Аутентификация в PHP

- Аутентификация
- Базовая аутентификация
- Аутентификация при помощи файлов .htaccess сервера Apache
- Использование аутентификации через модуль mod_auth_mysql

Аутентификация

- Просьба к пользователю доказать свою личность называется аутентификацией. Обычный метод аутентификации в Web — это требование к посетителям предоставить уникальное имя пользователя и пароль. Аутентификация обычно используется для разрешения или запрещения доступа к определенным страницам или ресурсам. Аутентификация может быть необязательной либо использоваться для других целей, например, для персонализации.

Аутентификация

- Простой контроль доступа реализовать несложно. Код, показанный в листинге, выводит одну из трех возможных страниц. Если этот файл загружен без параметров, будет отображаться HTML-форма с приглашением вести имя пользователя и пароль. Если при загрузке параметры присутствуют, но они неправильные, отображается сообщение об ошибке. Если при загрузке параметры присутствуют и они правильные, посетителю отображается секретное содержимое.

Аутентификация

- Этот код содержит несколько значительных проблем. Этот сценарий:
  - ✓поддерживает только одно жестко закодированное имя пользователя и пароль
  - ✓хранит пароль в виде простого текста
  - ✓защищает только одну страницу
  - ✓передает пароль в виде простого текста
- Упомянутые проблемы можно разрешить с различной степенью усилий и успеха.

Аутентификация

- Хранение паролей в отдельном файле на сервере позволит без труда написать программу для добавления и удаления пользователей, а также для изменения паролей.
- Если планируется сохранять большое количество элементов в файле или производить поиск в рамках большого числа элементов, то следует рассмотреть возможность использования базы данных вместо двумерного файла.
- Практический метод выбора гласит: если вы собираетесь хранить и производить поиск в более чем 100 элементах, следует отдать предпочтение базе данных.

Аутентификация

- Использование базы данных для хранения имен и паролей посетителей позволит быстро проводить аутентификацию множества пользователей. Это также упростит создание сценария для добавления и удаления пользователей, а также даст возможность пользователям изменять свои пароли.
- Сценарий для аутентификации посетителей страницы с использованием базы данных приведен в листинге.

Аутентификация

- Независимо от того, где хранятся пароли — в базе данных или в файле — хранение паролей в виде простого текста сопряжено с риском. Однонаправленный алгоритм хэширования обеспечит дополнительную защиту при незначительных дополнительных затратах.
- PHP-функция **`crypt()`** представляет собой однонаправленную криптографическую хэш-функцию. Прототип этой функции таков:

```
string crypt (string str[,string salt])
```

- Получив на входе строку **`str`**, эта функция возвращает псевдослучайную строку. Например, если передать в функцию строку "**`pass`**" и аргумент **`salt`** равный "**`xx`**", то **`crypt()`** вернет строку "**`xxkTlmYjlikoII`**".

Аутентификация

- Эта строка не может быть дешифрована и превращена обратно в "**pass**". Результат этой функции детерминирован. При каждом вызове с одними и теми же параметрами эта функция будет возвращать один и тот же результат.
- Вместо PHP-кода, такого как

```
if( $username  ==  "user" && password
  == "pass" ){ // Пароль совпадает
}
```

можно воспользоваться таким кодом

```
if($username=          'user'          &&
  crypt($password,'xx')                ==
  'xxkTlmYjlikoII'){// Пароль совпадает
}
```

Аутентификация

- Если для хранения данных аутентификации используется база данных MySQL, можно воспользоваться функцией **`PASSWORD()`**. Результат этих функций не совпадает, но они имеют одно предназначение. Обе функции — **`crypt()`** и **`PASSWORD()`** — получают строку как аргумент и применяют к поденной строке необращаемый алгоритм хэширования.
- Чтобы задействовать функцию **`PASSWORD()`** в листинге 12 запрос SQL следует переписать так:

```
select count (*) from auth where
name = '$name' and
pass = password('$password')
```

Аутентификация

- Этот запрос посчитает количество строк в таблице **`auth`**, в которых значение поля name совпадает с содержимым переменной **`$name`**, а поля pass - с результатом функции **`PASSWORD()`**, примененной к значению переменной **`$password`**. Если мы заставляем посетителей выбирать уникальные имена, результатом запроса может быть 0 или 1.

Аутентификация

- Защита более чем одной страницы с помощью подобных сценариев немного сложнее. Потребуется включить части кода в каждую страницу, которую необходимо защитить. При помощи директив **`auto_prepend_file`** и **`auto_append_file`** требуемый файл можно автоматически вставить в начало (**`prepend`**) или в конец (**`append`**) каждого файла.
- Однако недопустимо запрашивать пароль отдельно для каждой страницы, которую желает просмотреть пользователь.

Аутентификация

- Можно включить введенную пользователем информацию в каждую гиперссылку на странице. Так как пользователи могут применять пробелы или другие символы, запрещенные в URL, следует обратиться к функции `urlencode()`, чтобы безопасно упаковать подобные символы.
- Но защищенные страницы, которые посетил пользователь, могут быть просмотрены любым человеком, работающим за тем же компьютером. Для этого достаточно щелкнуть на кнопке "Назад" в окне браузера и просмотреть кэшированные копии страниц или заглянуть в историю посещения страниц.

Аутентификация

- Решить проблему можно с помощью двух механизмов — базовой HTTP-аутентификации и поддержки сеансов.
- Базовая аутентификация позволяет решить проблему кэширования, но сервер все равно отправляет пароль Web-браузеру в каждом запросе.
- Управление сеансами позволяет решить обе проблемы.

Базовая аутентификация

- Существуют возможности аутентификации, встроенные в HTTP-протокол. Web-серверы могут запрашивать аутентификацию у Web-браузера. После этого Web-браузер должен вывести на экран диалоговое окно и запросить у пользователя необходимую информацию.
- Браузер хранит детали аутентификации, пока открыто окно браузера, и автоматически отправляет их без вмешательства со стороны пользователя.
- Эта возможность HTTP-протокола называется базовой аутентификацией. Базовую аутентификацию можно включить средствами PHP или с помощью Web-сервера.

Базовая аутентификация

- Базовая аутентификация передает имя пользователя и пароль в виде простого текста. Протокол HTTP 1.1 обладает дайджест-аутентификацией. Этот метод использует алгоритм хэширования (как правило, MD5) для маскировки деталей транзакции. Дайджест-аутентификация поддерживается во многих Web-серверах, но лишь в браузере Microsoft Internet Explorer начиная с версии 5.0. Поддержка дайджест-аутентификации включена в Netscape Navigator 6.0.
- Использование протокола SSL и цифровых сертификатов позволяет более надежно защитить все части транзакций в Web.

Базовая аутентификация

- Базовая аутентификация позволяет защитить именованные области и требует от пользователей ввода правильного имени и пароля. Области именованные, поэтому на одном сервере может быть существовать множество областей. Различные файлы и каталоги на одном сервере могут принадлежать разным областям, каждая из которых защищена своими наборами имен пользователей и паролей. Именованные области позволяют также сгруппировать в одну область несколько каталогов на одном физическом или виртуальном узле и защитить всю область одним паролем.

Базовая аутентификация

- Использование базовой аутентификации базируется на переменных среды, устанавливаемых сервером. Сценарий HTTP-аутентификации должен определять тип сервера и вести себя соответствующим образом в зависимости от того, выполняется ли он как модуль Apache на сервере Apache или как ISAPI-модуль на сервере IIS.

Базовая аутентификация

- При обработке неудачных попыток аутентификации Internet Explorer дает пользователю три попытки, и если все они проходят неудачно, выводится сообщение об отказе в доступе. Netscape Navigator дает неограниченное число попыток, но между ними выводит диалоговое окно с запросом "Authentication Failed. Retry?”, и отображает сообщение об отказе в доступе, только когда пользователь щелкает на кнопке Cancel.
- Код данного примера можно вставить в начало каждого файла, который требуется защитить. Это можно сделать вручную или автоматически для каждого файла в каталоге.

Аутентификация при помощи файлов .htaccess сервера Apache

- Результат, подобный результату предыдущего сценария, может быть достигнут и без написания PHP-сценария. Сервер Apache обладает множеством различных методов аутентификации, которые можно использовать для определения правильности введенных пользователем данных. Наиболее простой из этих методов — это применение модуля **mod_auth**, которая сравнивает пары имя-пароль со строками текстового файла на сервере.
- Чтобы получить на экране пользователя результат предыдущего сценария, потребуется создать два HTML-файла — один для содержимого страницы и один для сообщения об отказе в доступе.

Аутентификация при помощи файлов .htaccess сервера Apache

- Ниже показан HTML-файл, который будут видеть пользователи, прошедшие аутентификацию и файл с сообщением об отказе в доступе. Этот файл называется reject.html. Создавать страницу, которая будет отображаться в случае ошибки, вовсе не обязательно. Можно использовать стандартный файл, выводимый при ошибке 401, либо ввести произвольный текст в файл конфигурации доступа к каталогу.

Аутентификация при помощи файлов .htaccess сервера Apache

- index.html
- <html><head><title>HTACCESS test page</title></head><body> <h1>
- Вы видите эту страницу, если верно ввели пароль из файла .htaccess
- </h1> </body></html>
- reject.html
- <html><head><title>HTACCESS test page</title></head><body> <h1>
- Вы не прошли авторизацию!!! Доступ запрещен!!!
- </h1> </body></html>

Аутентификация при помощи файлов .htaccess сервера Apache

- Самый главный файл в защищаемом каталоге - **.htaccess**. Он управляет доступом к файлам и подкаталогам каталога, в котором он расположен. Заметим, что в конфигурации Apache по умолчанию файлы, начинающиеся с .ht запрещены для просмотра через Web.
- С помощью этого файла можно установить различные параметры, однако в этом примере все шесть строк относятся к аутентификации.

Аутентификация при помощи файлов .htaccess сервера Apache

- Заметим, что допустимые параметры определяются при описании каталога в файле конфигурации **httdp.conf**, например, так:
- `<Directory "D:/Program Files/Apache Group/Apache/htdocs/access">`
- `Options Indexes`
- `AllowOverride All`
- `Order allow,deny`
- `Allow from all`
- `</Directory>`
- Здесь директива **AllowOverride** и определяет перечень директив, допустимых в файле **.htaccess**.

Аутентификация при помощи файлов .htaccess сервера Apache

- Первая строка
- **`ErrorDocument 401 /reject.html`**
- указывает серверу Apache, какой документ следует отобразить посетителям, не прошедшим аутентификацию. Директиву ErrorDocument можно использовать несколько раз в одном файле, чтобы указать собственные страницы для сообщений о других ошибках протокола HTTP, например, об ошибке 404. Синтаксис этой директивы таков:
- **`ErrorDocument error_number URL или`**
- **`ErrorDocument error_number “message`**

Аутентификация при помощи файлов .htaccess сервера Apache

- Важно, чтобы страница с сообщением об ошибке 401 была доступна для всех посетителей. Глупо располагать ее в каталоге, в который можно попасть только после успешного прохождения аутентификации.
- Строка
- **`AuthUserFile /home/book/.htpass`**
- указывает серверу, где искать файл, содержащий пользовательские пароли. Этот файл часто называют **.htpass**, но можно выбрать произвольное имя файла. Помните, однако, что файлы **`.ht*`** защищены от просмотра через Web.

Аутентификация при помощи файлов .htaccess сервера Apache

- Можно указать, что доступ к ресурсу разрешен санкционированным пользователям только определенной группы. В данном примере это не предпринимается и строка
- `AuthGroupFile /dev/null`
- устанавливает параметр **AuthGroupFile** указывающим на /dev/null.
- Как и в примере с PHP, для использования HTTP-аутентификации защищаемой области следует присвоить имя. Это выполняет следующая строка
- `AuthName "Realm-Name"`
- Имя области может быть произвольным, но следует помнить, что это имя будет видно посетителям.

Аутентификация при помощи файлов .htaccess сервера Apache

- Поскольку сервер поддерживает различные методы аутентификации, следует указать, какой именно метод следует использовать.
- **`AuthType Basic`**
- Также следует указать, кому разрешен доступ. Можно указать определенные группы, определенных пользователей, или, как сделано в примере, всех санкционированных пользователей. Строка
- **`require valid-user`**
- указывает, что доступ разрешен всем пользователям, успешно прошедшим аутентификацию.

Аутентификация при помощи файлов .htaccess сервера Apache

- Вернемся к файлу **.htpass**. Каждая строка в нем содержит имя пользователя и зашифрованный пароль, разделенные двоеточием. Чтобы создать такой файл, воспользуйтесь утилитой **htpasswd** из комплекта сервера Apache. Эту программу можно использовать одним из двух следующих способов.
- `htpasswd [-cmdps] passfile username`
- `htpasswd -b[cmdps] passfile username password`
- Главный аргумент — это `-c`. Он указывает, что требуется создать новый файл. Используйте этот аргумент только для создания первого пользователя. Если файл уже существует, то htpasswd удалит его и создаст новый файл с тем же именем.

Аутентификация при помощи файлов .htaccess сервера Apache

- Необязательные аргументы m, d, p, и s следует применять для выбора алгоритма шифрования (включая отказ от шифрования).
- **-m MD5 (по умолчанию).**
- **-d CRYPT.**
- **-p Не шифровать (plaintext).**
- **-s SHA.**
- Аргумент b заставляет утилиту ожидать пароль в качестве аргумента, а не запрашивать его отдельно. Этот аргумент полезен для запуска **htpasswd** в неинтерактивном режиме как часть командного файла, но этот аргумент не следует использовать при вызове **htpasswd** из командной строки.

Аутентификация при помощи файлов .htaccess сервера Apache

- Аутентификацию такого типа легко настроить, но с ней связано несколько проблем.
- Имена и пароли пользователей хранятся в текстовом файле. Каждый раз, когда браузер запрашивает файл, защищенный через **.htaccess**, сервер должен проанализировать этот файл и после этого проанализировать еще и файл **.htpass**, чтобы найти соответствующее имя пользователя и пароль.
- Независимо от места хранения директив сервера, файл паролей анализируется при каждом запросе. Это означает, что подобно другим рассмотренным методам, которые используют двумерный файл, данный метод не годится в случае обработки сотен тысяч пользователей.

Использование аутентификации через модуль mod_auth_mysql

- Модуль **mod_auth_mysql** использует базы данных и способен быстрее производить поиск в больших списках пользователей.
- Чтобы использовать этот модуль, его следует скомпилировать и установить, или обратиться к системному администратору. Заметим, что компиляция требует наличия исходных текстов сервера Apache, и компилятора, совместимого с **gcc**.
- Ниже показан пример файла **.htaccess**, который позволит аутентифицировать пользователей с шифрованием паролей, которые будут храниться в созданной ранее базе данных.

Использование аутентификации через модуль mod_auth_mysql

- `ErrorDocument 401 /reject.html`
- `AuthName "Realm Name"`
- `AuthType Basic`
- `Auth_MySQL_DB auth`
- `Auth_MySQL_Encryption_Types MySQL`
- `Auth_MySQL_Password_Table auth`
- `Auth_MySQL_Username_Field name`
- `Auth_MySQL_Password_Field pass`
- `require valid-user`

Использование аутентификации через модуль mod_auth_mysql

- Здесь указывается файл с сообщением об ошибке 401 (неудачная аутентификация). Снова указана базовая аутентификация и имя области аутентификации. И доступ разрешен любому пользователю, который успешно прошел аутентификацию. Однако, указываются и дополнительные директивы для настройки. Директивы

```
Auth_MySQL_DB,
Auth_MySQL_Password_Table,
Auth_MySQL_Username_Field,
Auth_MySQL_Password_Field
```

используются для указания, соответственно, имени базы данных, таблицы и полей имени и пароля пользователя.

Использование аутентификации через модуль mod_auth_mysql

- В листинге присутствует директива **Auth_MySQL_Encryption_Types**, которая обеспечивает выбор типа шифрования. В примере выбрано шифрование паролей MySQL. Для этой директивы приемлемыми являются значения Plaintext, Crypt_DES или MySQL. Значение Crypt_DES активизирует использование стандартного для UNIX алгоритма шифрования DES.

Использование аутентификации через модуль mod_auth_mysql

- С точки зрения пользователя, пример с модулем mod_auth_mysql работает в точности так, как пример с .htaccess. Пользователь видит диалоговое окно в своем браузере. Если пользователь проходит аутентификацию, он увидит защищенное содержимое страницы, а если нет — сообщение об ошибке.
- Для большинства Web-сайтов аутентификация с помощью модуля mod_auth_mysql подходит практически идеально. Этот метод позволяет использовать любой удобный механизм для создания новых учетных записей в базе данных.