Основные технологии на стороне сервера и клиента. CGI, PHP, ASP, JSP (можно небольшие примеры) основные характеристики технологий, используемых языков программирования и платформ (динамическая типизация, объектно-ориентированность и т.д., интерпретация или компиляция), особенности виртуальных машин CLR, JVM. Jit-компиляция. Фреймворки Node JS, django, ruby on rails. Java script, Dart, SWF, ACTIVEX, Haxe. CSRF – межсайтовая подделка запроса. Способы защиты. SQL инъекция, примеры. Какие еще виды атак на веб сайты вам известны.

**Термины**

**Компиляция** — трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду.

**Интерпретация** — пооператорный (покомандный, построчный) анализ, обработка и тут же выполнение исходной программы или запроса (в отличие от компиляции, при которой программа транслируется без её выполнения).

**Статически типизированные** языки проверяют типы и ищут ошибки типизации на стадии компиляции. **Динамически типизированные** языки проверяют типы и ищут ошибки типизации на стадии исполнения. Иными словами: статическое типизирование означает проверку типов перед запуском программы; динамическое — проверку типов пока программа запущена.

**Основные технологии?**

**CGI** (<https://www.youtube.com/watch?v=VWe0HcZTUmE>)

CGI (от англ. Common Gateway Interface — «общий интерфейс шлюза») — стандарт интерфейса, используемого для связи внешней программы с веб-сервером. Программу, которая работает по такому интерфейсу совместно с веб-сервером, принято называть шлюзом, хотя многие предпочитают названия «скрипт» (сценарий) или «CGI-программа».

Поскольку гипертекст статичен по своей природе, веб-страница не может непосредственно взаимодействовать с пользователем. До появления JavaScript, не было иной возможности отреагировать на действия пользователя, кроме как передать введенные им данные на веб-сервер для дальнейшей обработки. В случае CGI эта обработка осуществляется с помощью внешних программ и скриптов, обращение к которым выполняется через стандартизованный интерфейс — общий шлюз.

Сам интерфейс разработан таким образом, чтобы можно было использовать любой язык программирования, который может работать со стандартными устройствами ввода-вывода. Такими возможностями обладают даже скрипты для встроенных командных интерпретаторов операционных систем, поэтому в простых случаях могут использоваться даже командные скрипты.

Обобщенный алгоритм работы через CGI можно представить в следующем виде:

1. Клиент запрашивает CGI-приложение по его URI.
2. Веб-сервер принимает запрос и устанавливает переменные окружения, через них приложению передаются данные и служебная информация.
3. Веб-сервер перенаправляет запросы через стандартный поток ввода (stdin) на вход вызываемой программы.
4. CGI-приложение выполняет все необходимые операции и формирует результаты в виде HTML.
5. Сформированный гипертекст возвращается веб-серверу через стандартный поток вывода (stdout). Сообщения об ошибках передаются через stderr.
6. Веб-сервер передаёт результаты запроса клиенту.

Самым большим недостатком этой технологии являются повышенные требования к производительности веб-сервера. Дело в том, что каждое обращение к CGI-приложению вызывает порождение нового процесса, со всеми вытекающими отсюда накладными расходами. Если же приложение написано с ошибками, то возможна ситуация, когда оно, например, зациклится. Браузер прервет соединение по истечении тайм-аута, но на серверной стороне процесс будет продолжаться, пока администратор не снимет его принудительно.

**PHP**

PHP – скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов. Сам язык является интерпретируемым и использует динамическую типизацию.

Принцип, по которому PHP трансформирует веб-страницы на сервере перед отправкой браузеру, следующий. Он исполняет определённые схемы работы, в этом процессе изменяет или создает результат данного сценария, выраженный в HTML-коде. Сценарием называется программа на сервере, запускающаяся в ответ на запрос, который пришел от браузера. Затем PHP передает код браузеру, который не знает, каким образом страница была создана – верстальщиком (статично) или PHP (динамически). Но это и неважно, так как браузер будет работать со всем полученным от веб-сервера.

Именно поэтому новое определение PHP – препроцессор гипертекста. Любая запрошенная страница, имеющая адрес \*.php, заставляет браузер обратиться к PHP с запросом на выполнение сценария из файла, который расположен на этой ссылке.

Недостатки связаны с неоднозначностью кода, моделью работы с памятью, отсутствием статической типизации.

**ASP**

ASP (англ. Active Server Pages — «активные серверные страницы») — технология, предложенная компанией Microsoft в 1996 году для создания Web-приложений. Эта технология основана на внедрении в обыкновенные веб-страницы специальных элементов управления, допускающих программное управление. По своей сути, ASP — это технология динамического создания страниц на стороне сервера, приблизившая проектирование и реализацию Web-приложений к той модели, по которой проектируются и реализуются обычные приложения.

Для реализации приложений ASP используются языки сценариев (VBScript или JScript). Также допускается применение COM-компонентов. Технология ASP разработана для операционных систем из семейства Windows NT и функционирует под управлением веб-сервера Microsoft IIS.

Принцип работы. Вы пишите программу и складываете в файл на сервере. Браузер клиента запрашивает файл. Файл сначала интерпретируется сервером, на выходе производится HTML-код. Этот HTML посылается клиенту. Файлы с программами имеют расширение .asp. Файлы asp – это обычные текстовые файлы, содержащие исходные тексты программ. Файлы делаются с помощью любого текстового редактора. Каталог, в котором размещены файлы asp должен иметь права на выполнение, так как сервер исполняет эти файлы, когда браузер их запрашивает.

Технология ASP получила своё развитие в виде ASP.NET — технологии создания веб-приложений, основанной уже на платформе Microsoft .NET.

**ASP.NET**

ASP.NET (Active Server Pages для .NET) — платформа разработки веб-приложений, в состав которой входят: веб-сервисы, программная инфраструктура, модель программирования, от компании Майкрософт. ASP.NET входит в состав платформы .NET Framework.

ASP.NET внешне во многом сохраняет схожесть с более старой технологией ASP, что позволяет разработчикам относительно легко перейти на ASP.NET. В то же время внутреннее устройство ASP.NET существенно отличается от ASP, поскольку она основана на платформе .NET и, следовательно, использует все новые возможности, предоставляемые этой платформой.

Поскольку ASP.NET основывается на Common Language Runtime (CLR), которая является основой всех приложений Microsoft .NET, разработчики могут писать код для ASP.NET, используя языки программирования, входящие в комплект .NET Framework (C#, Visual Basic.NET, J# и JScript .NET).

**JSP**

JSP (Java Server Pages) — это технология Java, которая позволяет создавать динамические веб-страницы для Java приложений. Страница JSP содержит текст двух типов: статические исходные данные и JSP-элементы, которые конструируют динамическое содержимое. Кроме этого, могут использоваться библиотеки JSP-тегов, а также Expression Language (EL) для внедрения Java-кода в статичное содержимое JSP-страниц.

Сам по себе язык Java является объектно-ориентированным и строго типизированным, освобождение выделенной памяти под объект реализуется автоматически сборщиком мусора, программа предварительно транслируется в промежуточный байт-код, а затем исполняется jit-компилятором на лету машиной Java JVM (Java virtual machine). Это делает данную технологию платформо-независимой.

Код JSP-страницы транслируется в Java-код сервлета с помощью компилятора JSP-страниц Jasper, и затем компилируется в байт-код виртуальной машины Java (JVM). Контейнеры сервлетов, способные исполнять JSP-страницы, написаны на платформенно-независимом языке Java. Данные файлы обрабатываются на сервере, в результате чего все JSP теги преобразуются в html теги, и на выходе получается обычная html страница.

Технология JSP является платформенно-независимой, переносимой и легко расширяемой технологией для разработки веб-приложений.

В общем случае под JSP понимаются динамические веб-страницы, в которых динамическая часть генерируется с помощью Java, а статическая формируется за счет языков разметки, чаще всего HTML. Такая страница представляет собой текстовый документ с расширением .jsp, написанный на одном из языков разметки (таких как HTML, SVG, WML и XML), c вкраплениями JSP элементов (или jsp тегов). Внутри данных тегов содержатся обращения к серверному коду (данным), а также выполняются некоторые вычисления.

**Виртуальные машины**

**CLR**

Common Language Runtime (англ. CLR — общеязыковая исполняющая среда) — исполняющая среда для байт-кода CIL (MSIL), в который компилируются программы, написанные на .NET-совместимых языках программирования (C#, Managed C++, Visual Basic .NET, F# и прочие). CLR является одним из основных компонентов пакета Microsoft .NET Framework.

Среда CLR является реализацией спецификации CLI (англ. Common Language Infrastructure), спецификации общеязыковой инфраструктуры компании Microsoft.

**JVM**

Java Virtual Machine (JVM) — виртуальная машина Java — основная часть исполняющей системы Java, так называемой Java Runtime Environment (JRE). Виртуальная машина Java исполняет байт-код Java, предварительно созданный из исходного текста Java-программы компилятором Java (javac). JVM может также использоваться для выполнения программ, написанных на других языках программирования. Например, исходный код на языке Ada может быть скомпилирован в байт-код Java, который затем может выполниться с помощью JVM.

JVM является ключевым компонентом платформы Java. Так как виртуальные машины Java доступны для многих аппаратных и программных платформ, Java может рассматриваться и как связующее программное обеспечение, и как самостоятельная платформа. Использование одного байт-кода для многих платформ позволяет описать Java как «скомпилируй единожды, запускай везде» (compile once, run anywhere).

Виртуальные машины Java обычно содержат интерпретатор байт-кода, однако, для повышения производительности во многих машинах также применяется JIT-компиляция часто исполняемых фрагментов байт-кода в машинный код.

**JIT-компиляция**

JIT-компиляция (англ. Just-in-Time, компиляция «точно в нужное время»), динамическая компиляция (англ. dynamic translation) — технология увеличения производительности программных систем, использующих байт-код, путём компиляции байт-кода в машинный код или в другой формат непосредственно во время работы программы. Таким образом достигается высокая скорость выполнения по сравнению с интерпретируемым байт-кодом (сравнимая с компилируемыми языками) за счёт увеличения потребления памяти (для хранения результатов компиляции) и затрат времени на компиляцию. Технология JIT базируется на двух более ранних идеях, касающихся среды выполнения: компиляции байт-кода и динамической компиляции.

Так как JIT-компиляция является, по сути, одной из форм динамической компиляции, она позволяет применять такие технологии, как адаптивная оптимизация и динамическая рекомпиляция. Благодаря этому JIT-компиляция может показывать лучшие результаты в плане производительности, чем статическая компиляция. Интерпретация и JIT-компиляция особенно хорошо подходят для динамических языков программирования, при этом среда исполнения справляется с поздним связыванием типов и гарантирует безопасность исполнения.

JIT-компиляция может быть применена как ко всей программе, так и к её отдельным частям. Например, текстовый редактор может на лету компилировать регулярные выражения для более быстрого поиска по тексту. JIT используется в реализациях Java (JRE), JavaScript, .NET Framework, в одной из реализаций Python — PyPy. Существующие наиболее распространённые интерпретаторы языков PHP, Ruby, Perl, Python и им подобных, имеют ограниченные или неполные JIT.

Простыми словами, механизм Just-In-Time компиляции заключается в следующем: если в программе присутствуют части кода, которые выполняются много раз, то их можно скомпилировать один раз в машинный код, чтобы в будущем ускорить их выполнение. После компиляции такой части программы в машинный код, при каждом следующем вызове этой части программы виртуальная машина будет сразу выполнять скомпилированный машинный код, а не интерпретировать его, что естественно ускорит выполнение программы.

Ускорение работы программы достигается за счет увеличения потребления памяти (где-то же нам нужно хранить скомпилированный машинный код) и за счет увеличения временных затрат на компиляцию во время исполнения программы.

JIT компиляция — довольно сложный механизм. Всего существует 4 уровня JIT компиляции байт-кода в машинный код. Чем выше уровень компиляции, тем он сложнее, но и одновременно выполнение такого участка будет быстрее, чем участка с меньшим уровнем. JIT — компилятор самостоятельно решает, какой уровень компиляции задать для каждого фрагмента программы на основе того, как часто выполняется этот фрагмент. Под капотом **JVM** использует 2 JIT-компилятора — C1 и C2. C1 компилятор так же называется клиентским компилятором и способен скомпилировать код только до 3-его уровня. За 4-ый, самый сложный и быстрый уровень компиляции отвечает компилятор C2.

**Фреймворки**

**Node.js**

Node или Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения и даже программировать микроконтроллеры. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Раньше вы могли запустить JavaScript только в браузере, но однажды разработчики расширили его, и теперь вы можете запускать JS на своем компьютере в качестве отдельного приложения. Так появился Node.js.

Главная особенность Node.js — то, что вместо традиционной модели параллелизма на основе потоков автор выбрал событийно-ориентированный подход. Это такая парадигма программирования, в которой выполнение программ определяется событиями. Например, это действие пользователя, поступление сетевого пакета, сообщение из другой программы и т.д.

Такой подход сильно упрощает программирование некоторых приложений, особенно при реализации удобного интерфейса ввода-вывода (I/O), как у Node.js.

С распространением в интернете социальных сетей и других «интерактивных» сайтов резко выросла востребованность Node.js как платформы для приложений, реагирующих на действия пользователя: чатов, игр, инструментов совместной работы — теперь всё это делают на JavaScript с помощью Node.js.

Таким образом, Node.js — это событийно-ориентированный I/O фреймворк на JavaScript. Технические особенности Node.js делают приложения на его основе легковесными и эффективными. Этим и объясняется его огромная популярность.

**Django**

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Архитектура MVC позволяет разработчику работать с визуальным представлением и бизнес-логикой приложения отдельно.

Кстати, при работе с Django специалисты чаще используют термин MVT — Model-View-Template или модель-представление-шаблон. Компоненты MVT можно использовать независимо друг от друга. Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails).

Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself). Это принцип разработки программного обеспечения, нацеленный на снижение повторения информации различного рода, особенно в системах со множеством слоёв абстрагирования. Принцип DRY формулируется как: «Каждая часть знания должна иметь единственное, непротиворечивое и авторитетное представление в рамках системы. Когда принцип DRY применяется успешно, изменение единственного элемента системы не требует внесения изменений в другие, логически не связанные элементы. Те элементы, которые логически связаны, изменяются предсказуемо и единообразно.

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений. Регулярное выражение в Python – это последовательность символов, описывающая текстовые шаблоны. Используя регулярные выражения, мы можем сопоставлять входные данные с определенными шаблонами (например, поиск), извлекать совпадающие строки (фильтрация, разделение), а также заменять вхождения шаблонов подстановками, и все это с минимальным объемом кода.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных. ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение, или преобразование) — технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

**Ruby on Rails**

…

**Инструменты для оживления сайта**

**JavaScript** (<https://blog.skillfactory.ru/glossary/javascript/>)

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

JavaScript это язык, который позволяет вам применять сложные вещи на web странице — каждый раз, когда на web странице происходит что-то большее, чем просто её статичное отображение — отображение периодически обновляемого контента, или интерактивных карт, или анимация 2D/3D графики, или прокрутка видео в проигрывателе, и т.д. — можете быть уверены, что скорее всего, не обошлось без JavaScript. Это третий слой слоёного пирога стандартных web технологий, два из которых HTML и CSS.

JS поддерживают все популярные браузеры. Во frontend-части сайтов язык используют для создания интерактива (анимаций, всплывающих форм, автозаполнения), так как он связан с HTML и CSS и может ими манипулировать. В backend-части с языком JavaScript работают на платформе Node.js. С ее помощью, например, разрабатывают серверные веб-приложения и подключают библиотеки.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

JavaScript называют языком скриптов или сценариев. Скрипты — это набор инструкций, которые выполняются при загрузке страницы. Браузер самостоятельно интерпретирует код на JavaScript, для этого даже не требуется компиляция (перевод языка программирования в машинный код).

Скрипты можно прописать внутри кода страницы или подключить к HTML отдельным файлом. Например, в стандартной разметке index.html JS-код прописывают внутри тега script, помещенного в тег body.

Веб-страницы частично обрабатываются с помощью JavaScript на компьютере пользователя. Это снижает нагрузку на сервер: часть операций выполняется без запросов к нему, что экономит время и трафик.

**Dart** (<https://gb.ru/posts/dart_it>)

Dart — язык программирования, созданный Google. Dart позиционируется в качестве замены/альтернативы JavaScript. Один из разработчиков языка написал, что JavaScript «имеет фундаментальные изъяны», которые невозможно исправить. Поэтому и был создан Dart.

Изначально было предложено два способа исполнения Dart-программ: с использованием виртуальной машины (в поддерживающих язык браузерах) или с промежуточной трансляцией в javascript (универсальный).

Теоретически, знание TypeScript совместно с JavaScript практически полностью покрывает возможности Dart. С другой стороны, потенциал Dart куда выше и его стоит рассматривать скорее, как этап развития, а не как конкурента JavaScript.

Dart поддерживается всеми популярными IDE: IDEA, WebStorm, Atom, Emacs, Visual Studio, Sublime Text, Vim. Кроме того, для удобства использования Dart Google специально выпустил браузер Dartium, на основе Chromium, со встроенной виртуальной машиной.

**SWF**

SWF (ранее Shockwave Flash, теперь Small Web Format) — формат файла для флеш-анимации, векторной графики, видео и аудио в сети Интернет. Картинка, сохранённая в этом формате, масштабируется без видимых искажений, видеоролик имеет небольшой размер, происходит более быстрая загрузка видео файла и его воспроизведение.

Сфера использования SWF различна, это могут быть игры, веб-сайты, CD презентации, баннеры и просто мультфильмы. При создании программного обеспечения можно использовать медиа, звуковые и графические файлы, можно создавать интерактивные интерфейсы и полноценные веб-приложения с использованием PHP и XML.

Файлы SWF можно открыть программой Adobe Flash Player или любым web-браузером, поддерживающим Flash. Для воспроизведения Flash-приложений браузерами иногда необходимо установить подключаемый модуль Adobe Flash (если он не включён в установочный пакет).

Плагин Shockwave Flash — программный модуль, реализующий в браузерах функцию Adobe Shockwave — мультимедийной программной платформы, включающей проигрыватель Shockwave Player и инструментарий создания контента Adobe Director.

До 2021 года для открытия его на компьютерах использовалась утилита Adobe Flash Player, но 31 декабря 2020 года поддержка Флеш Плеера была прекращена, и проект закрыт.

**ACTIVEX**

ActiveX – технология, на основе которой создаются компоненты для программирования сайтов под Internet Explorer. Эти компоненты по своей сути являются обычными программами, с одной лишь разницей в том, что запускаются они не пользователем, а операционной системой, и исполняются только в браузере.

Приложения ActiveX называются управляющими элементами. Они существенно расширяют функционал сайта, чем активно пользуются различные компании. Наиболее вероятно «встретить» элементы управления ActiveX в корпоративном сегменте интернет-банков. С их помощью скачиваются и устанавливаются программы обеспечения безопасности финансовых операций, а в браузере отображаются диалоговые окна ввода пин-кодов, электронно-цифровые подписи и различные функции отправки бухгалтерской отчетности. Помимо этих сугубо деловых функций с помощью таких компонентов на любой сайт можно вывести аудио и видео флеш плеер ActiveX, открывать различные файлы с помощью браузера и проигрывать анимацию.

С одной стороны, технологию ActiveX поддерживает только Internet Explorer, а большинство опытных пользователей Интернета пользуется другими браузерами и не подозревает об этой технологии.

Но если Вы в своей жизни вынуждены использовать какие-либо сайты, в обязательном порядке требующие использование старых версий IE, то вам просто необходимо использовать компоненты ActiveX.

Однако пользоваться возможностями данной технологии надо предельно аккуратно. Существует довольно высокая вероятность «подцепить» вирус разрешив всем сайтам по умолчанию устанавливать приложения ActiveX. Злоумышленники применяют эту технологию для распространения шпионских программ, поэтому нам следует разобраться, как правильно настроить работу данного приложения.

**Haxe**

Haxe — это высокоуровневый кросс-платформенный язык программирования с открытым исходным кодом, а также компилятор, с помощью которого можно создавать приложения и генерировать исходный код для разных платформ, сохраняя единую кодовую базу.

Haxe включает в себя функциональность, поддерживаемую на всех платформах, например: числовые типы данных, строки, массивы, а также поддержку некоторых файловых форматов (xml, zip). Haxe также включает в себя поддержку специфических API для каждой целевой платформы компилятора.

Код, написанный на языке Haxe, может быть скомпилирован в код JavaScript, C++, Java, JVM, PHP, C#, Python, Lua и Node.js. Haxe-код также компилируется в SWF, HashLink и Neko, байт-код, а так же может быть выполнен в режиме интерпретации.

Основные пользователи Haxe — это TiVo, Prezi, Nickelodeon, Disney, Mattel, Hasbro, Coca Cola, Toyota и BBC. OpenFL и Flambe — популярные фреймворки Haxe для создания мультиплатформенного контента и программ из единой кодовой базы. В связи с всё большим вытеснением технологии Adobe Flash в последние годы в пользу HTML5, Haxe, Unity и другие кроссплатформенные инструменты уделяют последнему всё больше времени, сохраняя обратную поддержку с Adobe Flash Player.

**Защита**

**CSRF**

CSRF (англ. cross-site request forgery — «межсайтовая подделка запроса», также известна как XSRF) — вид атак на посетителей веб-сайтов, использующий недостатки протокола HTTP. Если жертва заходит на сайт, созданный злоумышленником, от её лица тайно отправляется запрос на другой сервер (например, на сервер платёжной системы), осуществляющий некую вредоносную операцию (например, перевод денег на счёт злоумышленника). Для осуществления данной атаки жертва должна быть аутентифицирована на том сервере, на который отправляется запрос, и этот запрос не должен требовать какого-либо подтверждения со стороны пользователя, которое не может быть проигнорировано или подделано атакующим скриптом.

Основное применение CSRF — вынуждение выполнения каких-либо действий на уязвимом сайте от лица жертвы (изменение пароля, секретного вопроса для восстановления пароля, почты, добавление администратора и т. д.). Также с помощью CSRF возможна эксплуатация отражённых XSS, обнаруженных на другом сервере.

**Способы защиты**

Защищаться должны все запросы, изменяющие данные на сервере, а также запросы, возвращающие персональные или иные чувствительные данные.

Наиболее простым способом защиты от данного типа атак является механизм, когда веб-сайты должны требовать подтверждения большинства действий пользователя и проверять поле HTTP\_REFERER, если оно указано в запросе. Но этот способ может быть небезопасен, и использовать его не рекомендуется.

Другим распространённым способом защиты является механизм, при котором с каждой сессией пользователя ассоциируется дополнительный секретный уникальный ключ, предназначенный для выполнения запросов. Секретный ключ не должен передаваться в открытом виде, например, для POST-запросов ключ следует передавать в теле запроса, а не в адресе страницы. Браузер пользователя посылает этот ключ в числе параметров каждого запроса, и перед выполнением каких-либо действий сервер проверяет этот ключ. Преимуществом данного механизма, по сравнению с проверкой Referer, является гарантированная защита от атак CSRF. Недостатками же являются требование возможности организации пользовательских сессий, требование динамической генерации HTML-кода страниц сайта, а также необходимость защиты от XSS и других атак, позволяющих получить злоумышленнику уникальный ключ.

Спецификация протокола HTTP/1.1 определяет безопасные методы запросов, такие как GET, HEAD, которые не должны изменять данные на сервере. Для таких запросов, при соответствии сервера спецификации, нет необходимости применять защиту от CSRF.

Может возникнуть желание подстраховаться и добавить ключ в каждый запрос, но следует иметь в виду, что спецификация HTTP/1.1 допускает наличие тела для любых запросов, но для некоторых методов запроса (GET, HEAD, DELETE) семантика тела запроса не определена и должна быть проигнорирована. Поэтому ключ может быть передан только в самом URL или в HTTP-заголовке запроса. Необходимо защитить пользователя от неблагоразумного распространения ключа в составе URL, например, на форуме, где ключ может оказаться доступным злоумышленнику. Поэтому запросы с ключом в URL не следует использовать в качестве адреса для перехода, то есть исключить переход по такому адресу клиентским скриптом, перенаправлением сервера, действием формы, гиперссылкой на странице и т. п. с целью сокрытия ключа, входящего в URL. Их можно использовать лишь как внутренние запросы скриптом с использованием XMLHttpRequest или обёрткой, например AJAX.

Существенен факт того, что ключ (CSRF-токен) может быть предназначен не для конкретного запроса или формы, а для всех запросов пользователя вообще. Поэтому достаточно утечки CSRF-токена c URL, выполняющего простое действие или не выполняющего действие вовсе, как защиты от подделки запроса лишается любое действие, а не только то, с которым связан ставший известным URL.

Существует более жёсткий вариант предыдущего механизма, в котором с каждым действием ассоциируется уникальный одноразовый ключ. Такой способ более сложен в реализации и требователен к ресурсам. Способ используется некоторыми сайтами и порталами, такими как Livejournal, Rambler и др. На 2016 год не было сведений о преимуществе более жёсткого варианта по сравнению с вариантом, в котором используется единственный для каждой сессии секретный ключ.

**SQL-инъекции**

Внедрение SQL-кода — один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.

Внедрение SQL, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.

Атака типа внедрения SQL может быть возможна из-за некорректной обработки входных данных, используемых в SQL-запросах.

Разработчик прикладных программ, работающих с базами данных, должен знать о таких уязвимостях и принимать меры противодействия внедрению SQL.

Существует три основных класса атак, основанных на внедрении SQL-кода:

1. Классическая SQL инъекция (Classic SQLi)

2. SQL инъекция, основанная на эксплуатации выводимых СУБД сообщений об ошибках (Error-based SQLi)

3. Слепая SQL инъекция (Blind SQLi)