PHP を用いたセキュアな学習環境～安全な文字入力の構築～

Secure Learning Environment with PHP

河地 裕介† 江見 圭司††
Yusuke KAWACHI† Keiji EMI††
†大阪大学 ††京都情報大学院大学
†Osaka University ††The Kyoto College of Graduate Studies for Informatics
Email: kawachi@lawschool.osaka-u.ac.jp

あらまし:大学などのプログラミング実習で作られたプログラムはセキュリティ対策が行われていないことが多い。最低限のセキュリティ対策について抑えるべきところをここで発表する.
キーワード:セキュリティ,問題解決型学習,マルチメディア利用

1. はじめに

筆者(河地)はかつて PHP の講義で他のサイトを用いてウェブサービスの構造を分析し,API として実装するプログラムを書いた.そのあと、ネットワーク技術者になりそのプログラムを公開しようと考えた.しかし PHP の講義知識だけでは公開に当たっては様々な問題があり、それを解決しなければならなかった.昨今では個人でもアプリケーションを無料で公開する人も出てきているがセキュリティに関しては意識していない人が多い.特に講義で作られたアプリは高確率でセキュリティ対策を行っていないことが多い。これはセキュリティが非機能要件であるので、教員自身が重要視していないことが原因である.この研究発表では PHP でプログラムを公開するにあたっての安全なプログラムの作り方について説明する.

今回のプログラムは以下図 1 の動作を行う.

図 1 想定する PHP プログラムの動作

今回使用するバージョンは Apache 2.6 系、PHP は 5.5 系,MySQL は 5.5 系である.

2. 確認すべき項目

PHP のプログラムで確認すべき点は最新版の関数やクラスを用いているか,入力に適切なエスケープ処理を施しているか,そして、データベースを用いるのであれば SQL インジェクション対策を施しているかどうかである.

2.1 関数が公式で非推奨

筆者はかつて PHP の古い参考書 具体的には PHP5.2 系で書かれたものや PHP 入門サイトを用いてプログラムを書いていた.しかし、それが数年後に PHP 公式から非推奨になる場合がある.例えば,MySQL を取り扱う関数群 mysql があるがこれは PHP5.5 以降非推奨となった.

```
//DB 接続 なければ DIE する
$link = mysql_connect(‘localhost’, ‘DB ユーザー名’, ‘パスワード')
        or die(mysql_error());
//データベースが違えばエラーを返す
if(!mysql_select_db(‘データベース名')){
    die('Could not select database: ' . mysql_error());
}
```

もし,古い教科書や更新されていない PHP 入門サイトを見て作られたプログラムがあれば一度確認することを推奨する.

2.2 入力文字列

次は入力文字列の確認である.特に HTML で POST された文字列は何も対策を施されていないと悪質な JavaScript 埋め込み攻撃(クロスサイトスクリプティング・通称 XSS 攻撃)や後述する SQL インジェクション攻撃の原因になる.例えば、簡単な文字列検索プログラムをセキュリティ対策もなしに作ったとしよう.その場合入力部にスクリプトタグを埋め込まれたら埋め込んだ攻撃者の思いのままになってしまう.

2.3 SQL インジェクション攻撃

SQL を用いて結果を出すアプリケーションにおいて一番気を付けなければならない.例えば以下の SQL 文について考える.

```
$sql=Select * from table_X where name =’$name’;
```

ここでの変数 name は 1 単語で検索することを想定している.だが,この変数 name が何も対策しないでおくとこのような値が代入される.

```
‘;Delete from table_x—
```

変数内で別の SQL 命令を書き、後で書かれたコードが実行されてしまうのである.PHP の場合変数に

型を指定しないため、仮に where 文を id=$id に変えたとしてもこれは

0;Delete from table_x

で攻撃できてしまう．これらを応用すると本来は触れることができないユーザー情報のテーブルを攻撃者が閲覧することができ，場合によってはデータベースサーバーの OS の攻撃の糸口まで発展できてしまう場合がある．

3. 対策

これら 3 つの確認すべき項目を挙げたがこれらによる対策は

関数を公式で推奨されているものに切り替える

入力部についてユーザーから入力されたものは基本的には信用せずエスケープ処理を施してからやるようにする

特に SQL は SQL 専用のエスケープ処理関数が用意されているので PHP の汎用エスケープ関数を利用せずこちらを使う

である．以下、順を追って説明しよう

3.1 関数が公式で推奨のものに切り替える

まず、PHP 公式サイトで今まで使っていた関数にこれがある場合は新しい関数に切り替える．入れ替えないと，次のバージョンで関数が削除され一切使えなくなる可能性がある．その場合 PHP マニュアルでこのような旨の文が出る．

警告

この拡張モジュールは PHP バージョン で非推奨になりました．将来のバージョンで削除される予定です．

そのあとに推奨される関数が出てくるのでそれを選択する．今回の場合は mysql を mysqli と PDO_MySQL のどちらかを使うことが推奨されている．mysql 関数の感覚，すなわち文字列連結でやる場合は mysqli を，オブジェクトのように取り扱いたい場合は PDO_MySQL を使うことをお勧めする．

3.2 入力文字列

先ほど、 でも述べたが基本的にユーザーが入力したものは信用しないことが大原則となる．そこでプログラムが安全な変数として使えるよう、エスケープ処理を行う必要がある．例えば，’ “ () <>等の文字は想定されるプログラムによっては不要になる場合がある．また、ブラックリストを用いて特定文字列の入力があった場合はエラーを返す方法もある．

3.2..1 SQL を使用しないエスケープ処理

’“ () <>等の文字は削除して文字列だけを表示する関数や数字であるかどうかを確認する関数が PHP で用意されている．is_string 関数や is_int 関数等を用いればよい．

3.2..2 SQL インジェクション対策

これについては 2 通りある．一つは PDO_MySQL を用いてプレースホルダで SQL を書く方法．IPA によればこの実装が一番安全であるそうだ．だが環境の都合で使えない場合は mysqli で実装する．2.1 の例ならばこのような書き方になる

```
    //DB 接続　なければ DIE する
  $link = mysqli_connect(‘localhost’, ‘DB ユーザー名’, ‘パスワード')
              or die(mysqli_error());
  //データベースが違えばエラーを返す
  if(!mysqli_select_db(‘データベース名')){
      die('Could not select database: ' . mysql_error());
}
```

そして入力部には必ず SQL 用のエスケープ処理を施す．そして一番重要なのは使う SQL 構文は必ず定数として定義しておくことである．SQL 構文の定数と SQL で使う変数を連結するときは quote メソッドでやることを IPA ではお勧めしている．注意点は PHP 標準のエスケープ関数は使用しないことである．必ずデータベース固有の関数を使用することを推奨する．

4. まとめ

ほかにも画像掲示板をやっていればディレクトリ・トラバーサルやログインシステムを実装しているのであればセッション対策をセットで行う必要がある．作ったプログラムに合わせてセキュリティ対策を行うことが重要である．PHP でプログラムを書いている人はやや古いがサイバーテロの技法は必ず読んでおくべきである．PHP に関するあらゆる攻撃が書かれている．WEB で正誤表が公開されているのでそれを確認してから読むことをお勧めする．また，IPA では，セキュリティ実装に必要なチェックリストを用意している．サーバー環境に合わせて確認することをお勧めする．こうすることで初めて公開に耐えうるプログラムが出来上がるのである．PHP で学習環境を整える教員はぜひ参考にしていただきたい．

参考文献

(1) IPA, "安全なウェブサイトの作り方第 6 版改訂", (2012) http://www.ipa.go.jp/files/000017316.pdf
(2) IPA, "安全な SQL の呼び出し方", (2010) http://www.ipa.go.jp/files/000017320.pdf
(3) PHP.net, "PHP マニュアル SQL インジェクション", http://php.net/manual/ja/security.database.sql-injection.php
(4) IPA, "セキュリティ実装チェックリスト", http://www.ipa.go.jp/files/000017317.xls
(5) GIJOE, "サイバーテロの技法　攻撃と防御の実際", , ソシム,(2005)