УДК 004.031.42

РЕАЛИЗАЦИЯ БЕЗОПАСНОСТИ WEB-ПРОЕКТА НА ЯЗЫКЕ PHP И СУБД MYSQL

Чубаха А.Е
Донецкий национальный технический университет
кафедра прикладной математики и информатики
E-mail: konstashka@yandex.ru

Аннотация

Чубаха А.Е. Реализация безопасности Web-проекта на языке PHP и СУБД MySQL. *Рассмотрена основная угроза безопасности web-ресурса. Определена основная задача при разработке сайта. Определены средства реализации данной задачи.*

Общая постановка проблемы

В последние годы во всем мире наблюдается постоянный рост активности в области онлайновой торговли, благодаря которой происходит ускорение бизнес-процессов за счет их проведения электронным образом. Создание статичных сайтов не вызывает проблем в обеспечении их безопасности. Но проблемы могут возникнуть с сайтами, которые принимают данные от пользователя.

Исследование проблемы и постановка задачи

В PHP хорошо реализована работа с различными файлами. Функции include(), require() и fopen() обрабатывают файлы, которые находятся на локальном диске, так и файлы, которые находятся на серверах в Интернете (для доступа к таким файлам используется их URL). Из-за неправильной обработки динамически подключаемых файлов или путей к файлам появляется большое количество уязвимых скриптов, с помощью которых можно узнать конфиденциальную информацию интернет-ресурса. Как известно, не существует ПО, которое не возможно было взломать. Поэтому главная задача - свести к минимуму вероятность взлома сайта.

Решение задачи

Прежде чем писать скрипт, нужно понимать, что данные, которые приходят из внешних источников, могут быть установлены во что угодно. Поэтому необходимо реализовать проверку пользовательского ввода. Например, когда необходимо передать id поля в базе данных в качестве параметра GET запроса, следует сделать такую проверку:

```
$id = (int)$HTTP_GET_VARS['id'];
```

Теперь можно быть уверенным, что $id содержит целое число, а не скрипт, который может похитить данные. Однако этот метод не подходит для проверки строковых выражений. Для этого можно использовать регулярное выражение.

```
<?php
        if (ereg("^[a-z]+\.html$", $id)) {
       echo "Все хорошо!";}
        else {
       die("Попытка взлома сайта!!!"); }
?>
```

В этом случае скрипт продолжит выполнение только в том случае, если переменная $id содержит имя файла, которое начинается с маленькой буквы(но не цифры) содержит только буквы и имеет html-расширение[1].

Наверное, наиболее спорным моментом в разработке PHP стала замена значения по умолчанию для опции register_globals с ON на OFF в версии 4.2.0. Если значение этого параметра ON и если учесть, что PHP не требует инициализации переменных, то написать опасный код очень легко, т.к. в случае неудачи при проверке авторизации, переменная,

которая отвечает за авторизацию, будет установлена автоматически благодаря register_globals. Таким образом, проверку можно пройти без авторизации. Поэтому, по моему мнению, целесообразно поменять register_globals с ON на OFF. Это приведет к тому, что все данные, которые приходят в скрипт через GET, POST-запросы, Cookie, переменные сервера, переменные окружения, а также переменные, которые хранятся в сессиях, не будут присутствовать больше в области видимости глобальных переменных.

Также существует возможность реализации реагирования на попытку подмены переменных. Так как во время разработки приложения мы знаем ожидаемое значение переменной, а также знаем ее достоверное значение, можно их сопоставить. Это не защитит код от подмены переменных, но усложнит перебор возможных вариантов[2].

```
<?php
if (isset($_COOKIE['MAGIC_COOKIE'])) {

   // MAGIC_COOKIE получена из достоверного источника.
   // Для полной уверенности необходимо проверить ее значение.

} elseif (isset($_GET['MAGIC_COOKIE']) || isset($_POST['MAGIC_COOKIE'])) {

  mail("admin@example.com", "Обнаружена попытка взлома",
$_SERVER['REMOTE_ADDR']);
  echo "Обнаружено нарушение безопасности.";
  exit;

} else {

  // MAGIC_COOKIE в данных запроса не присутствует
}
?>
```

Во время создания приложения очень полезно будет поставить уровень оповещения про ошибки приложения, так, чтобы выводились все ошибки, но в реально работающем приложении стоит указать уровень error_reporting равным 0. Желательно использовать функцию error_log() для ведения журнала приложения и даже для отправки оповещений на почтовый адрес. Даже можете сделать систему обнаружения попыток взлома и отправлять себе по почте оповещения, если кто-то пытается подменить данные, пересылаемые через параметры GET или POST запросов, либо через cookies[1].

Выводы

Программирование защищенных приложений требует намного больше времени, чем создание приложения, которое будет просто работоспособным. Также следует понимать, что один или несколько приемов, такие как установка register_globals в off, не сделает код безопасным. Необходимо каждую переменную, полученную от пользователя, проверять на соответствие ожидаемому значению и инициализировать все используемые переменные, многократно тестировать программный код по мере его написания.

Литература

1. Thomas Oertli. Основы безопасного программирования на PHP / Oertli Thomas,2006 г.

2.Руководство по PHP / Интернет-ресурс. –Режим доступа: www/URL: http://www.php.ru/manual/security.globals.html

3. Фленов, М.Е. PHP глазами хакера. –Спб.: БХВ-Петербург, 2005