Регламенты и методы разработки безопасных веб-приложений

Защита данных — Общий регламент по защите данных (GDPR)

Общий регламент по защите данных (GDPR), принятый 14 апреля 2016 г. и опубликованный в Официальном вестнике ЕС 4 мая 2016 г., применяется во всех странах-членах ЕС с 25 мая 2018 г. Он отменяет и заменяет Директиву 95/46/ЕС и имплементирующее законодательство ее стран-членов.

В основе правил Общего регламента по защите данных (GDPR) лежит концепция подотчетности — это означает, что организации в пределах ЕС или вне ЕС, которые ориентированы на резидентов ЕС, должны, в соответствии с GDPR, повысить прозрачность и добросовестность при работе с персональной информацией частного лица. Персональная информация здесь означает любую информацию в отношении идентифицируемого частного лица (указанного в регламенте как субъект данных).

ПЕРСОНАЛ

В соответствии с правилами Общего регламента по защите данных (GDPR), все сотрудники обязаны придерживаться положений о конфиденциальности данных.

СБОР ДАННЫХ

Персональные данные будут собираться только при условии их добровольного предоставления, например, посредством заполнения форм или отправки писем по электронной почте в рамках заказа продуктов или услуг, или посредством обращений или запросов информационных материалов. База данных и ее содержимое остается в компании и у поставщика услуг. Тем не менее, персональные данные ни в коем случае не будут перенаправлены каким-либо третьим лицам или любой стороной, уполномоченной, если только: пользователь не дал согласие на это; это находится в рамках выполнения контрактных обязательств; они предназначены для информирования клиентов, бизнес-партнеров и/или поставщиков услуг, или если они нужны для выполнения юридических обязательств в связи со статьей 6, параграф 1, пункт «c» Общего регламента по защите данных (GDPR).

От представителей компаний клиентов и партнеров можно требовать всю или часть следующей персональной информации, в зависимости от конкретной страны-члена зоны возмещения, и необходимую документацию: базовая персональная информация, т. е. фамилия, контактные данные, копия паспорта и адрес (в случае компании с одним владельцем).

Веб-портал можно посещать без предоставления каких-либо персональных данных. Однако, при открытии веб-сайта некоторые данные сохраняются для целей резервного копирования — такие как название вашего интернет-провайдера, веб-сайт, с которого вы перешли на наш веб-сайт и ваш IP-адрес. Эти данные, возможно, могут помочь в идентификации, тем не менее, не следует использовать их в отношении отдельных частных лиц. Эти данные могут анализироваться в целях статистики, но каждый отдельный пользователь всегда остается анонимным.

ИСПОЛЬЗОВАНИЕ КУКИ-ФАЙЛОВ

Дополнительно к вышеуказанному, временные куки-файлы сохраняются на компьютере при использовании веб-сайта. Куки-файлы являются небольшими текстовыми файлами, которые сохраняются на жестком диске браузером, который использует клиент, и посредством которых осуществляется отправка информации стороной, которая

устанавливает куки-файлы. Эти куки-файлы предназначены для большего удобства использования и большей эффективности веб-сайта, и они не могут выполнять программы или загружать вирусы на ваш компьютер.

Некоторые куки-файлы необходимы для эффективного использования сайта, но если вы не хотите, чтобы куки-файлы сохранялись на ваш компьютер, вы можете задать в настройках вашего браузера отклонение всех куки-файлов до осуществления доступа к веб-сайту. Большинство браузеров предоставляет вам возможность выбора приема или отклонения куки-файлов. Однако, обратите внимание на то, что некоторые функции в результате этого могут быть недоступны. Используя веб-сайт без деактивации приема куки-файлов, вы соглашаетесь на временную установку куки-файлов на ваш компьютер. Мы предоставляем вам возможность сделать выбор в отношении приема куки-файлов в запросе согласия на использование куки-файлов на веб-сайте.

Используемые куки-файлы

Сейчас используются следующие категории куки-файлов:

1. Необходимые куки-файлы

Необходимые куки-файлы помогают сделать веб-сайт удобным посредством предоставления базовых функций, таких как навигация по страницам и доступ к защищенным зонам веб-сайта. Веб-сайт не может надлежащим образом функционировать без этих куки-файлов.

2. Куки-файлы предпочтений

Куки-файлы предпочтений позволяют веб-сайту запоминать информацию, меняющую поведение или внешнее представление веб-сайта, — такую как ваш предпочитаемый язык или регион, в котором вы находитесь.

3. Куки-файлы веб-аналитики

На веб-сайте используются различные инструменты для анализа поведения при просмотре на страницах. Главным из них является Google Аналитика, сервис веб-аналитики, предоставляемый Google Inc. («Google»). Использование осуществляется на основании статьи 6, параграф 1, пункт 1 (f) Общего регламента по защите данных (GDPR). Сгенерированная информация в отношении использования вами веб-сайта может включать:

- тип/версию браузера;
- используемую операционную систему;
- URL-адрес ссылающегося домена (предварительно посещенный веб-сайт);
- имя хоста компьютера, с которого осуществляется доступ (IP-адрес), и
- время запроса к серверу.

Эта информация в основном передается на сервер Google в США и хранится там же.

ЦЕЛЬ И ПРАВОВОЕ ОСНОВАНИЕ ДЛЯ ОБРАБОТКИ

Персональные данные будут использоваться только для предоставления клиентам нужных продуктов и/или услуг или для других целей, для которых клиенты предоставили свое согласие, при условии, что это не противоречит каким-либо существующим правовым нормам. Это осуществляется в соответствии со статьей 6, параграф 1, пункт «b» Общего регламента по защите данных (GDPR).

ПРАВА СУБЪЕКТОВ ДАННЫХ

В случае, если вы обрабатываете персональные данные, полученные от клиентов, вы являетесь затронутым лицом в значении Общего регламента по защите данных (GDPR) и имеете следующие права в отношении вас, как ответственной стороны в значении GDPR:

ПРАВО НА ИНФОРМИРОВАНИЕ

Клиент имеет право подать запрос на ознакомление с использованием ваших персональных данных. Он может включать требование о получении информации о конкретных реципиентах или категориях реципиентов, которым его персональные данные были или будут раскрыты, или требование об уточнении в отношении цели или длительности обработки.

ПРАВО НА ВНЕСЕНИЕ ПОПРАВОК

Клиент имеет право на внесение поправок и/или предоставление полных данных, если обрабатываемые персональные данные в отношении него неточные или неполные. Для этого реализуйте связь с клиентом.

ПРАВО НА ОГРАНИЧЕНИЕ ОБРАБОТКИ

Клиент может подать запрос на ограничение обработки своих персональных данных, если:

• он не согласен с правильностью персональных данных

• обработка незаконна;

• оператору (т. е. вашей компании) больше не нужны персональные данные для целей обработки;

• клиент возразил против обработки согласно статье 21, параграф 1 Общего регламента по защите данных (GDPR) в ожидании верификации, имеют ли приоритет законные основания оператора над вашими аргументами.

Если обработка персональных данных, касающихся пользователя, была ограничена, такие данные — кроме сохранения — могут быть обработаны только с его согласия или для цели заявления, выполнения или защиты прав, или защиты прав другого физического или юридического лица, или по причинам важного общественного интереса союза или страны-члена.

Если ограничение обработки было выполнено в соответствии с вышеуказанными условиями, вам необходимо уведомить клиента.

ПРАВО НА УДАЛЕНИЕ (ПРАВО НА ЗАБВЕНИЕ)

У клиента есть право на подачу запроса на удаление персональных данных при условии наступления одной из следующих причин:

- персональные данные более не нужны в связи с целями, для которых они были собраны или обработаны другим способом;
- клиент отозвал ваше согласие, на основании которого осуществляется обработка в соответствии со статьей 6, параграф 1, пункт «a» или статьей 9, параграф 2 (a) Общего регламента по защите данных (GDPR) и если нет другой законной причины для обработки;
- персональные данные были обработаны незаконным способом;
- персональные данные необходимо удалить в соответствии с законными обязательствами в законодательстве союза или страны-члена

ПРАВО НА ПЕРЕНОСИМОСТЬ ДАННЫХ

У клиента есть право на подачу запроса о передаче нами персональных данных в отношении него непосредственно другому ответственному лицу без каких-либо препятствий с нашей стороны, насколько это технически возможно и при условии, что:

- обработка основывается на согласии согласно статье 6, параграф 1 пункт «a» Общего регламента по защите данных (GDPR) или статьи 9, параграф 2 пункт «a» GDPR или на основании контракта согласно статье 6, параграф 1 (b) GDPR и
- обработка осуществляется с использованием автоматизированных методов.

Свободы и права других лиц не должны быть затронуты вашим запросом, и право на переносимость данных не применяется к обработке персональных данных, необходимых для выполнения задачи в общественных интересах или при осуществлении официального полномочия, предоставленного ответственной стороне.

ПРАВО ОБЖАЛОВАНИЯ В НАДЗОРНОМ ОРГАНЕ

Не отказываясь от любых других административных или юридических средств защиты, у клиента есть право обжалования в надзорном органе, в частности, в стране-члене, в которой вы являетесь резидентом или работаете, или подозревается нарушение, если клиент считает, что обработка персональных данных в отношении него находится в противоречии с Общим регламентом по защите данных (GDPR).

ХРАНЕНИЕ ДАННЫХ

Не учитывая любые существующие требования законодательства, такие как текущие процессуальные действия и т. д., вы должны сохранять персональные данные до тех пор, пока они будут нужны для выполнения требуемой услуги, на которую клиент дал свое согласие. Его данные будут полностью удалены из базы данных после завершения договорных соглашений.

Безопасное программирование

Знайте свои технологии

У каждой технологии есть две стороны. Одна сторона — свойства, полезные нашему пользователю, мы используем их в своих продуктах; вторая сторона — свойства, которые может использовать взломщик, слабости технологии. Иногда технология слаба настолько, что ее использование нельзя оправдать ни при каких условиях. Другие технологии использовать можно, принимая необходимые для защиты меры. Так использование SQL сервера потенциально может открыть возможность для sql injection.

Стоит знать обе стороны технологий. Как сделать полезную программу — это главное знание любого программиста, благодаря этому знанию зарабатываются деньги. Поэтому вполне понятно, что основное время стоит посвятить изучению этой стороны.

Но хороший специалист понимает и вторую сторону. Вы не обязаны знать, как взломщик будет атаковать, не обязаны уметь создавать эксплойт под ту или иную уязвимость. Но должны понимать, какие действия могут привести к появлению уязвимостей, и как можно этого избежать.

Используйте библиотеки

Не изобретайте велосипед! Конечно, очень интересно сделать что-то свое. Но в безопасности изобретательство приводит к очень неприятным последствиям, поэтому используйте проверенные средства.

Сейчас существует большое количество разнообразных библиотек. Можно найти готовые решения почти для любой задачи, причем найти можно и платные библиотеки, и библиотеки с открытым кодом.

Например, для реализации криптографических методов можно рекомендовать openssl. Библиотека хорошо известна, проверена в достаточной для большинства из нас степени, распространяется под свободной лицензией.

Веб программистам, использующим Java EE, может понравиться ESAPI. Она реализует множество методов, необходимых для создания безопасных веб приложений. В библиотеке присутствуют функции, осуществляющие фильтрацию входных данных, аутентификацию пользователя, проверку прав доступа, и многое другое. Код лицензирован под BSD лицензией, допускающей очень широкое использование.

К сожалению, ESAPI сейчас не развивается. Тем не менее, это хорошо зарекомендовавшая себя библиотека, и есть основания надеяться, что ее поддержка будет возобновлена.

Это только некоторые возможности. Более того, многие современные фреймворки также содержат необходимые функции. Обращайте внимание, кто разрабатывал эту библиотеку, насколько хорошо она проверена и протестирована.

Иногда в проекте невозможно использование чужих библиотек. В крайнем случае (совсем-совсем крайнем!) напишите свою, тщательно ее проверьте и протестируйте, попросите специалистов проверить и протестировать, еще раз сами тщательно проверьте и протестируйте. Используйте функции из этой библиотеки в своем коде, не пишите новые методы, каждый раз, когда они вам нужны.

Делайте самопроверку

Когда вы только написали свой код, еще до его компиляции, сделайте небольшой перерыв, отдохните, забудьте про код. После перерыва посмотрите на написанное с разных сторон: проверьте синтаксис, соответствие кода корпоративному стандарту; проверьте логику кода; проверьте его соответствие требованиям безопасности («знайте свои технологии»).

Самопроверка является признанной практикой создания качественного ПО. Так она включена в PSP (Personal Software Process), процесс, разработанный специально для создания качественного продукта в условиях жестких временных и финансовых ограничений.

Используйте утилиты проверки кода

Человеку свойственно делать глупые ошибки. При этом все программисты знают, что именно глупые ошибки труднее всего обнаружить: какая-нибудь непоставленная точка с запятой может привести к потере целого дня (если не больше).

Человеку сложно помнить все сразу. Для уязвимостей существует огромное количество причин; чтобы избежать проблем, надо помнить их все, надо проверять программу на их наличие. Более того, взломщики не стоят на месте: то, что считалось безопасным вчера, сегодня уже может быть уязвимо.

Частично уменьшить проблемы с глупыми ошибками и быстро меняющимся миром могут утилиты автоматизированного анализа. Эти программы просматривают код и ищут в нем признаки возможных проблем с безопасностью. Автоматизированные утилиты хороши тем, что могут быстро и почти без участия человека находить многие типы уязвимостей, давая возможность программисту обращать больше внимания на другие проблемы. Зачастую, эти утилиты могут проводить достаточно сложный анализ, который у человека занял бы очень много времени. Некоторые из этих возможностей уже встроены в современные компиляторы, надо только знать, как их включить.

Не стоит только слишком полагаться на автоматизированный анализ. К информации, выдаваемой подобными программами, надо относиться примерно так же, как мы относимся к предупреждениям компилятора. Мы все знаем, что отсутствие предупреждения еще не значит отсутствие ошибки; такие сканеры обнаруживают далеко не все уязвимости, даже хорошо известного им типа. С другой стороны, наличие предупреждения еще не является

доказательством наличия проблемы, и бездумное стремление избавиться от всех предупреждений иногда может привести только к более серьезным уязвимостям.

Автоматизация может освободить наше время для чего-то более интересного. Если часть своей работы можно поручить компьютеру, то почему бы это не сделать?

Тестируйте

Чтобы быть безопасной, программа должна демонстрировать определенное поведение. Например, при попытке ввода слишком длинной строки программа должна ее либо отвергать, либо обрезать; при попытке ввода специальных символов данные должны либо отвергаться, либо символы должны специальным образом кодироваться.

Такое поведение может быть протестировано. И мы получаем все преимущества автоматизированного тестирования программы: его можно проводить часто, практически без участия человека.

В тестировании безопасного поведения нет никаких особенностей по сравнению с обычным функциональным тестированием. Вы можете использовать все свои наработки, непрерывную интеграцию, и все остальное, что вы знаете. Конечно, вы должны уметь описать безопасное поведение (опять, «знайте свои технологии»).

Не все может быть протестировано эффективно. Тестирование не может решить всех проблем безопасности. Но если что-то возможно протестировать автоматически, почему этим не пользоваться?

Делайте code review

Ревизия кода — самый эффективный способ обнаружения ошибок. Причем он наиболее эффективен как с точки зрения количества обнаруживаемых дефектов, так и с точки зрения стоимости (времени) их обнаружения. Так Стив Макконнел в своей книге «Совершенный код» приводит ссылку на исследование компании IBM, в котором обнаружили, что один час, инвестированный в ревизию кода, сохраняет до 100 часов тестирования и устранения ошибок.

Ревизию кода можно организовать по-разному. В этой роли может выступать и парное программирование, и неформальный просмотр кода коллегами-программистами, и очень формальная инспекция с привлечением специалиста по безопасному программированию. Многое зависит от поставленных целей: чем более высокие требования предъявляются к программе, тем большее количество людей должны быть вовлечены в ревизию, тем более формально она должна проводиться.

Организация ревизии зависит и от квалификации программистов. Очевидно, если они сами в достаточной мере знакомы с принципами безопасного программирования, привлекать стороннего специалиста не имеет смысла, и неформальной процедуры вполне может быть достаточно. С другой стороны, привлечение стороннего специалиста к участию в сильно формализованной инспекции кода может стать важной частью программы обучения программистов.

Проектирование безопасности

Принципы безопасных архитектуры и дизайна

Принципы:

- простота механизмов (economy of mechanism);
- безопасность по умолчанию (fail-safe defaults);
- полное проникновение защиты (complete mediation);
- открытый дизайн (open design);
- разделение полномочий (separation of privilege);
- минимум привилегий (least privilege);

- минимизация разделения ресурсов (least common mechanism);
- психологическая приемлемость (psychological acceptability).

Эти принципы были сформулированы почти 40 лет назад в статье “The Protection of Information in Computer Systems”. Сейчас к ним любят добавлять еще несколько правил (пример), но факт тот, что эти принципы известны уже давно, и они остаются верными до настоящего времени. Есть и примеры удачных решений, построенных в соответствии с этими принципами.

Но одних только принципов не достаточно. Строить безопасную систему, зная только общие принципы безопасности, можно. Но это будет напоминать решение задач по геометрии, с опорой только на ее аксиомы. Для того чтобы проектирование было эффективным, очень полезно знать типовые подходы, готовые решения.

Шаблоны безопасных архитектуры и дизайна

В нашей работе постоянно возникают повторяющиеся задачи. Каждая система, каждая программа в чем-то похожа на многие другие. И для стандартных задач существуют стандартные решения — шаблоны.

Шаблоны хороши своей проверенностью. Каждый шаблон возник из опыта, он является результатом проб, ошибок, улучшений, устранения найденных проблем. Применяя шаблон, мы можем достаточно точно предсказать, к каким последствиям это приведет, какая от него будет польза, с какими проблемами придется бороться.

В программировании хорошо известны design patterns. Это шаблонные решения задач, возникающих при проектировании программ. Их использование существенно упрощает жизнь: надо привести стоящую перед нами задачу к стандартной, и применить уже имеющееся, заведомо работающее решение.

Шаблонами проектирование дело не ограничивается: шаблоны существуют на всех уровнях. Есть шаблоны архитектуры, есть шаблоны написания кода, есть шаблоны пользовательского интерфейса, есть шаблоны поведения программы. Практически на каждую возникающую задачу есть свои шаблонные решения.

Нас, конечно, интересуют шаблоны безопасности приложений. Здесь тоже можно говорить о разных уровнях: существуют шаблоны безопасного поведения, и шаблоны структуры безопасной программы.

Безопасность программы часто связывается именно с ее безопасным поведением. Оно включает в себя, например аутентификацию пользователя, проверку прав его доступа, фильтрацию входных данных. Поведение — это то, что легко можно показать, протестировать; именно поэтому маркетологи обычно «продают» исключительно функции безопасности.

Но от структуры программы безопасность зависит в не меньшей степени, чем от ее поведения. От структуры программы зависит вероятность сделать в ней ошибку; от структуры программы зависит, станет ли эта ошибка уязвимостью; от структуры программы зависит, насколько серьезна будет эта уязвимость. Например, задумайтесь, какая ошибка приведет к более серьезным последствиям: в коде, который имеет доступ к важным данным, или в коде, который такого доступа не имеет?

Заключение

Простую программу сделать безопасной очень легко. Для этого не надо много думать, планировать: на безопасность надо только немного обратить внимание, применить практики безопасного программирования.

Задумываясь о дизайне средней по сложности программы, вы сэкономите себе много времени, денег и, возможно, сохраните свою репутацию.

Качественная архитектура и дизайн сложной программы — единственный способ сделать ее безопасной.

Вспомните об этом, когда следующий раз будете планировать новую программу или рефакторинг уже существующей.