**Лекция 15. Основные методы обеспечения безопасности** **Web-приложений**

Общие рекомендации по обеспечению безопасности веб-приложения. Безопасный доступ к базам данных. Безопасные сообщения об ошибках. Безопасное хранение секретной информации.Безопасное использование файлов Cookie. Защита от атак типа "отказ в обслуживании"

Проблема создания безопасного веб-приложения очень обширна. Для ее рассмотрения необходимо полное понимание слабых мест в системе безопасности. Также необходимо ознакомиться с функциями обеспечения безопасности Windows, .NET Framework и ASP.NET. И, наконец, очень важно понимать способы использования этих функций безопасности для борьбы с угрозами.

Даже при отсутствии опыта обеспечения безопасности существуют основные меры обеспечения безопасности веб-приложения, которые следует предпринять. В следующем списке приводятся рекомендации по обеспечению минимального уровня безопасности, применимых к любым веб-приложениям, которые следует соблюдать.

**Общие рекомендации по обеспечению безопасности веб-приложения**

Даже самые тщательно продуманные системы безопасности приложения могут быть взломаны, если злонамеренный пользователь может воспользоваться простыми способами доступа к компьютерам. Необходимо соблюдать следующие правила.

* Чаще создавайте резервную копию и храните ее в физически безопасном месте.
* Размещайте компьютер веб-сервера в физически безопасном месте, куда не могут попасть злоумышленники, выключайте его или забирайте с собой.
* Используйте файловую систему Windows NTFS, а не FAT32. NTFS обеспечивает значительно более высокий уровень безопасности, чем FAT32. Подробные сведения см. в документации Microsoft Windows.
* Компьютер веб-сервера и все компьютеры одной сети необходимо защитить с помощью надежных паролей.
* Обеспечьте безопасность служб IIS.
* Закройте неиспользуемые порты и отключите неиспользуемые службы.
* Установите антивирусное программное обеспечение, отслеживающее входящий и исходящий трафик.
* Создайте и внедрите политику, запрещающую пользователям записывать свои пароли в местах, где их легко найти.
* Используйте брандмауэр. См. рекомендации на веб-узле обеспечения безопасности Майкрософт [Рекомендации Майкрософт по использованию брандмауэра](http://go.microsoft.com/fwlink/?LinkId=50217).
* Установите последние пакеты исправлений от корпорации Майкрософт и других разработчиков. Например, на веб-узле [Центр безопасности TechNet](http://go.microsoft.com/fwlink/?LinkId=50530) приводится список последних бюллетеней безопасности для всех продуктов Майкрософт. Другие разработчики имеют такие же веб-узлы.
* Используйте журналы событий Microsoft Windows и регулярно проверяйте их на предмет подозрительных операций. Следует обращать внимание на частые попытки входа в систему или на слишком большое число запросов к веб-серверу.

**Запускайте приложения с наименьшими привилегиями**

Каждое приложение запускается в некотором контексте, обладающем определенными правами доступа на локальном компьютере, а также, возможно, на удаленных компьютерах. Чтобы запустить приложение с минимальными необходимыми привилегиями, выполняйте следующие рекомендации.

* Не запускайте приложение с правами системного пользователя (администратора).
* Запускайте приложение в контексте пользователя с минимальными необходимыми привилегиями.
* Задайте разрешения доступа (списки управления доступом) ко всем ресурсам, необходимым приложению. Используйте настройки с минимальным уровнем разрешений. Например, сделайте файлы доступными только для чтения, если это приемлемо для приложения.
* Храните файлы веб-приложения в папке под корнем приложения. Не разрешайте пользователям задавать пути доступа к файлам приложения. Это позволит избежать получения пользователями доступа к корню сервера.

**Выполняйте идентификацию и проверку подлинности пользователей**

Во многих приложениях пользователи могут получить анонимный доступ к узлу (не вводя учетные данные). При этом приложение получает доступ к ресурсам, запускаясь в контексте стандартного пользователя. По умолчанию используется контекст локального пользователя ASPNET (для Windows 2000 или Windows XP) или пользователя NETWORK SERVICE (для Windows Server 2003) на компьютере веб-сервера. Чтобы разрешить доступ к ресурсам только тем пользователям, которые прошли проверку подлинности, необходимо соблюдать следующие правила.

* Если приложение — это внутреннее сетевое приложение, настройте его так, чтобы оно использовало встроенные средства обеспечения безопасности ОП. Таким образом, учетные данные пользователя для входа в систему можно использовать для доступа к ресурсам. Дополнительные сведения см. в разделе [Олицетворение ASP.NET](https://msdn.microsoft.com/ru-ru/library/xh507fc5%28v%3Dvs.100%29.aspx).
* Чтобы потребовать от пользователя ввода своих учетных данных, используйте одну из стратегий проверки подлинности ASP.NET.

**Защита от ввода вредоносных данных**

Как правило, вводимые пользователем данные не следует считать безопасными. Злоумышленники могут легко отправить потенциально опасную информацию из клиента в приложение. Чтобы защититься от данных, вводимых злоумышленником, соблюдайте следующие правила.

* На страницах ASP.NET фильтруйте вводимые пользователем данные и проверяете их на наличие тегов HTML, которые могут содержать сценарий. [Защита от использования сценариев в веб-приложениях с помощью применения кодирования HTML к строкам](https://msdn.microsoft.com/ru-ru/library/a2a4yykt%28v%3Dvs.100%29.aspx).
* Не отображайте введенные пользователем данные без их предварительной фильтрации. До отображения непроверенных данных выполните HTML-кодирование, для преобразования потенциально вредоносного сценария в отображаемые строки.
* Никогда не храните нефильтрованные введенные пользователем данные в базе данных.
* Чтобы принять от пользователя HTML-данные, фильтруйте их вручную. В фильтре следует явным образом определить принимаемые данные. Не пытайтесь создать фильтр, пытающийся отфильтровать вредоносные данные, поскольку трудно предусмотреть все возможные типы таких данных.
* Не предполагайте, что информация, получаемая из заголовка HTTP-запроса (в объекте [HttpRequest](https://msdn.microsoft.com/ru-ru/library/system.web.httprequest%28v%3Dvs.100%29.aspx)), безопасна. Принимайте меры безопасности в отношении строк запросов, файлов Сookie и других элементов. Если информация, передаваемая обозревателем серверу (информация агента пользователя), важна для приложения, следует помнить, что она может быть подделана.
* По возможности не храните конфиденциальные сведения в местах, доступных из обозревателя (например в скрытых полях или файлах Cookie). К примеру, не следует сохранять имя пользователя или пароль в файле Cookie.

Состояние просмотра сохраняется в скрытом поле в закодированном формате. По умолчанию оно включает код проверки подлинности сообщения (MAC), позволяющий определить, не были ли подделаны данные состояния просмотра. Если конфиденциальные данные хранятся в состоянии просмотра, шифруйте их, присвоив свойству [ViewStateEncryptionMode](https://msdn.microsoft.com/ru-ru/library/system.web.ui.page.viewstateencryptionmode%28v%3Dvs.100%29.aspx) страницы значение **true**.

**Безопасный доступ к базам данных**

Как правило, базы данных имеют собственные средства обеспечения безопасности. Важным аспектом при создании безопасного веб-приложения является разработка способа доступа к базе данных из приложения. Необходимо соблюдать следующие правила.

* Для ограничения доступа к ресурсам базы данных используйте ее встроенные средства безопасности. Конкретная стратегия зависит от базы данных и приложения:
* По возможности используйте встроенные средства безопасности, чтобы доступ к базе данных был возможен только для пользователей, прошедших проверку подлинности Windows. Встроенные функции обеспечения безопасности более надежны, чем передача явно указанных учетных данных в базу данных.
* Если приложение позволяет анонимный вход в систему, создайте единого пользователя с очень ограниченным числом разрешений и выполняйте запросы, подключаясь как этот пользователь.
* Не создавайте инструкции SQL путем объединения строк, содержащих введенные пользователем данные. Вместо этого лучше создать параметризованный запрос и использовать введенные пользователем данные в качестве значений этих параметров.
* Если необходимо сохранять имя пользователя и пароля, чтобы использовать их в качестве учетных данных для входа в базу данных, храните из в файле Web.config и обеспечьте безопасность этого файле с использованием защищенной конфигурации.

**Создание безопасных сообщений об ошибках**

Если не предпринять необходимые меры предосторожности, злонамеренный пользователь может почерпнуть важные сведения о приложении из отображаемых им сообщений об ошибках. Необходимо соблюдать следующие правила.

* Не включайте в сообщения об ошибках информацию, которая может быть использована злонамеренными пользователями (например, имя пользователя).
* Настройте приложение таким образом, чтобы оно не выводило подробных сведений об ошибках для пользователей. Для отображения подробных сообщений об ошибках в целях отладки сначала определите, является ли пользователь локальным пользователем веб-сервера.
* С помощью элемента конфигурации [customErrors](https://msdn.microsoft.com/ru-ru/library/h0hfz6fc%28v%3Dvs.100%29.aspx) определите, кто может просматривать сведения об исключениях с сервера.
* Для операций, при выполнении которых могут происходить ошибки, таких как доступ к базе данных, напишите код обработки ошибок.

**Защита конфиденциальной информации**

Конфиденциальная информация — это любые сведения, разглашение которых нежелательно. Типичными примерами важной информации являются пароль и ключ шифрования. Если злонамеренный пользователь получает доступ к важной информации, то эти данные подвергаются опасности. Необходимо соблюдать следующие правила.

* Если приложение передает конфиденциальную информацию между обозревателем и сервером, используйте протокол SSL.
* Для защиты важной информации в файлах конфигурации, таких как Web.config или Machine.config, используйте защищенную конфигурацию.
* Чтобы хранить конфиденциальную информацию, не сохраняйте ее на веб-странице, даже в такой форме, в которой, по вашему мнению, пользователи не смогут ее увидеть (например, в серверном коде).
* Используйте алгоритмы строгого шифрования, реализованные в пространстве имен [System.Security.Cryptography](https://msdn.microsoft.com/ru-ru/library/system.security.cryptography%28v%3Dvs.100%29.aspx).

**Безопасное использование файлов Cookie**

Файлы Cookie — это простой и удобный способ предоставления доступа к специфичной для пользователя информации. Однако, поскольку файлы Сookie отправляются на компьютер обозревателя, они уязвимы для подмены и других злонамеренных действий. Необходимо соблюдать следующие правила.

* Не храните в файлах Сookie важную информацию. Например, не следует, даже временно, сохранять в файле Cookie пароль пользователя. Как правило, не следует сохранять в файлах Cookie такие сведения, которые при их краже могут нарушить безопасность приложения. Вместо этого храните в файлах Cookie ссылку на расположение информации на сервере.
* Устанавливайте для файлов Cookie минимальный срок действия. По возможности избегайте использования постоянных файлов Cookie.
* Шифруйте информацию в файлах Cookie.
* Свойствам [Secure](https://msdn.microsoft.com/ru-ru/library/system.web.httpcookie.secure%28v%3Dvs.100%29.aspx) и [HttpOnly](https://msdn.microsoft.com/ru-ru/library/system.web.httpcookie.httponly%28v%3Dvs.100%29.aspx) файла Cookie присвойте значение **true**.

**Защита от атак типа "отказ в обслуживании"**

Злонамеренный пользователь может причинить косвенный вред приложению, сделав его недоступным. Злоумышленник может настолько нагрузить приложение, что оно не может обслуживать других пользователей или происходит сбой приложения Соблюдайте следующие правила.

* Используйте обработку ошибок (например, try-catch). Включите блок Finally, в который освобождаются ресурсы в случае сбоя.
* Настройте службы IIS для регулирования процесса, что позволяет избежать использования неограниченного объема времени ЦПУ приложением.
* Проверяйте вводимые пользователем данные на соответствие ограничению размера до их использования или сохранения.
* Установите ограничения размера для запросов к базе данных. Например, до отображения результатов запроса на веб-странице ASP.NET убедитесь, что допустимое число записей не превышено.
* Установите ограничения на размер для загрузки файлов, входящих в состав приложения. Можно задать ограничение в файле Web.config с помощью синтаксиса, аналогичного приведенному ниже, где значение maxRequestLength дано в килобайтах.

<configuration>

 <system.web>

 <httpRuntime maxRequestLength="4096" />

 </system.web>

</configuration>

Можно также использовать свойство [RequestLengthDiskThreshold](https://msdn.microsoft.com/ru-ru/library/system.web.configuration.httpruntimesection.requestlengthdiskthreshold%28v%3Dvs.100%29.aspx) для снижения объема используемой памяти при загрузке больших объемов и передаче форм.

**Комплекс по защите web-приложений**

Комплекс по защите web-приложений включает в себя модуль:

- проактивныйфильтр (WebApplicationFireWall);

- панель безопасности с уровнями защищенности;

- технологию одноразовых паролей (OTP);

- защиту авторизованных сессий;

- контроль активности;

- защиту редиректов от фишинга;

- внешний контроль инфосреды;

- шифрование канала передачи через SSL;

- журнал вторжений;

- защиту административных разделов по IP;

- стоп-листы;

- контроль целостности скрипта;

- рекомендации по настройке;

- монитор обновлений.

**Проактивный фильтр**

Проактивный фильтр – обеспечивает защиту от большинства известных атак на web-приложения. В потоке внешних запросов пользователей проактивный фильтр распознает большинство опасных угроз и блокирует вторжения на сайт. Проактивный фильтр – наиболее эффективный способ защиты от возможных ошибок безопасности, допущенных при реализации интернет-проекта (XSS, SQL инъекций, PHP инъекций и ряда других). Действие фильтра основано на анализе и фильтрации всех данных, поступающих от пользователей через переменные и куки.Функциональные возможности фильтра:

- экранирование приложения от наиболее активно используемых атак;

- создание списка страниц-исключений из фильтрации (по маске);

- распознавание большинства опасных угроз;

- блокировка вторжений на сайт;

- защиты от возможных ошибок безопасности;

- фиксирование попыток атак в журнале;

- информирование администратора о случаях вторжения.

Настройка активной реакции - действий системы при попытке вторжения на сайт:

- сделать данные безопасными;

- очистить опасные данные;

- добавить IP адрес атакующего в стоп-лист на “ХХ” минут;

- занести попытку вторжения в журнал.

**Панель безопасности с уровнями защищенности**

Любой web-проект, работающий под управлением «1С-Битрикс: Управление сайтом», обязательно имеет начальный уровень защиты. Однако, с помощью модуля «Проактивная защита» можно значительно повысить защищенность собственного сайта. Нужно выбрать и настроить один из уровней безопасности модуля: стандартный; высокий; повышенный. При этом, система подскажет - какое действие необходимо установить для каждого параметра на выбранном текущем уровне.

Стандартный уровень – в проекте задействованы стандартные инструменты проактивной защиты продукта:

- проактивный фильтр (на весь сайт без исключений);

- ведется журнал вторжений за последние 7 дней;

- включен контроль активности;

- повышенный уровень безопасности для группы администраторов;

- использование CAPTCHA при регистрации (Completely Automated Public Turing test to tell Computers and Humans Apart- небольшой тест для ввода, код капчи позволяет отличить реальных людей от компьютеров);

- режим вывода ошибок.

Высокий уровень – рекомендованный уровень защиты, получают проекты, выполнившие требования стандартного уровня, и дополнительно включившие:

- журналирование событий главного модуля;

- защита административной части;

- хранение сессий в базе данных;

- смена идентификатора сессий.

Повышенный уровень – специальные средства защиты, обязательные для сайтов, содержащих конфиденциальную информацию пользователей, для интернет-магазинов, для тех, кто работает с критичной информацией.

Дополнительно к высокому уровню:

- включение одноразовых паролей;

- контроль целостности скрипта контроля.

**Журнал вторжений**

В Журнале регистрируются все события, происходящие в системе, в том числе необычные или злонамеренные. Оперативный режим регистрации этих событий позволяет просматривать соответствующие записи в Журнале сразу же после их генерации. В свою очередь, это позволяет обнаруживать атаки и попытки атак в момент их проведения. Это значит, у вас есть возможность немедленно принимать ответные меры, и, в некоторых случаях, даже предупреждать атаки.

Оперативная регистрация всех событий в системе в случае срабатывания проактивного фильтра запись в Журнале в одной из категорий атак:

- попытка внедрения SQL;

- попытка атаки через XSS;

- попытка внедрения PHP;

- отбор злонамеренных событий по фильтру;

- просмотр и анализ событий в реальном времени;

- немедленная реакция - ответная мера на событие;

- профилактика и предупреждение событий на основе их анализа.

**Одноразовые пароли**

Модуль «Проактивная защита» позволяет включить поддержку одноразовых паролей и использовать их выборочно для любых пользователей на сайте. Однако особо рекомендуется задействовать систему одноразовых паролей администраторам сайтов, поскольку это сильно повышает все уровни безопасности пользовательской группы «Администраторы».

Система одноразовых паролей дополняет стандартную систему авторизации и позволяет значительно усилить систему безопасности интернет-проекта. Для включения системы необходимо использовать аппаратное устройство (например, Aladdin eToken PASS) или соответствующее программное обеспечение, реализующее OTP.

Эта технология даетоднозначную уверенность, что на сайте авторизуется именно тот пользователь, которому выдан брелок. При этом какое-то похищение и перехват паролей теряет всякий смысл, так как пароль одноразовый. Брелок же физический, дает уникальные одноразовые пароли и только при нажатии. А это значит, что владелец брелка не сможет передать пароль другому человеку, продолжая пользоваться входом на сайт.

Функциональные возможности модуля:

- усиление системы безопасности интернет-проекта;

- использование аппаратных устройств;

- использование ПО, реализующего OTP;

- расширенная аутентификация с одноразовым паролем - при авторизации на сайте пользователь в дополнение к паролю дописывает одноразовый пароль;

- авторизация только с использованием имени (login) и составного пароля;

- заполнение при инициализации двух последовательно сгенерированных одноразовых паролей, полученных с устройства;

- восстановление синхронизации в случае нарушения синхронизации счетчика генерации в устройстве и на сервере.

**Контроль целостности файлов**

Контроль целостности файлов необходим для быстрого выяснения - вносились ли изменения в файлы системы.

В любой момент вы можете проверить целостность ядра, системных областей, публичной части продукта. Функциональные возможности модуля:

- отслеживание изменений в файловой системе;

- проверка целостности ядра;

- проверка системных областей;

- проверка публичной части продукта.

**Проверка целостности скрипта контроля**

Перед проверкой целостности системы необходимо проверить скрипт контроля на наличие изменений. При первом запуске скрипта введите в форму произвольный пароль (состоящий из латинских букв и цифр, длиной не менее 10 символов), а также произвольное кодовое (ключевое) слово (отличное от пароля), и нажмите на кнопку «Установить новый ключ».

**Защита административного раздела**

Эта защита позволяет компаниям строго регламентировать сети, которые считаются безопасными и из которых разрешается сотрудникам администрировать сайт. Специальный интерфейс, в котором все это и делается - задается список или диапазоны IP адресов, из которых как раз и позволяется управление сайтом.

Функциональные возможности модуля:

- любые XSS/CSS атаки на компьютер пользователя становятся неэффективными, а похищение перехваченных данных для авторизации с чужого компьютера - абсолютно бесполезным;

- ограничение доступа к административной части всех IP адресов, кроме указанных;

- автоматическое определение системой IP адреса пользователя;

- ручной ввод разрешенного IP адреса;

- установка диапазона IP адресов, с которых разрешен доступ к административной части.

**Защита сессий**

Большинство атак на web-приложения ставят целью получить данные об авторизованной сессии пользователя. Включение защиты сессий делает похищение авторизованной сессии неэффективным. И, если речь идет об авторизованной сессии администратора, то ее надежная защита с помощью данного механизма является особо важной задачей. В дополнение к стандартным инструментам защиты сессий, которые устанавливаются в настройках группы, механизм защиты сессий включает специальные - и в некотором роде уникальные.

Хранение данных сессий в таблице модуля позволяет избежать чтения этих данных через скрипты других сайтов на том же сервере, исключив ошибки конфигурирования виртуального хостинга, ошибки настройки прав доступа во временных каталогах и ряд других проблем настройки операционной среды. Кроме того, это разгружает файловую систему, перенося нагрузку на сервер базы данных. Полный список возможностей:

- защита сессий несколькими способами:

- время жизни сессии (минуты);

- смена идентификатора сессии раз в несколько минут;

- маска сети для привязки сессии к IP;

- хранение данных сессий в таблице модуля;

- исключение ошибок конфигурирования виртуального хостинга;

- исключение ошибок настройки прав доступа;

- исключение проблем настройки операционной среды;

- разгрузка файловой системы;

- бесполезность похищения сессий злоумышленниками.

**Контроль активности**

Контроль активности позволяет установить защиту от чрезмерно активных пользователей, программных роботов, некоторых категорий DDoS-атак, а также отсекать попытки подбора паролей перебором. В настройках можно установить максимальную активность пользователей для вашего сайта (например, число запросов в секунду, которые может выполнить пользователь). Функциональные возможности модуля:

- защита от чрезмерно активных пользователей;

- защита от программных роботов;

- защита от некоторых категорий DDoS-атак;

- отсекание попыток подбора паролей перебором;

- установка максимальной активности пользователей для сайта;

- фиксирование превышения лимита активности пользователя в

журнале вторжений;

- блокирование пользователя, превысившего количество запросов в заданный временной интервал;

- вывод для заблокированного пользователя специальной информационной страницы.

**Стоплист**

Стоплист – таблица, содержащая параметры, используемые для ограничения доступа посетителей к содержимому сайта и перенаправлению на другие страницы. Все пользователи, которые попытаются зайти на сайт с IP адресами, включенными в стоп-лист, будут блокированы (рис. 17).Функциональные возможности модуля:

- перенаправление посетителей, которые найденв стоп-листе;

- блокировка пользователей по IP адресам из стоп-листа;

- ручное пополнение стоп-листа новыми записями;

- учет статистики пользователей, которым запрещен доступ к сайту;

- установка периода действия запрета на доступ к сайту для пользователя, IP-сети, маску сети, UserAgent и ссылку, по которой пришел пользователь;

- изменяемое сообщение, которое будет показано пользователю при попытке доступа к сайту.

**Шифрование хранилища базы данных РНР**¶

Вы можете установить соединения через SSL для шифрования связи клиент/сервер для повышения безопасности, или вы можете использовать ssh для шифрования сетевого соединения между клиентами и сервером базы данных. Если любой из них используется, то мониторинг вашего трафика и получение информации о вашей базы данных будет трудным для потенциального злоумышленника.

В случае, если взломщик получил непосредственный доступ к БД (в обход веб-сервера), он может извлечь интересующие данные или нарушить их целостность, если информация не защищена на уровне самой БД. Шифрование данных - хороший способ предотвратить такую ситуацию, но лишь незначительное количество БД предоставляют такую возможность.

Наиболее простое решение этой проблемы - установить вначале обыкновенный программный пакет для шифрования данных, а затем использовать его в ваших PHP-скриптах. PHP может вам помочь с этой задачей с помощью таких расширений как [Mcrypt](https://www.php.net/manual/ru/ref.mcrypt.php) и [Mhash](https://www.php.net/manual/ru/ref.mhash.php), реализующих довольно большое число алгоритмов шифрования. При таком подходе скрипт вначале шифрует сохраняемые данные, а затем дешифрует их при запросе. Ниже приведены примеры того, как работает шифрование данных в PHP-скриптах.

**Хеширование** ¶

В случае работы со скрытыми служебными данными, если не требуется их нешифрованное представление (т.е. его не нужно показывать), то, как следствие, можно использовать хеширование. Хорошо известный пример хеширования - хранение криптографического хеша от пароля в БД, вместо хранения оригинального значения.

В PHP 5.5 и выше, функция [password](https://www.php.net/manual/ru/ref.password.php) позволяет удобно хешировать конфиденциальные данные и работать с этими хешами. В 5.3.7+ можно использовать библиотеку  [password\_compat](https://github.com/ircmaxell/password_compat).

[password\_hash()](https://www.php.net/manual/ru/function.password-hash.php) позволяет хешировать заданную строку самым сильным из доступных алгоритмом и [password\_verify()](https://www.php.net/manual/ru/function.password-verify.php) проверяет, что заданный пароль совпадает с сохраненным в БД хешем.

Пример #1 Хеширование полей с паролями

<?php

// Сохраняем хеш пароля

$query  = sprintf("INSERT INTO users(name,pwd) VALUES('%s','%s');",

            pg\_escape\_string($username),

            password\_hash($password, PASSWORD\_DEFAULT));

$result = pg\_query($connection, $query);

// проверка введенного пользователем пароля на корректность

$query = sprintf("SELECT pwd FROM users WHERE name='%s';",

            pg\_escape\_string($username));

$row = pg\_fetch\_assoc(pg\_query($connection, $query));

if ($row && password\_verify($password, $row['pwd'])) {

    echo 'Добро пожаловать, ' . htmlspecialchars($username) . '!';

} else {

    echo 'Ошибка авторизации, ' . htmlspecialchars($username) . '.';

}

?>

В более старых версиях PHP можно использовать функцию [crypt()](https://www.php.net/manual/ru/function.crypt.php).

Пример #2 Хеширование паролей используя [crypt()](https://www.php.net/manual/ru/function.crypt.php)

<?php

// сохранение хешированного пароля

// $random\_chars получены например использованием /dev/random

$query  = sprintf("INSERT INTO users(name,pwd) VALUES('%s','%s');",

            pg\_escape\_string($username),

            pg\_escape\_string(crypt($password, '$2a$07$' . $random\_chars . '$')));

$result = pg\_query($connection, $query);

// проверка введенного пользователем пароля на корректность

$query = sprintf("SELECT pwd FROM users WHERE name='%s';",

            pg\_escape\_string($username));

$row = pg\_fetch\_assoc(pg\_query($connection, $query));

if ($row && crypt($password, $row['pwd']) == $row['pwd']) {

    echo 'Добро пожаловать, ' . htmlspecialchars($username) . '!';

} else {

    echo 'Авторизация не удалась, ' . htmlspecialchars($username) . '.';

}

?>

**SQL-инъекции ¶**

Многие веб-разработчики даже не догадываются, что SQL-запросы могут быть подделаны, и считают, что SQL-запросы всегда достоверны. На самом деле поддельные запросы могут обойти ограничения доступа, стандартную проверку авторизации, а некоторые виды запросов могут дать возможность выполнять команды операционной системы.

Прямое внедрение вредоносных инструкций в SQL-запросы - это методика, в которой взломщик создает или изменяет текущие SQL-запросы для отображения скрытых данных, их изменения или даже выполнения опасных команд операционной системы на сервере базы данных. Атака выполняется на базе приложения, строящего SQL-запросы из пользовательского ввода и статических параметров. Следующие примеры, к сожалению, построены на реальных фактах.

Благодаря отсутствию проверки пользовательского ввода и соединению с базой данных под учетной записью суперпользователя (или любого другого пользователя, наделенного соответствующими привилегиями), взломщик может создать еще одного пользователя БД с правами суперпользователя.

Пример #1 Постраничный вывод результата... и создание суперпользователя в PostgreSQL

<?php

$offset = $argv[0]; // внимание, нет проверки вводимых данных!

$query  = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";

$result = pg\_query($conn, $query);

?>

Обычно пользователи кликают по ссылкам 'вперед' и 'назад', вследствие чего значение переменной *$offset* заносится в URL. Скрипт ожидает, что *$offset* - десятичное число. Однако, взломщик может попытаться взломать систему, присоединив к URL дополнительную строку, обработанную функцией [urlencode()](https://www.php.net/manual/ru/function.urlencode.php):

0;

insert into pg\_shadow(usename,usesysid,usesuper,usecatupd,passwd)

 select 'crack', usesysid, 't','t','crack'

 from pg\_shadow where usename='postgres';

--

Если это произойдет, скрипт предоставит взломщику доступ к базе с правами суперпользователя. Заметим, что значение 0; использовано для того, чтобы задать правильное смещение для первого запроса и корректно его завершить.

Замечание:

Часто используемой техникой для игнорирования SQL-парсером оставшейся части запроса является использование --, означающей комментарий.

Еще один вероятный способ получить пароли учетных записей в БД - атака страниц, предоставляющих поиск по базе. Взломщику нужно лишь проверить, используется ли в запросе передаваемая на сервер и необрабатываемая надлежащим образом переменная. Это может быть один из устанавливаемых на предыдущей странице фильтров, таких как WHERE, ORDER BY, LIMIT и OFFSET, используемых при построении запросов SELECT. В случае, если используемая вами база данных поддерживает конструкцию UNION, взломщик может присоединить к оригинальному запросу еще один дополнительный, для извлечения пользовательских паролей. Настоятельно рекомендуем использовать только зашифрованные пароли.

Пример #2 Листинг статей... и некоторых паролей (для любой базы данных)

<?php

$query  = "SELECT id, name, inserted, size FROM products

           WHERE size = '$size'";

$result = odbc\_exec($conn, $query);

?>

Статическая часть запроса может комбинироваться с другим SELECT-запросом, который выведет все пароли:

union select '1', concat(uname||'-'||passwd) as name, '1971-01-01', '0' from usertable;

--

Если этот запрос (использующий ' и --) присоединить к значению одной из переменных, используемых для формирования *$query*, то запрос заметно преобразится.

Команды UPDATE также могут использоваться для атаки. Опять же, есть угроза разделения инструкции на несколько частей и присоединения дополнительного запроса. Также взломщик может видоизменить выражение SET. В этом случае потенциальному взломщику необходимо обладать некоторой дополнительной информацией о структуре базы данных для успешного манипулирования запросами. Эту информацию можно получить, проанализировав используемые в форме имена переменных, либо просто перебирая все наиболее распространенные варианты названия соответствующих полей (а их не так уж и много).

Пример #3 От восстановления пароля... до получения дополнительных привилегий (для любой базы данных)

<?php

$query = "UPDATE usertable SET pwd='$pwd' WHERE uid='$uid';";

?>

Но злоумышленник может ввести значение ' or uid like'%admin%' для переменной *$uid* для изменения пароля администратора или просто присвоить переменной *$pwd* значение hehehe', trusted=100, admin='yes для получения дополнительных привилегий. При выполнении запросы переплетаются:

<?php

// $uid: ' or uid like '%admin%

$query = "UPDATE usertable SET pwd='...' WHERE uid='' or uid like '%admin%';";

// $pwd: hehehe', trusted=100, admin='yes

$query = "UPDATE usertable SET pwd='hehehe', trusted=100, admin='yes' WHERE

...;";

?>

Пугающий пример того, как на сервере баз данных могут выполняться команды операционной системы.

Пример #4 Выполнение команд операционной системы на сервере (для базы MSSQL)

<?php

$query  = "SELECT \* FROM products WHERE id LIKE '%$prod%'";

$result = mssql\_query($query);

?>

Если взломщик введет значение a%' exec master..xp\_cmdshell 'net user test testpass /ADD' -- для переменной *$prod*, тогда запрос *$query* будет выглядеть так:

<?php

$query  = "SELECT \* FROM products

           WHERE id LIKE '%a%'

           exec master..xp\_cmdshell 'net user test testpass /ADD' --%'";

$result = mssql\_query($query);

?>

MSSQL сервер выполняет SQL-команды в пакетном режиме, в том числе и операции по заведению локальных учетных записей базы данных. В случае, если приложение работает с привилегиями администратора sa и сервис MSSQL запущен с необходимыми привилегиями, то выполнив приведенные выше действия, взломщик получит аккаунт для доступа к серверу.

Замечание:

Некоторые приведенные в этой главе примеры касаются конкретной базы данных. Это не означает, что аналогичные атаки на другие программные продукты невозможны. Работоспособность вашей базы данных может быть нарушена каким-либо другим способом.

Способы защиты ¶

Хотя по-прежнему очевидно, что взломщик должен обладать по крайней мере некоторыми знаниями о структуре базы данных чтобы провести успешную атаку, получить эту информацию зачастую очень просто. Например, если база данных является частью open-source или другого публично доступного программного пакета с инсталляцией по умолчанию, эта информация является полностью открытой и доступной. Эти данные также могут быть получены из закрытого проекта, даже если он закодирован, усложнен, или скомпилирован, и даже из вашего личного кода через отображение сообщений об ошибках. К другим методам относится использование распространенных (легко угадываемых) названий таблиц и столбцов. Например, форма логина, которая использует таблицу 'users' c названиями столбцов 'id', 'username' и 'password'.

Большинство успешных атак основывается на коде, написанном без учета соответствующих требований безопасности. Не доверяйте никаким вводимым данным, особенно если они поступают со стороны клиента, даже если это списки в форме, скрытые поля или куки. Первый приведенный пример показывают, как подобные запросы могут привести к катастрофе.

* Никогда не соединяйтесь с базой данных, используя учетную запись владельца базы данных или суперпользователя. Всегда старайтесь использовать специально созданных пользователей с максимально ограниченными правами.
* используйте подготовленные выражения с привязанными переменными. Эта возможность предоставляется расширениями [PDO](https://www.php.net/manual/ru/pdo.prepared-statements.php), [MySQLi](https://www.php.net/manual/ru/mysqli.quickstart.prepared-statements.php) и другими библиотеками.
* Всегда проверяйте введенные данные на соответствие ожидаемому типу. В PHP есть множество функций для проверки данных: начиная от простейших [функций для работы с переменными](https://www.php.net/manual/ru/ref.var.php) и [функций определения типа символов](https://www.php.net/manual/ru/ref.ctype.php) (таких как [is\_numeric()](https://www.php.net/manual/ru/function.is-numeric.php) и [ctype\_digit()](https://www.php.net/manual/ru/function.ctype-digit.php) соответственно) и заканчивая [Perl-совместимыми регулярными выражениями](https://www.php.net/manual/ru/ref.pcre.php).
* В случае, если приложение ожидает цифровой ввод, примените функцию [ctype\_digit()](https://www.php.net/manual/ru/function.ctype-digit.php) для проверки введенных данных, или принудительно укажите их тип при помощи [settype()](https://www.php.net/manual/ru/function.settype.php), или просто используйте числовое представление при помощи функции [sprintf()](https://www.php.net/manual/ru/function.sprintf.php). Пример #5 Более безопасная реализация постраничной навигации

<?php

settype($offset, 'integer');

$query = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";

// обратите внимание на формат %d, использование %s было бы бессмысленно

$query = sprintf("SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET %d;",

                 $offset);

?>

* Если на уровне базы данных не поддерживаются привязанные переменные, то всегда экранируйте любые нечисловые данные, используемый в запросах к БД при помощи специальных экранирующих функций, специфичных для используемой вами базы данных (например, [mysql\_real\_escape\_string()](https://www.php.net/manual/ru/function.mysql-real-escape-string.php), sqlite\_escape\_string() и т.д.). Общие функции такие как [addslashes()](https://www.php.net/manual/ru/function.addslashes.php) полезны только в определенных случаях (например MySQL в однобайтной кодировке с отключенным NO\_BACKSLASH\_ESCAPES), поэтому лучше избегать их использование.
* Ни в коем случае не выводите никакой информации о БД, особенно о ее структуре. Также ознакомьтесь с соответствующими разделами документации: "[Сообщения об ошибках](https://www.php.net/manual/ru/security.errors.php)" и "[Функции обработки и логирования ошибок](https://www.php.net/manual/ru/ref.errorfunc.php)".
* Вы можете использовать хранимые процедуры и заранее определенные курсоры для абстрагированной работы с данными, не предоставляя пользователям прямого доступа к данным и представлениям, но это решение имеет свои особенности.