Безопасность в Интернете

Веб-сайты содержат несколько различных типов информации. Некоторые из них не являются конфиденциальными, например, копия, показанная на общедоступных страницах. Некоторые из них являются конфиденциальными, например имена пользователей, пароли и банковская информация, или внутренние алгоритмы и частная информация о продукте.

Конфиденциальная информация должна быть защищена, и на этом сосредоточена веб-безопасность. Если эта информация попадет не в те руки, она может быть использована для:

Поставьте компании в невыгодное положение, поделившись своей информацией с конкурентами. Отключите или захватите их службы, что снова вызовет серьезные проблемы с их работой. Подвергают риску конфиденциальность своих клиентов , делая их уязвимыми для профилирования, таргетинга, потери данных, кражи личных данных или даже финансовых потерь. Современные браузеры уже имеют несколько функций для защиты безопасности пользователей в Интернете, но разработчикам также необходимо применять передовые методы и тщательно кодировать, чтобы обеспечить безопасность своих веб-сайтов. Даже простые ошибки в вашем коде могут привести к уязвимостям, которые злоумышленники могут использовать для кражи данных и получения контроля над службами, для которых у них нет авторизации.

В этой статье представлено введение в веб-безопасность, включая информацию, которая поможет вам понять аспекты, которые могут сделать веб-сайты уязвимыми, и способы их защиты.

Связь между безопасностью и конфиденциальностью Безопасность и конфиденциальность — разные темы, но они также тесно связаны. Стоит знать разницу между ними и то, как они соотносятся.

Безопасность — это процесс защиты личных данных и систем от несанкционированного доступа. Сюда входят как данные компании (внутренние), так и данные пользователей и партнеров (внешние). Конфиденциальность означает предоставление пользователям права контролировать сбор, хранение и использование их данных, а не их безответственное использование. Например, вы должны четко сообщать пользователям ваших сайтов, какие данные вы собираете, кому они будут переданы и как они будут использоваться. Пользователям должна быть предоставлена возможность согласиться с вашей политикой конфиденциальности, получить доступ к своим данным, которые вы сохранили, и удалить свои данные, если они больше не хотят, чтобы они у вас были. Хорошая безопасность очень важна для хорошей конфиденциальности. Вы можете следовать всем советам, перечисленным в нашем руководстве по конфиденциальности в Интернете , но действовать добросовестно и иметь надежную политику конфиденциальности бесполезно, если ваш сайт небезопасен, а злоумышленники все равно могут просто украсть данные.

Службы безопасности, предоставляемые браузерами Веб-браузеры имеют строгую модель безопасности, обеспечивающую высокий уровень безопасности основного контента, соединений и транспорта. В этом разделе рассматриваются основы.

Политика одинакового происхождения и CORS Политика того же источника — это фундаментальный механизм безопасности в Интернете, который ограничивает взаимодействие документа или сценария, загруженного из одного источника , с ресурсом из

другого источника. Это помогает изолировать потенциально вредоносные документы, уменьшая возможные векторы атак.

Как правило, документы из одного источника не могут делать запросы к другим источникам. Это имеет смысл, потому что вы не хотите, чтобы сайты могли мешать друг другу и получать доступ к вещам, которым они не должны. В некоторых случаях имеет смысл ослабить это ограничение; например, у вас может быть несколько сайтов, которые взаимодействуют друг с другом, и вы можете захотеть, чтобы эти сайты запрашивали ресурсы друг у друга, например, используя файлы fetch().

Это может быть разрешено с помощью Cross-Origin Resource Sharing (CORS) — механизма на основе HTTP-заголовков, который позволяет серверу указывать любые источники (домен, схему или порт), отличные от его собственных, из которых браузер должен разрешать загрузку ресурсов.

HTTP-модель для связи Протокол HTTP используется веб-браузерами и серверами для связи друг с другом, запроса ресурсов, предоставления ответов (например, предоставления запрошенного ресурса или детализации причины сбоя запроса) и обеспечения функций безопасности для этой связи.

Безопасность транспортного уровня (TLS) обеспечивает безопасность и конфиденциальность путем шифрования данных во время передачи по сети и является технологией, лежащей в основе протокола HTTPS . TLS хорош для конфиденциальности, поскольку не позволяет третьим сторонам перехватывать передаваемые данные и использовать их злонамеренно.

Все браузеры по умолчанию требуют HTTPS; это уже практически так, потому что без этого протокола в сети мало что можно сделать.

Похожие темы:

Безопасность транспортного уровня (TLS) Протокол Transport Layer Security (TLS) — это стандарт, позволяющий двум сетевым приложениям или устройствам безопасно и конфиденциально обмениваться информацией. Приложения, использующие TLS, могут выбирать свои параметры безопасности, что может существенно повлиять на безопасность и надежность данных. В этой статье представлен обзор TLS и типов решений, которые вам необходимо принять для защиты вашего контента.

HTTP Strict-Transport-Security Заголовок Strict-Transport-Security: HTTP позволяет веб-сайту указать, что к нему можно получить доступ только с использованием HTTPS.

Прозрачность сертификата Прозрачность сертификатов — это открытая платформа, предназначенная для защиты от неправильной выдачи сертификатов и отслеживания таких случаев. Вновь выпущенные сертификаты «записываются» в общедоступные, часто независимые журналы CT, которые поддерживают только добавление, криптографически гарантированную запись о выпущенных сертификатах TLS.

Смешанное содержание Страница HTTPS, которая включает содержимое, полученное с использованием незашифрованного текста HTTP, называется страницей со смешанным содержимым . Подобные страницы зашифрованы лишь частично, поэтому незашифрованный контент остается доступным для снифферов и злоумышленников.

Как исправить сайт с заблокированным смешанным контентом Если ваш веб-сайт предоставляет страницы HTTPS, весь активный смешанный контент, доставляемый через HTTP на этих страницах, будет заблокирован по умолчанию. Следовательно, ваш веб-сайт может показаться пользователям неработающим (если не загружаются iframe или плагины и т. д.). Пассивное смешанное содержимое отображается по умолчанию, но пользователи могут настроить блокировку и этого типа содержимого. На этой странице объясняется, что вы должны знать как веб-разработчик.

Слабые алгоритмы подписи Надежность алгоритма хеширования, используемого при подписании цифрового сертификата, является критическим элементом безопасности сертификата. В этой статье содержится некоторая информация об известных алгоритмах подписи, поэтому вы можете избегать их, когда это уместно.

Безопасные контексты и разрешения функций Браузеры контролируют использование нескольких «мощных функций» несколькими способами. Под «мощными функциями» мы подразумеваем такие вещи, как создание системных уведомлений сайтом, использование веб-камеры пользователя для доступа к медиапотоку, управление системным графическим процессором и использование веб-платежей. Если бы сайт мог просто без ограничений использовать API-интерфейсы, управляющие такими функциями, разработчики-злоумышленники могли бы, например:

Раздражайте пользователей ненужными уведомлениями и другими функциями пользовательского интерфейса. Засоряют их браузер/систему для создания атак типа «отказ в обслуживании» (DoS). Украсть данные или деньги. Такие функции браузера управляются следующим образом:

Во-первых, использование таких функций разрешено только в безопасных контекстах . Безопасный контекст — это a windowили a worker, для которого есть достаточная уверенность в том, что контент доставлен безопасно (через HTTPS/TLS). В безопасном контексте возможности для связи с контекстами, которые не являются безопасными, ограничены. Безопасные контексты также помогают предотвратить доступ злоумышленников к мощным API-интерфейсам браузера.

Примечание. См. также раздел Функции, ограниченные безопасным контекстом . В этом справочнике перечислены функции веб-платформы, доступные только в безопасном контексте.

Во-вторых, использование этих функций ограничено системой пользовательских разрешений — пользователи должны явно согласиться на предоставление доступа к таким функциям, а это означает, что эти функции не могут использоваться автоматически. Запросы разрешений пользователя выполняются автоматически, но вы можете запросить состояние разрешения API с помощью API разрешений .

В-третьих, многие другие функции браузера могут использоваться только в ответ на действия пользователя, такие как нажатие кнопки, а это означает, что их нужно вызывать из соответствующего обработчика событий. Это часто называют временной активацией . Дополнительную информацию см. в разделе «Функции, закрытые активацией пользователя» .

Вопросы безопасности для разработчиков на стороне клиента Есть много аспектов веб-безопасности, о которых необходимо подумать на стороне сервера и клиента. В следующих разделах основное внимание уделяется вопросам безопасности на стороне клиента. Вы можете найти полезный обзор безопасности с точки зрения сервера, который также включает описания распространенных атак, на которые следует обратить внимание, в разделе «

Безопасность веб-сайта» (часть нашего модуля обучения программированию веб-сайта на стороне сервера).

Храните данные на стороне клиента ответственно Ответственное обращение с данными в значительной степени связано с сокращением использования файлов cookie и осторожностью в отношении данных, которые вы в них храните. Традиционно веб-разработчики использовали файлы cookie для хранения всех видов данных для самых разных целей, и злоумышленникам было легко использовать эту тенденцию. В результате браузеры начали ограничивать действия, которые вы можете выполнять с помощью межсайтовых файлов cookie, с целью полностью закрыть к ним доступ в будущем.

Вам следует подготовиться к удалению межсайтовых файлов cookie, ограничив количество действий по отслеживанию, на которые вы полагаетесь, и/или реализуя желаемое сохранение информации другими способами.

Например:

Используйте альтернативный механизм хранения на стороне клиента, например API веб-хранилища, для сохранения данных. Недостатком веб-хранилища является то, что данные хранятся для каждого источника, поэтому ими нельзя поделиться. Обратите внимание, что веб-хранилище имеет два варианта — sessionStorageи localStorage. Мы рекомендуем использовать его sessionStorageдля дополнительной безопасности, так как данные сохраняются только в течение времени жизни окна или вкладки, в которой они существуют. localStorageДанные сохраняются даже после закрытия и повторного открытия окна или вкладки. Это означает, что существует большая вероятность того, что он попадет в чужие руки, например, на общую рабочую станцию. Используйте такие технологии, как IFrame без учетных данных , собственные наборы и/или API доступа к хранилищу , чтобы разрешить вашим сайтам разрешать использование межсайтовых файлов cookie безопасным и контролируемым способом или полностью их блокировать. В настоящее время эти технологии имеют ограниченную поддержку браузеров. Используйте решение на стороне сервера для сохранения данных. Ознакомьтесь с нашим руководством по конфиденциальности и, в частности, с сокращением отслеживающих файлов cookie , чтобы узнать больше об этом.

Защитите личность пользователя и управляйте логинами При внедрении безопасного решения, включающего сбор данных, особенно если данные являются конфиденциальными, например учетные данные для входа в систему, имеет смысл использовать надежное решение от уважаемого поставщика. Например, любая респектабельная серверная среда будет иметь встроенные функции для защиты от распространенных уязвимостей. Вы также можете рассмотреть возможность использования специализированного продукта для своих целей, например решения поставщика удостоверений или поставщика безопасных онлайн-опросов.

Если вы хотите внедрить собственное решение для сбора пользовательских данных, убедитесь, что вы понимаете все аспекты и требования. Наймите опытного разработчика серверной части и/или инженера по безопасности для внедрения системы и убедитесь, что она тщательно протестирована. Используйте многофакторную аутентификацию (MFA), чтобы обеспечить лучшую защиту. Рассмотрите возможность использования выделенного API, такого как веб-аутентификация или интегрированное управление учетными данными , для оптимизации клиентской части приложения.

Вот еще несколько советов по обеспечению безопасного входа в систему:

При сборе информации для входа в систему используйте надежные пароли, чтобы нельзя было легко угадать данные вашей учетной записи пользователя. Слабые пароли являются одной из основных причин нарушений безопасности. Кроме того, поощряйте своих пользователей использовать менеджер паролей, чтобы они могли использовать более сложные пароли, не беспокоясь о том, чтобы запомнить их, и не создавали угрозы безопасности, записывая их. См. также нашу статью о небезопасных паролях . Вы также должны информировать своих пользователей о фишинге . Фишинг — это действие по отправке пользователю сообщения (например, по электронной почте или SMS), содержащего ссылку на сайт, который выглядит как сайт, который он посещает каждый день, но на самом деле им не является. Ссылка сопровождается сообщением, предназначенным для того, чтобы заставить пользователей ввести свое имя пользователя и пароль на сайте, чтобы они могли быть украдены, а затем использованы злоумышленником в злонамеренных целях. Примечание. Некоторые фишинговые сайты могут быть очень сложными, и их трудно отличить от настоящего веб-сайта. Поэтому вам следует научить своих пользователей не доверять случайным ссылкам в электронных письмах и SMS-сообщениях. Если они получают сообщение типа «Срочно, вам нужно войти в систему сейчас, чтобы решить проблему», они должны перейти на сайт непосредственно в новой вкладке и попробовать войти напрямую, а не щелкать ссылку в сообщении. Или они могут позвонить или написать вам по электронной почте, чтобы обсудить полученное сообщение.

Защитите от атак методом грубой силы на страницы входа с помощью ограничения скорости , блокировки учетной записи после определенного количества неудачных попыток и проверки CAPTCHA . Управляйте сеансами входа пользователей с уникальными идентификаторами сеансов и автоматически отключайте пользователей после периодов бездействия. Не включайте конфиденциальные данные в строки запроса URL Как правило, вы не должны включать конфиденциальные данные в строки запроса URL : если третья сторона перехватит URL-адрес (например, через Refererзаголовок HTTP), они могут украсть эту информацию. Еще более серьезным является то, что эти URL-адреса могут быть проиндексированы общедоступными поисковыми роботами, HTTP-прокси и инструментами архивирования, такими как интернет- архив , а это означает, что ваши конфиденциальные данные могут храниться на общедоступных ресурсах.

Используйте POSTзапросы, а не GETзапросы, чтобы избежать этих проблем. В нашей статье Политика заголовков Referer: вопросы конфиденциальности и безопасности более подробно описываются риски конфиденциальности и безопасности, связанные с Refererзаголовком, и предлагаются советы по снижению этих рисков.

Примечание. Отказ от передачи конфиденциальных данных в URL-адресах через GETзапросы также помогает защититься от подделки межсайтовых запросов и атак повторного воспроизведения .

Применение политик использования Рассмотрите возможность использования таких инструментов, как политика безопасности контента (CSP) и политика разрешений , чтобы обеспечить использование набора функций и ресурсов на вашем сайте, что затрудняет внедрение уязвимостей.

CSP позволяет вам добавить уровень безопасности, например, разрешив загрузку изображений или сценариев только из определенных надежных источников. Это помогает обнаруживать и смягчать определенные типы атак, включая межсайтовый скриптинг (XSS) и атаки с внедрением данных. Эти атаки включают в себя ряд вредоносных действий, включая кражу данных, порчу сайта и распространение вредоносных программ.

Политика разрешений работает аналогичным образом, за исключением того, что она больше связана с разрешением или блокировкой доступа к определенным «мощным функциям» (как упоминалось ранее).

Примечание. Такие политики очень полезны для обеспечения безопасности сайтов, особенно если вы используете много стороннего кода на своем сайте. Однако имейте в виду, что если вы заблокируете использование функции, от которой зависит работа стороннего скрипта, вы можете нарушить функциональность вашего сайта.

Поддерживать целостность данных Следуя предыдущему разделу, когда вы разрешаете использование функций и ресурсов на своем сайте, вы должны попытаться убедиться, что ресурсы не были подделаны.

Похожие темы:

Целостность субресурсов Целостность подресурсов (SRI) — это функция безопасности, которая позволяет браузерам проверять, что ресурсы, которые они извлекают (например, из CDN) , доставляются без непредвиденных манипуляций. Он работает, позволяя вам предоставить криптографический хэш, которому должен соответствовать извлеченный ресурс.

HTTP Access-Control-Allow-Origin Заголовок Access-Control-Allow-Originответа указывает, может ли ответ использоваться совместно с кодом запроса из данного источника .

HTTP X-Content-Type-Options HTTP-заголовок ответа X-Content-Type-Options— это маркер, используемый сервером для указания того, что типы MIME , объявленные в Content-Typeзаголовках, не должны изменяться и им следует следовать. Это способ отказаться от прослушивания MIME-типов , или, другими словами, сказать, что MIME-типы настроены преднамеренно.

Дезинфекция ввода формы Как правило, не доверяйте ничему, что пользователи вводят в формы. Заполнять онлайн-формы сложно и утомительно, и пользователи могут легко ввести неправильные данные или данные в неправильном формате. Кроме того, злоумышленники хорошо разбираются в искусстве ввода определенных строк исполняемого кода в поля формы (например, SQL или JavaScript). Если вы не будете осторожны с обработкой этих типов входных данных, он может либо выполнить вредоносный код на вашем сайте, либо удалить ваши базы данных. См. SQL-инъекцию для хорошего примера того, как это может произойти.

Чтобы защититься от этого, вы должны тщательно дезинфицировать данные, введенные в ваши формы:

Вы должны реализовать проверку на стороне клиента, чтобы информировать пользователей о том, что они ввели данные в неправильном формате. Вы можете сделать это, используя встроенные функции проверки формы HTML, или вы можете написать свой собственный код проверки. Дополнительную информацию см. в разделе Проверка формы на стороне клиента . Вы должны использовать кодировку вывода при отображении пользовательского ввода в пользовательском интерфейсе приложения, чтобы безопасно отображать данные точно так, как их ввел пользователь, чтобы избежать их выполнения в виде кода. Дополнительную информацию см. в разделе Кодирование вывода . Однако в целях безопасности нельзя полагаться только на проверку на стороне клиента. Это полезное улучшение пользовательского опыта для ваших пользователей, потому что оно дает им мгновенную обратную связь проверки без необходимости ждать, пока сервер туда и обратно. В то же

время проверку на стороне клиента слишком легко обойти злоумышленнику (например, отключив JavaScript в браузере, чтобы обойти проверку на основе JavaScript), поэтому ее следует сочетать с проверкой на стороне сервера.

Любая авторитетная серверная структура предоставит функциональность для проверки отправки форм. Кроме того, общепринятой передовой практикой является экранирование любых специальных символов, которые являются частью синтаксиса исполняемого файла, что делает любой введенный код более неисполняемым и обрабатывается как обычный текст.

Защита от кликджекинга При кликджекинге пользователя обманом заставляют щелкнуть элемент пользовательского интерфейса, который выполняет действие, отличное от ожидаемого пользователем. Это может быть риск, связанный со встроенным сторонним контентом (убедитесь, что вы доверяете тому, что встраивается на ваш сайт), а также может сочетаться с фишингом.

Следующие функции могут помочь защититься от кликджекинга:

Параметры HTTP X-Frame Заголовок ответа HTTP можно использовать, чтобы указать, следует ли разрешить браузеру отображать страницу в формате , , или . Сайты могут использовать это, чтобы избежать атак кликджекинга , гарантируя, что их контент не будет встроен в другие сайты.X-Frame-Options <frame><iframe><embed><object>

CSP: фреймы-предки Директива HTTP Content-Security-Policy(CSP) frame-ancestorsуказывает действительных родителей, которые могут встраивать страницу с помощью <frame>, <iframe>, <object>или <embed>.