスポンスを返す。以下の図に Web ページ閲覧時の HTTP リクエストとレスポンスを示す。

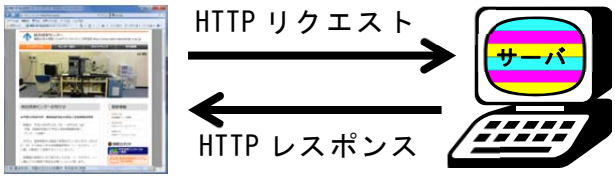

図 2 HTTP のリスエスとレスポンス

情報のやりとりの中で GET メソッドと POST メソッドの使い分けも定義されており，例えば秘密情報を送信する場合は GET メソッドを用いると URL 上に指定されたパラメータが Referer 経由で外部に漏洩するので POST メソッドを用いる。また HTTP 通信の内容を表示させるには Fiddler ソフトウェア経由で通信を行うことで可能である。以下の図に Fiddler の動作画面を示す。

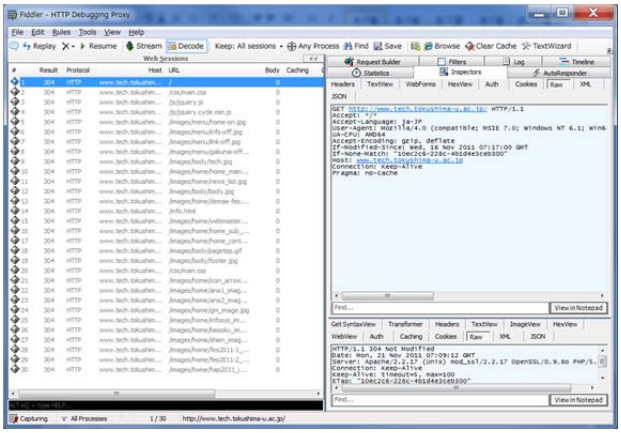

図 3 Fiddler による HTTP 通信の表示

○セッション管理

HTTP 通信の状態を記憶するためにクッキーという仕組みがある。クッキーはサーバ側からブラウザに対して「名前＝変数」の組を覚えておくように指示するものである。いったんクッキー値を覚えたブラウザは，その後おなじサイトにリクエストを送信する際には覚えたクッキー値を送信する。以下に通信の例を示す。

サーバ側：

```
HTTP/1.1 200 OK
Set-Cookie: PHPSESSID=gg35343533344665 path=/
Content-Length: 279
Content-Type: text/html charset=UTF-8
<HTML>
省略
</HTML>
```

サーバ側はブラウザに対してクッキー値を覚えるように指示する。

ブラウザ側：

```
POST /31/31-021.php HTTP/1.1
Referer: http://example.jp/31/31-021.php
Content-Type:application/x-www-form-urlencoded
Host: example.jp
Content-Length: 18
Cookie: PHPSESSID=gg35343533344665
<HTML>
省略
</HTML>
```

ブラウザ側は覚えたクッキー値を使い通信を行う。PHPSESSID のクッキー値はセッション ID と呼ばれ，セッション情報にアクセスするためのキー情報となる。

クッキーは少量のデータをブラウザ側で覚えておけるもので，アプリケーションデータを保持する目的でクッキーそのものにデータを入れない。クッキーの値は保持できる値の個数や文字列長に制限があり，また利用者本人には参照・変更できるので秘密情報の格納には向かない。実際のデータはサーバ側で管理する方法が広く用いられている。

3.2 受動的攻撃と同一生成元ポリシー

Web アプリケーションに対する代表的な攻撃において受動的攻撃と能動的攻撃がある。またブラウザ側で用意されているセキュリティ対策の制限として，JavaScript によるサイトをまたがったアクセスを禁止する同一生成元ポリシーがある。以下に考え方を説明する。

○能動的攻撃と受動的攻撃

能動的攻撃は攻撃者が Web サーバに対して直接攻撃すること。能動攻撃の代表例として SQL インジェクションがる。以下の図に能動的攻撃を示す。

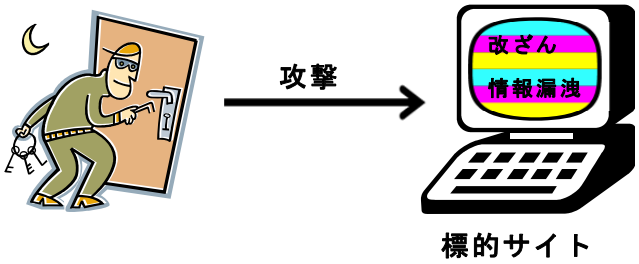

図 4 能動的攻撃のイメージ

受動的攻撃は攻撃者がサーバを直接攻撃するのではなく，Web サイトの利用者に罠を仕掛けることにより，罠を閲覧したユーザを通してアプリケーションを攻撃する手法である。以下の図に受動的攻撃を示す。

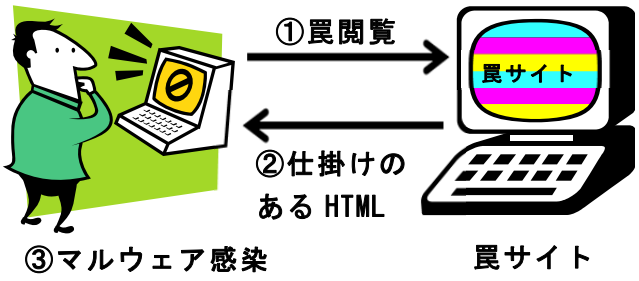

図 5 受動的攻撃のイメージ

対策としてブラウザではサンドボックスという考え方があり，プログラムができることに制約を設け，悪意のあるプログラムが動いても利用者に被害が及ばないように配慮されている。サンドボックスは英語で「砂場」という意味で，砂場で幼児が好きなだけ騒いでも外部に迷惑を及ぼさないことからの連想で，サンドボックスという用語が用いられている。

○同一生成元ポリシー

JavaScript によるサイトをまたがったアクセスを禁止するセキュリティ上の制限であり，ブラウザのサンドボックスに用意された制限 1 つである。同一生成元とは以下の全てを満たす場合である。

- URL のホスト（FQDN）が一致している。
- プロトコルが一致している。
- ポート番号が一致している。

異なるホストに設置された JavaScript にアクセスが出来るとセキュリティ上問題なので同一生成元ポリシーによりアクセスが拒否された。以下に図を示す。

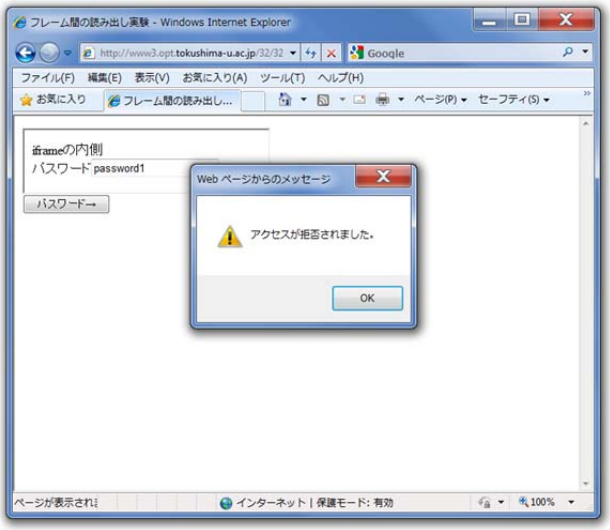

図 6 同一生成元ポリシーアクセス拒否

○まとめ

安全な Web アプリケーションの作り方において，どの情報が漏洩しやすいのか，どの情報が書き換えられるのか，安全に情報を保持するにはどのようにすればよいのかを考えることは重要である。また，近年では受動的攻撃という攻撃手法も増えてきておりサーバ・Web アプリケーション・OS・ブラウザの総合的な知識が必要である。

4.1 WEB アプリケーションの機能と脆弱性の対応（担当：木戸）

WEB アプリケーションの機能とそこに存在する脆弱性について説明をする前に，サーバがどのように構築されるか，またどのように動いているか知っておくほうが理解が深まるため，以下の事を説明した。

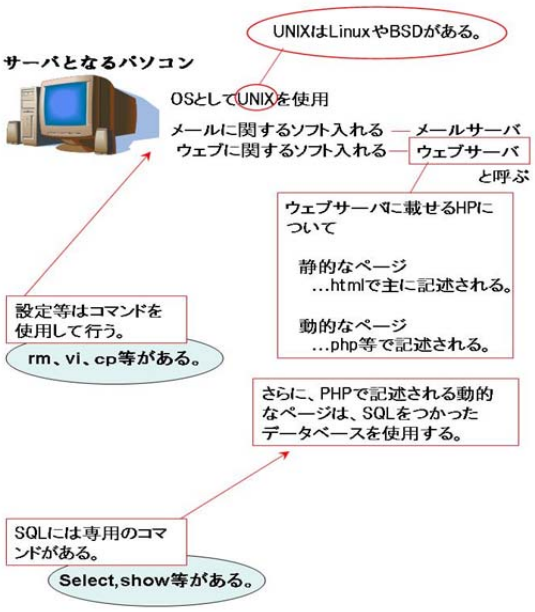

図 7 テキスト使用前に説明したサーバ構築の概念図

- OS のインストールおよびその設定には，コマンドを使用する。
- メールサーバ，およびウェブサーバを構築するためには，必要なソフトをインストールしなければならない。
- ウェブサーバ上に載せるサイトには，動的なホームページと静的なホームページの 2 種類がある。その違い（html と php）を説明し，今回の研修で扱っているものが動的なページであることを把握する。
- 動的なホームページにおいては，SQL などのデータベースを使用する場合がある。SQL を使用する際には，専用のコマンドを使用する場合がある。
- 動的なホームページを作成する場合，PHP 言語等が使用される。

これらの事を理解してもらうために，ホワイトボードに，図 7 の概念図を記述し，説明をした。

この説明を終了したあと，脆弱性はどこで発生するのか参考書に即して列挙した。列挙した項目は，以下のとおりである。

- HTMLの出力（クロスサイト・スクリプティング）
- HTTPヘッダの出力（HTTPヘッダ・インジェクション）
- SQLの呼び出し（SQLインジェクション）
- シェルコマンドの呼び出し（OSコマンド・インジェクション）
- メールヘッダおよび本文の出力(メールヘッダ・インジェクション)

上記のインジェクション系の脆弱性に関する詳細な説明は,別の受講生が説明するので,簡単な紹介にとどめた。

しかし,受講する人々に具体的なイメージを持ってもらうために，SQL インジェクションについては例をあげた。その後，参考書の記述に即して説明した。説明の内容は以下の通りである。

WEB アプリケーションでは，テキスト形式のインターフェースを多用するが，これらのテキスト形式は，決められた文法により構成され，演算子やデータ，命令などが存在している。データはシングルクォートやダブルクォートで囲むか，カンマやタブ，改行で区切られて識別される。多くの WEB アプリケーションでは,テキストの構造を決めておき，そこにデータを入れるようにするが，その際にシングルクォート等を使用することで，データをそこで終わらせ，それ以降の文字列の構造を変化させることができる。その結果，データベースを破壊すること等が可能である。

4.2 入力処理とセキュリティの説明について

動的なホームページにおいては，そのホームページ閲覧者が入力作業を行う事がある。その入力の際に発生する問題について，参考書に即して説明した。説明した内容を，ここに記す。

WEB アプリケーションの入力には，HTTP リクエストとして渡されるパラメータ（GET，POST，クッキー）がある。

入力処理では，入力値に対しては以下の処

理を行う。

- 文字エンコーディングの妥当性検証
- 文字エンコーディングの変換(必要な場合のみ)
- パラメータ文字列の妥当性検証

文字コードという用語には，文字集合と文字エンコーディング（文字符号化方式）の概念が合わされている。文字集合とは，アルファベットや数字といった文字を集めたものであり,ASCIIやUnicodeといったものがある。

文字集合の扱いが原因で起こる脆弱性も存在する。Unicodeのある文字をJIS系の文字集合に変換する際，エスケープが必要な場合がある。このエスケープが必要な場合，処理の順序によってはエスケープ処理が抜けてしまい，脆弱性の原因になることがある。

文字集合には，1バイト文字集合と2バイト文字集合が存在する。これらを併用するための符号化を文字エンコーディングと呼ぶ。この文字エンコーディングには，Shift_JISやEUC_JP等があり，2バイト文字の先行バイトと後続バイトの領域が重なることで，別の文字と認識される問題等がある。

PHPの場合,文字エンコーディングの検証には,mb_check_encoding関数が利用できる。また，文字エンコーディングを変換するにはphp.iniの設定等により，自動変換することができるが，mb_convert_encodingを使用して，明示的に変換することもできる。

文字エンコーディングの自動変換を使用するとプログラミングは楽になるが，以下のデメリットも存在する。

- 文字が化けた場合でも利用者が気づかずに処理を継続してしまう。
- サーバの移転などで，php.iniが変更された場合に,チェックがかからなくなる危険性がある。

文字エンコーディング関連の処理が終わると入力値の検証を行う。しかし，入力値検証はアプリケーションの仕様が基準なので，入力時点では，何も防げない場合がある。そのため，あくまで入力値検証は保険的対策として考えるほうがよい。

入力値を検証する際の基準はアプリケーションの仕様である。例えば，ユーザIDは英数字8文字等，文字種や文字数などの書式を仕様にもとづいて確認する。その際，制御文字のチェックや文字数のチェック等を行うべきである。

入力値検証の実装には正規表現の利用が便利である。PHPの正規表現関数には，pregあるいはmb_eregがある。pregは文字エンコーディングがUTF-8の場合のみ日本語が扱えるが，mb_eregは様々な文字エンコーディングが利用できる。

○まとめ

輪講形式であるため，自分が説明する箇所については，あらかじめ調べておく必要があった。また，調べたことについて，どのように受講する人々に伝えるか，考えておく必要があった。その工夫として，自分が担当した場所だけを，輪講当日に読み上げるだけでなく，サーバ構築についての話もした。結果として,冗長になったという反省はあるものの,受講する人々の理解が深まったという実感を得ることができた。

また，自分がテキストに記述されていることを説明した際に，受講する数名から間違いを指摘されることがあった。プログラム作成等の仕事を通して，ある程度の自負を持っていたが，WEBアプリケーション作成に対しては，未だ経験が浅く，プログラム以前に知っておくべき知識については不十分であることを痛感した。しかし，間違いを指摘される際に，難詰するようなことはなかったため，自由に議論することができた。

テキストを読むことによって知識を得るだけではなく，輪講形式で他人に説明し，議論することで，説明することの難しさを学ぶことができたのは，非常に有意義であった。

今後とも，分野内研修に限らず，仕事においても積極的に議論を交わし，自らの知識を深めていきたい。

4.3　表示処理に伴う問題（担当：片岡）

4.3.1　クロスサイトスクリプト（XSS）

出力をブラウザ上で表示する際に特殊な意味を持つ文字の処理が不十分なときに発生する。一般に XSS と略される。
対策はエスケープ処理をすることが主である。

HTML で特殊な文字として使用される文字を列挙すると,「<」(>),「>」(<) ,「&」(&) ,「”」(") ,「’」(')があります。エスケープして変換しなければならない文字列を括弧で表示する。参考までに,「&」を使用した文字としては空白（ ）などもある。同じ文字・文字列でもパーセントエンコーディングや 16 進表記など複数の表示手段があることに注意が必要になる。

同様に JavaScript で特殊な文字として対応しなければならない文字を列挙すると,「¥」(¥¥),「‘ 」(¥’) ,「“ 」(¥”) ,改行 (¥n)がある。

表示処理に関する場所・方法によって必要とするエスケープ処理等の処理が異なるのでそれぞれについて述べていく。同様の処理については重複を省くため略している。

〇要素内容

HTML 文字参照をエスケープ（「<」「&」）しなければならない。PHP においては htmlspecialchars 関数を使用する。第 2 引数は「ENT_NOQUOTES」で十分です。また，第 3 引数のエンコードは必ず指定する必要がある。

〇属性値

フォームの属性値はダブルクオーテーションで必ず囲み，その上で HTML 文字参照をエスケープ（「<」「”」「&」）しなければいけない。PHP においては，htmlspecialchars 関数を使用します。第 2 引数は「ENT_COMPAT」（下位互換）もしくは「ENT_QUOTES」が必要である。

〇属性値（URL）

「javascript:JavaScript 式」という形式の JavaScript スキームに注意が必要となり，URL 形式検査をします。その後に，通常の属性値としてエスケープ処理を行う。参考図書に掲載されたサンプル関数を図 8 に記述する。外部ドメインに対するリンクが自明な場合以外は以下の対処をする。①リンク先 URL が外部ドメインである場合はエラーにする。②注意を喚起するクッションページを表示する。

```
function check_url($url){
  if(preg_match('/¥Ahttp:/',$url) ||
    preg_match('/¥Ahttps:/',$url) ||
    preg_match('#¥A/#',$url)){
    return true;
  }else{
    return false;
  }
}
```

図 8 http:または https で始まる絶対 URL または　スラッシュ（/）で始まる相対 URL（絶対パス参照）をチェックする関数

〇イベントハンドラ

イベントハンドラ内の文字列リテラルを適切に処理する。まず，JavaScript としてのエスケープをする。（JavaScript としてのエスケープをした後で）通常の属性値としてエスケープし，値はダブルクオーテーションで必ず囲む。参考図書に掲載されたサンプル関数を図 9 に記述する。

```
function escape_js($s) {
  return mb_ereg_replace('([¥¥¥¥'"])', '¥¥¥1', $s);
}
<script>
  $('#name').text(
      '<?php echo escape_js($_GET['name']); ?>');
</script>
```

図 9 JavaScript としてのエスケープ

〇スクリプト要素内の文字列リテラル

JavaScript としてのエスケープをして，さらに終端の目印になっている「</」をリテラルに入れない。一番の対策は JavaScript に動的なパラメータを渡さない。渡さなければならない場合の対策が二つある。

① Unicode エスケープ

英数字以外（マイナスとピリオドも）をすべてエスケープする。JavaScript の¥uXXXX 形式の読み込みを利用する。参考図書に掲載されたサンプルを図 10 に記述する。

```
<?php
// 文字列を全て¥uXXXX 形式に変換する
function unicode_escape($matches) {
  $u16 = mb_convert_encoding($matches[0],
 'UTF-16');
  return preg_replace('/[0-9a-f]{4}/' , '¥u$0',
bin2hex($u16));
}
// 英数字以外を¥uXXXX 形式でエスケープする
function escape_js_string($s) {
  return preg_replace_callback('/[^-¥.0-9a-z]+/u',
'unicode_escape', $s);
}
?>
【呼び出し例】
<script>
  alert('<?php echo escape_js_string('吉 and 吉'); ?>');
</script>
```

図 10 Unicode エスケープ

②スクリプト要素の外部でパラメータ（例えば hidden で）を定義して JavaScript から参照する。参考図書に掲載されたサンプルを図 11 に記述する。

```
<input type="hidden" id="familyname" value="
    <?php echo htmlspecialchars($_GET['name'],
                              ENT_COMPAT,
'UTF-8'); ?>">
<script src="jquery-1.4.4.min.js"></script>
こんにちは<span id="name"></span>さん
<script>
  var familyname =

document.getElementById('familyname').value;
  $('#name').text(familyname);
</script>
```

図 11 外部パラメータで定義

〇DOM based XSS 対策

JavaScript で HTML に(document.write 関数で)文字を出力する際にエスケープがされていないことから起こる。JavaScript 標準関数にエスケープ機能がないので Jquery などのライブラリを使用する。

〇保険的対策

①入力値の妥当性検証をする。　参考までに，HTML や PHP タグを取り除くには strip_tags()が PHP にある。

②クッキーの httponly 属性を付与する。

③Apache で Trace メソッドを（XST 対策として）無効にする。ただし，XST については最近の主要なブラウザでは対策済みである。

（httpd.conf）TraceEnable OFF

4.3.2　エラー処理

エラーメッセージは攻撃のヒントを与えるので極力，差し障りのない利用者向けのメッセージを表示し，詳細な内容はエラーログに出力するようにする。

（PHP.ini）display_errors=Off

プログラム上での Tips として，出力する PHP エラーの種類を設定する関数である error_reporting を使うと良い。

「error_reporting(0);」とすることですべてのエラー表示をオフにできる。

4.4　SQL 呼び出しに伴う脆弱性

（担当：藤田）

現在 Web アプリケーションの多くはデータベースアクセスのために SQL を利用している。データベースアクセスの実装に不備がある Web アプリケーションを作成した場合，SQL インジェクションと呼ばれる脆弱性が生じる。この脆弱性を利用することで，サーバに対して以下のような攻撃を受ける可能性がある。

1) データベース内の全ての情報が外部から盗まれる
2) データベースの内容が書き換えられる
3) 認証を回避される（ID とパスワードを用いずにログインされる）
4) その他，データベースサーバ上のファイルの呼び出し，書き込み，プログラムの実行などを行われる

これらの攻撃がもたらすサーバへの影響力は非常に大きいため，アプリケーション開発において SQL インジェクションの脆弱性が絶対に生じないプログラミングをする必要がある。

〇SQL インジェクション

SQL インジェクションとは，ユーザが Web アプリケーションの SQL 呼び出しを行う所に不正な命令を混入させる事で発生します。不正な命令とは，Web アプリケーション内部にて実行される SQL 文に追加のコマンド（この命令よりサーバ内情報の読み出し，書き込み，削除などの攻撃を行うことが出来ます)，条件が常に真となるような条件式（この命令より，アプリケーションの認証を回避することが出来ます）を挿入することである。

○対策

SQLインジェクション脆弱性の根本原因は，与えられた引数により，Web アプリケーション内の SQL 文が変更されることです。そのため，SQL インジェクション脆弱性は，SQL 文を構築する際に SQL 文が変更を防ぐことにより解消することが出来る。その方法とは，以下の通りである。

1) プレースホルダを用いて SQL 文を構築する
2) アプリケーション側で SQL 文を構築する際に，SQL 文が変更されないようにする

ただし，2)の方法は，全ての SQL インジェクションに対して完全な対応が求められるため実装することは困難であり，通常は 1)の方法を用いる。

○プレースホルダ

プレースホルダとは，SQL 文中に変数や式など可変のパラメータが挿入される場所を確保することある。プレースホルダは，SQL 呼び出しライブラリより実装することが出来る。また，プレースホルダには，「静的プレースホルダ」と「動的プレースホルダ」の２種類の実装方法がある。どちらの方法を実装しても SQL インジェクションを防ぐことが出来ますが，原理的に SQL インジェクションの可能性がないという点で静的プレースホルダの方が優れている。２種類のプレースホルダの違いについては以下の通りである。

1) 静的プレースホルダ

Web アプリケーションより与えられた値を，データベースエンジン側に渡し，データベース内で SQL 文を構築し，SQL 文を実行する。

2) 動的プレースホルダ

Web アプリケーション内で SQL 文を構築し，構築した SQL 文をデータベース側に渡して SQL 文を実行する。Web アプリケーション内での処理にバグがある場合，SQL インジェクションが発生する可能性がある。

○まとめ

SQL インジェクションは，サーバに対して非常に脅威ではあるが，プレースホルダを実装することで比較的容易に防ぐことが出来る。ただし，プレースホルダ自体は用いるライブラリに依存しているため，現在使用している方法が永続して安全であるとは言えない。そのため，ソフトの更新，新たなる脅威に対する対策手法の学習など常に新しい動向に応じ，対策を心がける必要がある。

4.5 「重要な処理」の際に混入する脆弱性（担当：片岡）

4.5.1 クロスサイト リクエスト フォージェリ（CSRF）

重要な処理に対するリクエストが正規利用者の意図したものであることを確認する必要がある。その対策として CSRF 対策の必要なページ（登録や課金が発生するなど取り消しできない重要な処理）を区別する。それは Web アプリケーションの設計の段階から（要件定義工程）考慮すると良い。

正規利用者の意図したものであると確認する方法を列挙していく。

○秘密情報（トークンの）埋め込み

一般的な対策方法で推奨される。hidden でトークン（例えばセッション ID）を埋め込み，POST でおくる。注意点としては実行ページの直前に埋め込むことである。
参考図書に掲載されたサンプルを図 12 に記述する。

```
<form method=”POST”>
<input type=”hidden” name=”token” value=”
      <?php echo htmlspecialchars(session_id(),
      ENT_COMPAT,’UTF-8’); ?>”>
 …
 確認で if(session_id!==$_POST[‘token’])…
```

図 12 トークンの埋め込み

〇パスワード再入力

パスワード入力の手間は増えるのが欠点である。

〇Referer のチェック

$_SERVER[‘HTTP_REFERER’]が直前のページであることを確認する。

しかし，古い携帯電話など Referer がオフにしている使用者には使えない。したがって利用者の環境が限定される場合に良い。

〇保険的な対策

重要な処理後には登録済みメールアドレスに通知メールを出すなどすると良い。

4.6 セッション管理の不備（担当:山中）

Web アプリケーションでは，ステートレスな HTTP のために,セッション管理が用いられる。このセッション管理に関する脆弱性について,テキストの「4.6　セッション管理の不備」を用いて学習を行った。

4.6.1 セッションハイジャック

第３者にセッションIDを知られてしまうと，利用者に成りすますことが可能になってしまう。これをセッションハイジャックという。

セッションIDを知るための方法として，

・セッションIDの推測

・セッションIDの盗み出し

・セッションIDの強制（固定化）

がある。

4.6.2 セッション ID の推測を防ぐ

セッション ID を作り出す処理を自作してしまうと，脆弱性を作りこんでしまう可能性がある。自作せずに，実績のあるプログラム言語などによるセッション管理用のライブラリなどを使うようにしないといけない。

4.6.3 セッション ID の盗み出しを防ぐ

セッション ID を盗まれてしまう原因の一つとしてURL埋め込みのセッションIDがある。

```
http://example.jp/mail/123?SESSID=2F3BE9A31F900
```

図 13 URL 埋め込みのセッション ID の例

これを使っていると,Referer経由でセッションIDが第３者に知られてしまう場合がある。

これを防ぐため，セッションIDはクッキーに保存するようにする。PHPの場合なら，次のように設定するとよい。

```
[Session]
session.use_cokies = 1
sesion_.use_only_cookies = 1
```

図14 PHPでセッションIDを
クッキーに保存する設定

4.6.4 セッション ID の固定化を防ぐ

セッションIDの固定化とは，外部からセッションIDを強制し，利用者にむりやり使わせる方法である。

これへの対処として,「認証後にセッションIDを変更する」方法がある。PHPでこの処理をするためには,session_regenerate_id関数を使う。

```
bool session_regenerate_id([bool $delete_old_session = false])
```

図15 セッションIDを変更する関数

使い方は，認証の処理が終わった時点で，この関数を呼び出せばよい。

※セッション ID の変更ができない場合

Web アプリケーションに用いるプログラム言語によっては､セッション ID の変更をできないものがある。そのような場合は、トークンと呼ばれる乱数文字列を使用する。

トークンは予測ができないような乱数（/dev/urandomなど）を使って生成する。

使い方は、前述のセッションIDを変更する関数と同じように、認証の処理が終わったところで、トークンを作り、セッションIDと同じように扱うことになる。

4.6.5 ログイン前のセッションIDの固定化を防ぐ

ログイン前にもセッション ID の固定化の危険があり，これに完全に対処するのは難しいため，

・ログイン前のセッション変数には秘密情報を保存しないようにする。

・URL 埋め込みのセッション ID を使わない

・地域型ドメインを使わない

という対処を組み合わせる。

4.7 リダイレクト処理にまつわる脆弱性（担当：菊地）

リダイレクトとは，あるファイルやドメインにアクセスしたときに，別ファイルやドメインにアクセスするように仕向けるものであり，サイト移転した際などに使用される。Web アプリケーションでのリダイレクト処理に際して発生する脆弱性について説明する。

4.7.1 オープンリダイレクタ脆弱性

Web アプリケーションでは，外部から指定した URL に直接リダイレクトすることがあり，詳細なログを取る時などに使用されている。そのように任意の URL にリダイレクトできる機能をオープンリダイレクタと呼ぶが，その処理には脆弱性が含まれていることがあり，オープンリダイレクタ脆弱性と呼ぶ。脆弱性を放置すると，他のドメインにリダイレクトされフィッシング詐欺の踏み台とされる可能性もあり，実際に悪用された例もある。脆弱性の原因は，リダイレクト先の URL を外部から指定できること，また，リダイレクト先のドメインのチェックがないことである。その対策としては，次のいずれかを実行する必要がある。

・リダイレクト先の URL を固定にする。
・URL を直接指定せず番号指定にする。
・ドメインをチェックする。

4.7.2 HTTP ヘッダ・インジェクション

Web アプリケーションの中には，リクエストに対して出力する HTTP レスポンスヘッダを，外部からのパラメータを元に生成するものがある。HTTP ヘッダ・インジェクションは，リダイレクト処理（Location）以外にクッキー出力（Set-Cookie）など，すべての HTTP レスポンスヘッダ出力処理で発生する恐れのある脆弱性である。

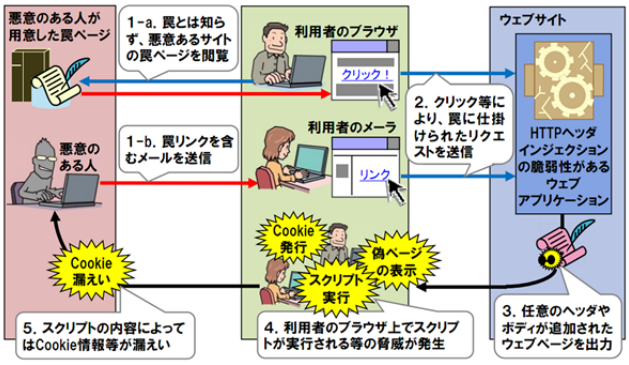

図 16 HTTP ヘッダ・インジェクション（「安全なウェブサイトの作り方」IPA より）

ヘッダ・インジェクションの攻撃手法は，任意のレスポンスヘッダの追加やレスポンスボディの偽造である。その攻撃により，任意のクッキー生成，外部ドメインへのリダイレクト，表示内容の改変，JavaScript 実行による XSS と同様の被害などを受けることになる。HTTP レスポンスヘッダはテキスト形式で記述され，１行で１つのヘッダを定義する。ヘッダの記述は改行で終了し，それ以降は新たなヘッダとして処理される。脆弱性は，入力値を出力に用いている箇所において，文法上特殊な意味を持つ文字（改行）をエスケープせずに用いることにより生じる。

改行の後に Set-Cookie ヘッダを挿入することで，任意の値のクッキーを設定させることができる。Web アプリケーションのセッション ID は通常，クッキーで管理されており，固定値で上書きできればセッション固定攻撃を実現できる。セッション固定攻撃はセッションハイジャックを容易にする。

連続する改行を挿入すると，HTTP ヘッダの終了およびボディの開始を指示することができる。連続する改行の後の内容はボディの先頭部分として処理されるため，ボディに任意の記述を挿入できることになる。これはクロスサイト・スクリプティング等に繋がる可能性がある。

脆弱性への最も効果的な対策は，外部からのパラメータを HTTP レスポンスヘッダとして出力しないことである。ただ，上記の対策がとれない場合には，以下の対策を実施する必要がある。

・リダイレクトやクッキー出力を専用 API にまかせる。
・ヘッダ生成するパラメータの改行文字をチェックする。

4.7.3 HTTP レスポンス分割

HTTP レスポンス分割攻撃とは，HTTP ヘッダ・インジェクション攻撃により複数の HTTP レスポンスを作り，プロキシ（キャッシュサーバ）を汚染するという攻撃手法である。この攻撃は，ウェブページの改ざんと同

じ脅威であり，この攻撃を受けたウェブサイトにアクセスする利用者は，この差し替えられた偽のウェブページを参照し続けることになる。

4.7.4 まとめ

リダイレクト処理に関わる脆弱性への対策として，リダイレクト処理にはできるだけ専用APIを利用する,リダイレクト先は固定する,もし固定できない場合には外部から使用するリダイレクト先のURLは，必ず文字種とドメイン名をチェックする必要がある。また，外部からのパラメータをHTTPレスポンスヘッダとして出力しないなどの対策を取る，必要がある。

4.8 クッキー出力にまつわる脆弱性

クッキーはサーバと利用者間の状態を管理するために使われ,ステートレスなHTTPプロトコルにおいて，ステートフルな状態を実現することができる。しかし，クッキーの扱い方により脆弱性が発生する場合がある。

4.8.1 クッキーの不適切な利用

クッキーにデータを保存することにより脆弱性が生ずることがある。セッション変数は外部から書き換えができないが，クッキー値はアプリケーション利用者によって書き換えできる。たとえば，ユーザIDや権限情報など，書き換えられると困る情報をクッキーに保存すると，脆弱性が発生する。そのため，クッキーでは実現できるが，セッション変数では実現できない場合のみ，クッキーに情報を保存し，その他の場合にはセッション変数を利用する。

4.8.2 クッキーのセキュア属性不備

ブラウザは，クッキーをセットされたサーバへアクセスするたび（domain と path が一致した場合にのみ）に，そのクッキー情報をサーバへ送る 。クッキーにセキュア属性を付けた場合には，HTTPS通信（暗号化されている）でのみ送信するが，セキュア属性を付けない場合には，HTTPS通信でも平文で送信する事があるので，盗聴される危険性がある。クッキーを盗聴されると成りすましの被害を受けることがある。

クッキーのセキュア属性不備への対策は，クッキーのセキュア属性を設定することである。ただ，HTTPとHTTPSの混在するサイトでは，セッションIDに対してセキュア属性を設定するとアプリケーションが動かなくなる場合があるので，トークンをセキュア属性つきクッキーとして発行し，ページごとに確認する。すなわち,トークンを使用すると，サーバとブラウザの双方向で確実に暗号化されること，HTTPSのページを閲覧するには第三者が知り得ないトークンが必要であることから，安全性が確保される。

4.8.3 クッキーのセキュア属性以外の属性値について Domain 属性

クッキーにはセキュア属性以外にも属性があり，セキュリティ上の影響がある。

〇Domain 属性

Domain属性はデフォルト状態が最も安全な状態である。ただし，複数のサーバでクッキーを共有する場合にDomain属性を指定する場合がある。

〇Path 属性

ディレクトリごとに異なるセッションIDを発行する場合には，Path属性を指定する。

〇Expires 属性（自動ログインなど）

セッションIDのクッキーはExpires属性をつけずにブラウザ終了と同時にクッキーが削除されるようにするが，自動ログインなどを行いたい場合には，この属性を設定する。

〇HttpOnly 属性

HttpOnly属性をつけたクッキーは，クライアント側のスクリプトからアクセスできなくなる。この属性は，クロスサイト・スクリプティング攻撃の影響を軽減する効果がある。

4.8.4 まとめ

原則として，クッキーはセッションIDのみに用いること，HTTPS通信を用いるアプリケーションのクッキーにはセキュア属性を指定することが重要である。

4.9 メール送信の問題（担当：石井）

メール送信問題について以下の3つがある。

・メールヘッダ・インジェクション脆弱性
・hiddenパラメータによる宛先保持
・メールサーバによる第三者中継（参考）

このうち，メールサーバによる第三者中継

は，アプリケーションが原因で脆弱性が起こらないので，参考として説明した。

hidden パラメータによる宛先保持は，hidden パラメータの送信先アドレスを任意に変更し，迷惑メール送信に利用される。対策は，送信先メールアドレスを hidden パラメータに保持するのではなく，ソースコードにハードコーディングするか，サーバ上の安全な場所に保持する。

メールヘッダ・インジェクション脆弱性は，改行文字を使ってメールヘッダや本文を追加・変更する手法である。メールヘッダ・インジェクション脆弱性による影響は次の 3 つがある。

・件名や送信元，本文を改変される
・迷惑メールの送信に悪用される
・ウイルスメールの送信に悪用される

この脆弱性に対する対策は以下の 3 点である。

・メール送信には専用のライブラリを使用する
・外部からのパラメータをメールヘッダに含ませないようにする
・外部からのパラメータには改行を含まないようにメール送信時にチェックする

この章では，メール送信の問題に関する脆弱性とその対策について説明した。

4.10 ファイルアクセスにまつわる問題

4.10.1 ディレクトリ・トラバーサル脆弱性

ファイル名に対するチェック不足によって起こる脆弱性のことを，ディレクトリ・トラバーサル脆弱性という。この脆弱性による影響は，以下の 2 つがある。

・Web サーバ内のファイル閲覧
・Web サーバ内のファイル改ざん，削除

次にディレクトリ・トラバーサル脆弱性が生まれる原因として以下の 3 つがある。

・ファイル名を外部から指定できる
・ファイル名として，絶対パスや相対パスの形で異なるディレクトリを指定できる
・組み立てたファイル名に対するアクセスの可否をチェックしていない

上記の 3 つの条件がすべて満たされないとディレクトリ・トラバーサル脆弱性にはならないので，上記のいずれかをなくせば，解消できる。この脆弱性に対する対策は，以下の 3 つである。

・外部からファイル名を指定できる仕様を避ける
・ファイル名にディレクトリ名が含まれないようにする
・ファイル名を英数字に限定する

このうち，外部からファイル名を指定できる仕様を避ける方法が根本的に解決できる。具体的には，ファイル名を固定する。

4.10.2 意図しないファイル公開

非公開ファイルを公開ディレクトリに置いたことによって起こる脆弱性である。この脆弱性が生まれる原因として以下の 3 つがある。

・ファイルが公開ディレクトリに置かれている
・ファイルに対して URL を知る手段がある
・ファイルに対するアクセス制限が掛かっていない

この脆弱性に対する対策は以下の 2 つがある。

・非公開ファイルを公開ディレクトリに置かない
・保険的にディレクトリ・リスティングを無効にする

この章では，ファイルアクセスにまつわる問題に関する脆弱性とその対策について説明した。

4.11 OS コマンド呼出しの際に発生する脆弱性

シェル呼び出しの際に問題があると，意図しない OS コマンドが実行可能になることがある。これを OS コマンド・インジェクション脆弱性という。OS コマンド・インジェクション脆弱性による影響は次の 3 つである。

・Web サーバ内のファイルの閲覧・改ざん削除
・外部へのメール送信
・別サーバへの攻撃（踏み台）

次に OS コマンド・インジェクション脆弱

性が生まれる原因として以下の 3 つがある。
・シェルを呼び出す機能のある関数の利用
・シェル呼び出しの機能のある関数にパラメータを渡している
・パラメータ内に含まれるシェルのメタ文字をエスケープしていない

OS コマンド・インジェクション脆弱性が発生するには，上記の 3 つの条件をすべて満たすことなので，上記のいずれかをなくせば、解消できる。この脆弱性に対する対策は，以下の 4 つである。
・OS コマンド呼び出しを使用しない実装方法の選択
・シェル呼び出し機能のある関数の利用を避ける
・外部から入力された文字列をコマンドラインのパラメータに渡さない
・OS コマンドに渡すパラメータを安全な関数によりエスケープする

上記の対策をしっかりと行えば，この脆弱性は解消できるが，対策に漏れがあった場合，被害の影響が大きいので，攻撃の被害を軽減する保険対策を実施することが必要である。この保険的対策は以下の 3 つである。
・パラメータの検証
→ファイル名を英数字に限定
・アプリケーションの稼働する権限を最小限にする
→コマンドの実行権限＝アプリケーションの持つ権限よって必要最小限にしておくと，被害が少なくなる
・Web サーバの OS やミドルウェアのパッチ適用
→サーバ内部からの攻撃を受けた場合，最悪 root 権限を奪われるので，それを避けるため，パッチを適用する。OS のバージョンを最新のものにしておく

この章では，OS コマンド呼出しの際に発生する脆弱性とその対策について説明した。

4.12 ファイルアップロードにまつわる問題（担当：井上）

Webアプリケーションにおいて，アップローダやダウンローダに発生する脆弱性がある。

4.12.1 ファイルアップロードの問題と概要

アップローダに対する攻撃は以下のものがある。

a. アップロード機能に対するDoS攻撃

アップロード機能に巨大なファイルを連続して送信することで，Webサイトに過大な負荷を掛けるDoS攻撃。

影響：応答速度の低下，サーバの停止

対策：アップロードファイルの容量制限

b. サーバ上のファイルをスクリプトとして実行する攻撃

アップロードファイルがWebサーバの公開ディレクトリに保存される場合,外部からアップロードしたスクリプトファイルがWebサーバ上で実行される。

影響:OSコマンド・インジェクション攻撃

対策：4.12.2で示す

c. 仕掛けを含むファイルを利用者にダウンロードさせる攻撃

仕掛けを含ませたファイルを攻撃者がアップロードするもので,利用者がファイルを閲覧すると,利用者のPC上でJavaScriptの実行やマルウェア感染が起こされる。

影響:利用者PCでのJavaScript実行やマルウェア感染

対策：ウィルスソフトの導入

d. ファイルの権限を超えたダウンロード

限られた利用者のみがダウンロードできるファイルに対するアクセス制限緩いとき場合，URLの推測により，権限のない利用者までダウンロードできる。

対策：認可制度の充実

4.12.2 アップロードファイルによるサーバ側スクリプト実行

アップローダの中には利用者がアップロードしたファイルをWebサーバの公開ディレクトリに保存するものがあり，また，ファイル名の拡張子として,php,asp,aspx,jspなどスクリプト言語の拡張子が指定できると,ア

ップロードしたファイルをスクリプトとしてWebサーバ上で実行できる。外部から送り込んだスクリプトが実行され，OSコマンド・インジェクション攻撃と同じ影響がでる。

影響：Webサーバ内のファイルの閲覧・改ざん・削除，外部へのメール送信，別のサーバへの攻撃（踏み台）

対策:利用者にアップロードされたファイルを公開ディレクトリには置かず，スクリプト経由で閲覧させる。また，拡張子をスクリプト実行の可能性のないものに制限する。

4.12.3 ファイルダウンロードによるクロスサイト・スクリプティング(XSS)

アップロードしたファイルを利用者がダウンロードする際に,ブラウザがファイルタイプを誤認する場合がある。画像データ中にHTMLタグが含まれていると,条件によってはブブラウザがHTMLファイルと誤認識し,画像ファイル中に埋め込まれたJavaScriptを実行する場合がある。この脆弱性を悪用する攻撃者は,HTMLやJavaScriptを仕込んだ画像ファイルやPDFファイルをアップロードして公開する。通常の参照の仕方ではHTMLとは認識されないが,攻撃者がアプリケーション利用者に罠を仕掛け,HTMLと認識するように仕向けることでXSS攻撃が成立してしまう。

対策：

a)ファイルのcontent-Typeを正しく設定する

b)画像の拡張子と画像の中身が対応していることを確認する。

c)ダウンロードを想定したファイルにはレスポンスヘッダとして，[Content - Disposition:attachment]を指定する。

4.13 インクルードにまつわる問題

4.13.1 ファイルインクルード攻撃

PHPなどのスクリプト言語にはスクリプトのソースの一部を別ファイルから読み込む機能がある。Includeのどに指定するファイル名を外部から指定できる場合,アプリケーションが意図市内ファイルを指定することにより脆弱となることがある。これをファイルインクルード脆弱性と呼ぶ。PHPの場合，設定によっては外部サーバのURLをファイル名として指定できる場合がある。これをリモート・ファイルインクルード(RFI)と呼ぶ。

ファイルインクルード攻撃による影響：

a. Webサーバ内のファイルの閲覧による情報漏洩

b. 陰萎スクリプトの実行による影響
＝＞サイト改ざん，不正な機能実行，
他サイトへの攻撃（踏み台）

ファイルインクルード攻撃への対策：

a. インクルードするパス名に外部からのパラメータを含めない。

b. インクルードするパス名に外部からのパラメータを含める場合は，英数字に限定する。

5 分野別研修（担当：岡山）

今や PC や携帯はメールやインターネットで利用しない日はないほど，身近なツールになって久しい。私も調べたいことを検索する，ネットショッピングを利用する，チケットを予約するなど便利に使っているが，一方で知らないうちにトラブルに巻き込まれるのでは，と不安も抱えている。

少しでも IT の仕組みがわかればと興味本位で申し込んでしまったが，分厚いテキストが届いた時にはかなり後悔した。本を開いても目次の段階で早くもわからない言葉だらけ。ネットで用語を調べると説明文にも謎の言葉が。なかなか先に進まない。

輪読形式だったが，発表者に質問しやすい環境だったので，補足説明を入れてもらったりしながらなんとか最後まで参加できた。

受講するうち，パスワードに「英数字のみ」の指示があること，ログイン後でも「レジに進む」時に ID とパスワードを再度入力する必要がある理由や，メールの中の URL を安易にクリックしてはいけないといわれるのはなぜか，というようなことも分かってきた。

今回初めて，他の分野の分野別研修に参加したが、普段あまり話すことのない方々の話を聞くことができ、業務のご苦労を知

ることができただけでも参加してよかったと思う。以下に簡単なイントロダクションを述べる。

Web セキュリティの基礎

○脆弱性をなくす

Web アプリケーションに脆弱性があると，機密情報の漏えいや内容の書き換え，閲覧者のＰＣをウィルス感染させる等が起こる可能性があり，管理者，利用者ともに被害を受ける。開発者は悪用されないよう脆弱性のない安心なアプリケーションを作らなくてはならない。

○HTTP とセッション管理

HTTP とは WWW サービスにおいて，サーバとクライアント間で通信を行うために使用するネットワークプロトコル(決まり事)。ステートレス(1 回きり)のやり取りのため，ユーザが複数回アクセスしてもサーバは特定のユーザと認識しない。

Web サイトに，アクセスするユーザを特定する，ユーザに関する情報をサーバ側で保持することをセッション管理という。

セッション管理のための識別コードを「セッション ID」といい，通常クッキーとしてブラウザに記憶され，サイトにアクセスするときにはブラウザが自動的に送信している。サイト側のサーバが ID を振り分けるので，推測，漏洩などにより第三者が他の利用者に成りすましができないよう管理が必要である。

○受動的攻撃と同一生成元ポリシー

受動的攻撃とは，攻撃者がサーバを直接攻撃するのではなく，Web サイトの利用者を罠サイトに誘導し，罠を閲覧した利用者を通してアプリケーションを攻撃する手法。（フィッシング詐欺など）

同一生成元ポリシーとは，JavaScript によりサイトをまたがったアクセスを禁止するセキュリティ上の制限。元の URL と同じホストへの問い合わせしかできない。
しかし，ブラウザや Web アプリケーションに脆弱性があるとこれも回避され，攻撃されることになる。

最近の攻撃は不特定多数ではなく、ターゲットを絞って攻撃してくるものも多いらしい。思わぬところからほころびが生まれ穴が開いてしまうのだろう。

またセキュリティ重視で何重にも壁を作ると安全ではあるが使いにくいアプリケーションになり，悩みどころである。

6 分野研修最終

研修の最終日には，情報システム統一研修の報告と情報セキュリティ関連のテストを行った。最後に IPA が公開している「2011 年版 10 大脅威 進化する攻撃 その対策で十分ですか？」について実例を取り入れながら全員で確認した。以下に 10 大脅威を示す。

第 1 位	「人」が起こしてしまう情報漏洩
第 2 位	止まらない！ウェブサイトを経由した攻撃
第 3 位	定番ソフトウェアの脆弱性を狙った攻撃
第 4 位	狙われだしたスマートフォン
第 5 位	複数の攻撃を組み合わせた「新しいタイプの攻撃」
第 6 位	セキュリティ対策不備がもたらすトラブル
第 7 位	携帯電話向けウェブサイトのセキュリティ
第 8 位	攻撃に気づけない標的型攻撃
第 9 位	クラウド・コンピューティングのセキュリティ
第 10 位	ミニブログサービスや SNS の利用者を狙った攻撃

図 17 IPA が公開している「2011 年版 10 大脅威 進化する攻撃 その対策で十分ですか？」

6.1 人が起こしてしまう脅威

10 大脅威において，第 1 位（「人」が起こしてしまう情報漏洩）は毎年，上位に挙がる脅威の一つである。インターネット経由や USB 等からの情報漏洩は予防が困難であり，操作ミスや置き忘れなどの人が起

こしてしまう脅威である。セキュリティポリシを守る以外に予防する方法がない。

6.2 新技術の脆弱性

第 4 位（狙われだしたスマートフォン）や第 9 位（クラウド・コンピューティングのセキュリティ）については新しい技術の中から生まれた脅威の一つであり，新しい技術は攻撃のターゲットになりやすい可能性がある。脆弱性も新しいものであり，対策の方法も確立されていないものが多く，未確認の脆弱性も存在する。新しいファームウェアを利用することが大切である。またスマートフォンにおいてはキャリア IQ（携帯電話からのデータを分析して提供するソフトウェア）の問題も浮上しており新技術には利用上の注意も必要である。

6.3 標的型攻撃

第 8 位（攻撃に気づけない標的型攻撃）は，ある特定の場所がターゲットになるために発見が困難であり推測がつきにくい為に予防も困難である。国内外を問わず世界中からの攻撃に備えなければならないサイバー攻撃の一つである。今年度も日本の企業や政府機関が標的型攻撃を受け，情報の一部が流出したとのニュースが複数報じられた。知らない人からのメールや添付ファイルは開封しないといった基本的なセキュリティルールを守ることが大切である。

6.4 Web サイトを経由した攻撃

第 2 位に挙げられる「止まらない！ウェブサイトを経由した攻撃」は近年に増加している脅威の一つである。Web サイトの攻撃を止めるのは不可能に近く，困難である。いかに Web アプリケーションの脆弱性が少ないサイトを運用するかが重要である。

今回の分野研修において行った受動的攻撃手段である XSS（クロスサイト・スクリプティング）や SQL インジェクションの脆弱性対策などについて，大変有効であることがわかった。また安全な Web アプリケーションを作成及び運用することで重大な脅威から回避できることがわかった。

7 まとめ

今回の研修は，参考書「安全な Web アプリケーションの作り方」を使って輪講形式で輪読・議論を行うことにより、Web アプリケーションの脆弱性だけではなく、ネットワーク全般の攻撃と防御の知識を得る。また脆弱性の確認や、それに対する防御の仕組みを具体的に学ぶことであった。

参考書の第 4 章（Web アプリケーションの機能別に見るセキュリティバグ）を中心に輪講形式で行った。業務上で経験している目線から意見を出し合い，活発な情報交換や議論ができた。また Web アプリケーションで発生する脆弱性や対応方法も，実際のデモを通して確認できた。

Web アプリケーションには専門用語やよく似た意味の単語が多く，前半は戸惑いが生じたが，後半は少し克服できた感じを持った。しかし Web サイトや Web アプリケーションへの攻撃は増加することが予想されるので継続した学習が必要である。

8 謝辞

分野研修受講に際し情報システム技術分野のスタッフや多くの分野から御参加を頂き，大変有意義な時間を過ごすことができここに謝意を表します。

参考文献

1.「安全な Web アプリケーションの作り方 -脆弱性が生まれる原理と対策の実践-」徳丸浩 SoftBankCreative

2.「安全なウェブサイトの作り方　改訂第５版」独立行政法人情報処理推進機構セキュリティセンター

3.「2011 年版 10 大脅威 進化する攻撃... その対策で十分ですか？」独立行政法人情報処理推進機構セキュリティセンター http://www.ipa.go.jp/about/press/20110324.html