УДК 004.056
DOI: 10.33764/2618-981X-2019-6-2-13-19

МЕРЫ И ТРЕБОВАНИЯ К ЗАЩИЩЕННЫМ ВЕБ-ПРИЛОЖЕНИЯМ

Марк Александрович Иконников
Сибирский государственный университет геосистем и технологий, 630108, Россия, г. Новосибирск, ул. Плахотного, 10, магистрант, тел. (999)451-59-67, e-mail: mark-ikon@outlook.com

Игорь Николаевич Карманов
Сибирский государственный университет геосистем и технологий, 630108, Россия, г. Новосибирск, ул. Плахотного, 10, кандидат технических наук, доцент, зав. кафедрой информационной безопасности, тел. (903)937-24-90, e-mail: i.n.karmanov@ssga.ru

В статье обсуждаются требования, предъявляемые к защищенным веб-приложениям, и меры для обеспечения доступности, конфиденциальности и целостности обрабатываемых и хранимых данных. Анализируются наиболее опасные угрозы для веб-приложений по версии OWASP. Выбраны наиболее оптимальные и необходимые меры и требования для обеспечения безопасности веб-приложений.

Ключевые слова: веб-приложения, безопасность, клиент-серверные приложения, OWASP, кибератаки.

MEASURES AND REQUIREMENTS TO PROTECTED WEB APPLICATIONS

Mark A. Ikonnikov
Siberian State University of Geosystems and Technologies, 10, Plakhotnogo St., Novosibirsk, 630108, Russia, Graduate, phone: (999)451-59-67, e-mail: mark-ikon@outlook.com

Igor N. Karmanov
Siberian State University of Geosystems and Technologies, 10, Plakhotnogo St., Novosibirsk, 630108, Russia, Ph. D., Associate Professor, Head of Department of Information Security, phone: (903)937-27-90, e-mail: i.n.karmanov@ssga.ru

The article discusses requirements for protected web-based applications and measures to ensure the availability, confidentiality and integrity of data being processed and stored. The most dangerous threats of web applications according to OWASP are analyzed. The most optimal and necessary measures and requirements for ensuring security of web applications were selected.

Key words: web applications, security, client-server applications, OWASP, cyber-attacks.

Введение

Развитие глобальной сети Интернет и увеличение доступности ресурсов в ней привело к расширению разнообразности задач, решаемых с использованием веб-технологий. В связи с этим получил распространение особый вид приложений – веб-приложения, от которых напрямую зависит функционирование бизнес-процессов многих организаций. Веб-приложение – это клиент-серверное приложение, где в качестве клиента выступает браузер, который ото-

бражает пользовательский интерфейс, формирует запросы к серверу и обрабатывает ответы от него. А серверная часть представляет собой веб-сервер, обрабатывающий запросы клиентов. Взаимодействие между клиентом и сервером, как правило, осуществляется посредством протокола HTTP [1].

Основными особенностями, повлиявшими на распространение веб-приложений, являются:

1) доступность – данный вид приложений не привязан к определенному терминалу или локальной сети, доступ может осуществляться из любой точки Земного шара, где присутствует соединение с Интернет;

2) кроссплатформенность – клиенту достаточно иметь браузер, соответствующий стандартам, а операционная система и тип устройства не имеют никакого значения. Не требуется установка и настройка отдельного приложения на клиентском устройстве;

3) автоматическое обновление – клиент всегда работает с самой актуальной версией приложения, так как все обновления происходят на стороне сервера.

Но в то же время в процессе эволюции веб-приложений разработчики сталкивались с рядом проблем, среди которых выделяется проблема безопасности веб-приложений. Проблемы безопасности вытекают из основных особенностей веб-приложений, например, из-за невозможности изолировать веб-приложения от попыток несанкционированного доступа извне ввиду их доступности.

Целью данной работы является выбор оптимальных мер и требований для обеспечения информационной безопасности веб-приложений.

Методы обеспечения безопасности веб-приложений

Актуальность проблем безопасности веб-приложений подкрепляется тем, что в них используется конфиденциальная информация, например:

– персональные данные;

– сведения, составляющие коммерческую, банковскую или государственную тайну;

– другие сведения и процессы, зависящие от рода деятельности компании [2].

В связи с этим среди разработчиков появились рекомендации к обеспечению безопасности веб-приложений, которые вылились в проект под названием: Open Web Application Security Project (OWASP).

OWASP – это открытый проект обеспечения безопасности веб-приложений, который включает в себя корпорации, образовательные организации и индивидуальных разработчиков, которые совместными усилиями формируют статьи, рекомендации и учебные пособия, находящиеся в свободном доступе и рекомендуемые при разработке веб-приложений. Также проект включает в себя ряд тренировочных задач и инструменты для анализа безопасности веб-приложений.

OWASP на протяжении всего своего существования занимается изучением наиболее опасных уязвимостей. Рассмотрим наиболее серьезные угрозы безопасности веб-приложений, согласно исследованиям OWASP Top 10 – 2017 [3].

1. Инъекции (Injection) – внедрение в запросы к базе данных кода, дополняющего данный запрос и дающего злоумышленнику неавторизованный доступ к базе данных.

2. Уязвимости аутентификации (Broken Authentification) – распространенная уязвимость, связанная с недостаточно проработанной системой валидации пользователей в приложении, приводит к получению неавторизованного доступа.

3. Незащищенность важных данных (Sensitive Data Exposure) – многие приложения не используют механизмы для защиты передаваемых данных, такие как, например, HTTPS.

4. Внедрение внешних сущностей в XML (XML External Entities) – вид инъекции, основанный на внедрении в XML-запрос к серверу атрибутов и сущностей, позволяющих получить неавторизованный доступ к данным.

5. Небезопасный контроль доступа (Broken Access Control) – уязвимости в методах авторизации, позволяющие злоумышленнику получить повышенные привилегии.

6. Небезопасная конфигурация (Security Misconfiguration) – веб-приложение – это сложная система, состоящая из многих компонентов, таких как веб-сервер, СУБД и др. Неверная конфигурация одного из компонентов может привести к серьезным проблемам с безопасностью всего приложения.

7. XSS (Cross-Site Scripting) – внедрение (инъекция) вредоносного кода в HTTP-ответ, получаемый клиентом и выполняющийся на стороне клиента.

8. Небезопасная десериализация (Insecure Deserialization) – десериализация преобразует последовательность бит в структурированные данные, зачастую, на данном этапе не уделяется достаточно внимания безопасности, например, отсутствует валидация типов данных, что создает возможность их подмены.

9. Использование компонентов с известными уязвимостями (Using Components with Known Vulnerabilities) – зачастую, при разработке веб-приложений используются библиотеки, фреймворки и компоненты сторонних разработчиков, которые могут содержать различные недостатки (уязвимости), в связи с этим важно использовать самые актуальные версии, в которых исправляются известные уязвимости.

10. Недостаточное журналирование и мониторинг (Insufficient Logging & Monitoring) – для своевременного обнаружения несанкционированного доступа, утечки информации и т. д. необходимо использовать средства автоматизированного мониторинга трафика, а также журналирования, которые помогут понять сущность атаки, в кратчайшие сроки устранить уязвимости, а также вернуть работоспособность и исходное состояние веб-приложений.

Данный список уточняется OWASP в соответствии с развитием модели поведения нарушителя и актуальности той или иной проблемы безопасности веб-приложений. Но, к сожалению, некоторые уязвимости (например, SQL-инъекции) находятся на вершине списка долгое время, хотя уже давно разрабо-

таны методы устранения данной уязвимости [4]. Это связано с тем, что не всегда разработчики уделяют достаточно внимания безопасности веб-приложений.

В соответствии со списком угроз OWASP, первый шаг к обеспечению защищенности – правильная настройка всех компонентов системы, обновление всего используемого программного обеспечения до актуальных версий, отключение всех неиспользуемых служб, смена всех паролей по умолчанию, отключение неиспользуемых учетных записей, в том числе системных. Важно также обеспечить шифрование данных всех сетевых соединений внутри системы. Если веб-приложение позволяет настроить разграничение прав доступа пользователей к данным и настройкам системы, необходимо сделать это, ограничив права минимально необходимыми. Зачастую, уже только перечисленные базовые меры позволяют снизить риски до допустимого уровня. Если базовых мер окажется недостаточно, необходимо обратить внимание на разнообразные специализированные средства защиты информации.

Одно из наиболее известных слабых мест – это аутентификация пользователей веб-приложения. Например, система при регистрации пользователя не проводит проверку на сложность пароля, или пароль не имеет срока действия, или при аутентификации пароль от браузера передается к серверу в URL запросе. Для устранения подобных проблем существуют решения для усиления аутентификации в веб-приложении, реализованные как в виде встраиваемых модулей, так и в виде отдельных серверов аутентификации. Встроенная в приложение аутентификация позволяет не использовать внешние средства аутентификации, однако, это не исключает возможные ошибки в самом веб-приложении, позволяющие обойти систему аутентификации. Решить эту проблему можно, например, путем внедрения процесса менеджмента уязвимостей и периодическими тестированиями на проникновение.

Если в компании используются приложения, разработанные на заказ сторонним подрядчиком, и исходный код приложений недоступен, сотрудникам информационной безопасности при приемке программного обеспечения имеет смысл использовать сканер уязвимостей, который работает с веб-приложением по принципу «черного ящика» [5, 6]. При нахождении критических уязвимостей приложение возвращается на доработку разработчику. В случае, когда веб-приложение является внутренней разработкой компании, рекомендуется выстраивать процесс безопасной разработки программного обеспечения с использованием анализаторов исходного кода, проверяя весь разрабатываемый код на отсутствие ошибок, приводящих к уязвимостям. Такой подход позволяет исправить ошибки в приложении на раннем этапе и избежать лишних затрат. С помощью большинства анализаторов кода можно проводить как статический анализ кода без его выполнения, так и динамический, проверяя уже установленное и запущенное приложение. В последнем случае требуется указание точек входа и большое количество входных данных. Сочетание перечисленных методов позволяет выявлять уязвимости до внедрения приложений в компании. В то же время реализация в компании процессов верификации приложений может потребовать значительных материальных и временных затрат.

Иногда просто нет возможности оперативно вносить изменения в уже работающее приложение. Если ценность данных не позволяет закрыть глаза на возможные уязвимости, имеет смысл рассмотреть возможность использования специализированных средств защиты веб-приложений – Web Application Firewall (WAF). Существуют как коммерческие, так и свободно распространяемые системы. Принцип работы WAF состоит в том, что HTTP-трафик от пользователей до веб-приложения проходит сначала через WAF либо, в зависимости от задач и возможностей, на WAF направляется копия трафика. Далее трафик подвергается декодированию и проверке на наличие атак. Если WAF установлен «в разрыв» (прокси, мост), атаки могут быть заблокированы. В пассивном режиме работы (копия трафика) возможны только мониторинг и оповещение об атаках. Для обнаружения атак могут использоваться такие методы, как [7, 8]:

– сигнатурный анализ;
– репутационные списки;
– автоматическое обучение;
– поведенческий анализ;
– вручную настраиваемые правила.

Кроме этого, WAF может иметь модули для динамического анализа уязвимостей защищаемых приложений, виртуального патчинга найденных уязвимостей, управления аутентификацией пользователей, взаимодействия с другими системами защиты. Все это позволяет снизить количество актуальных для веб-приложения угроз.

Опасность для информации в веб-приложении представляют и внутренние нарушители – сотрудники, которые имеют доступ к данным для выполнения служебных обязанностей, администраторы с прямым доступом к серверу баз данных (в обход веб-приложения или локально). В этом случае для обеспечения безопасности веб-приложений возможно использовать решения, реализующие разный подход к мониторингу и контролю обращений к базам данных. Системы защиты информации в базах данных можно разделить на три типа.

Системы первого типа используют принцип работы, схожий с WAF – перехват трафика, но идущего не от пользователя до сервера приложений, а от сервера приложений к серверам баз данных. Производится декодирование протоколов баз данных с последующим анализом, используя правила, настроенные сотрудником информационной безопасности. Возможна работа в активном (блокирующем) режиме, для этого система устанавливается «в разрыв» (прокси, мост), а также в пассивном режиме мониторинга, для этого достаточно подать копию трафика. Для контроля локальных и прямых сетевых подключений к базам данных используются агенты, устанавливаемые непосредственно на серверы баз данных. При использовании таких систем для защиты данных в веб-приложениях с трехзвенной архитектурой может возникнуть сложность с определением пользователя, сделавшего запрос к базе данных: в трафике, идущем от сервера приложений к серверу баз данных, все обращения производятся от имени служебной учетной записи. Для персонификации сотрудника предусмотрена интеграция с WAF, который анализирует трафик до сервера

приложений, либо подача копии этого трафика непосредственно на систему защиты баз данных. Также в системах рассматриваемого типа могут быть реализованы возможности, косвенно повышающие защищенность: сканирование на уязвимости баз данных, обнаружение баз данных, маскирование критичной информации, например номеров кредитных карт (при установке «в разрыв»), создание матрицы разграничения прав доступа, мониторинг изменений, позволяющий вовремя отследить несанкционированное повышение прав пользователя [9].

В случаях, когда возможность анализа копии трафика отсутствует, или необходимо применить маскирование и блокировки, но установить систему защиты «в разрыв» невозможно, используются решения, основанные на других принципах перехвата обращений пользователей к базам данных. Такие решения используют в качестве точки съема агент, устанавливаемый на защищаемый сервер приложения и взаимодействующий с драйверами, через которые веб-приложение передает запросы пользователей к базам данных. Так как агент находится на сервере веб-приложения, он обрабатывает и запросы клиентов к приложению, и запросы приложения к базам данных, персонифицируя запросы. Создавая правила обработки запросов, можно маскировать «на лету» любые поля в ответах от базы данных, блокировать нелегитимные запросы, полностью управлять бизнес-процессом работы пользователя с приложением.

Третий тип – системы защиты информации от утечек, основанные на использовании криптографии и позволяющие выборочно шифровать информацию, хранящуюся в таблицах баз данных. Доступ к информации предоставляется только авторизованным пользователям с ведением детальных протоколов их действий. Для повышения защищенности информации такие системы могут дополнительно реализовывать механизмы строгой двухфакторной аутентификации [10].

Использование перечисленных средств позволит существенно снизить риск утечки данных из веб-приложения, а при инцидентах поможет отыскать необходимую для расследования информацию. Последний рубеж защиты – правильное хранение резервных копий баз данных: если средства шифрования при работе с базой данных не используются, необходимо шифровать ее резервные копии.

Результаты

В результате можно предложить следующие меры для повышения защищенности веб-приложений. Необходимо:

– ознакомиться и следить за обновлениями документов OWASP;

– использовать методологии тестирования веб-приложений, направленные на поиск уязвимостей (например, OWASP Testing Project);

– регулярно обновлять программное обеспечение веб-сервера;

– следить за корректной настройкой сетевых устройств, служб и программного обеспечения;

– следить за обновлениями используемых в приложении фреймворков и библиотек и своевременно устранять найденные в них уязвимости;

– использовать протоколы с шифрованием (HTTPS);

– повсеместно использовать средства обнаружения атак, мониторинга активности, журналирования и системы транзакций.

Заключение

В статье проанализированы методы обеспечения соответствия веб-приложений требованиям по защите информации. В результате предложен перечень мер, которые позволят значительно повысить уровень защищенности, что приведет к снижению рисков при использовании веб-приложений.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Вора П. Шаблоны проектирования веб-приложений. – М. : Эксмо, 2011. – 870 с.
2. Пьюривал С. Основы разработки веб-приложений. – СПб. : Питер, 2015. – 272 с.
3. OWASP Top 10 – 2017 The Ten Most Critical Web Application Security Risks [Электронный ресурс]. – Режим доступа: https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf (дата обращения: 01.03.2019).
4. Как хакеры атакуют веб-приложения: боты и простые уязвимости. Блог компании Positive Technologies [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/pt/blog/327344/ (дата обращения: 01.03.2019).
5. Петухов А. Как выбрать сканер уязвимостей веб-приложений? [Электронный ресурс]. – Режим доступа: https://www.securitylab.ru/blog/personal/andrepetukhov/143697.php (дата обращения: 01.03.2019).
6. Томилов И. О., Трифанов А. В. Фаззинг. Поиск уязвимостей в программном обеспечении без наличия исходного кода // Интерэкспо ГЕО-Сибирь-2017: XIII Междунар. науч. конгр.: Магистерская науч. сессия «Первые шаги в науке»: сборник материалов в 2 т. (Новосибирск, 17–21 апреля 2017 г.). – Новосибирск: СГУГиТ, 2017. Т. 2. – С. 74–80.
7. Мельников В. Г., Гребень А. Е., Макарова Д. Г. Исследование межсетевых экранов для веб-приложений с открытым исходным кодом // Интерэкспо ГЕО-Сибирь-2018: XIV Междунар. науч. конгр.: Междунар. науч. конф. «Наука. Оборона. Безопасность-2018»: сб. материалов (Новосибирск, 23–27 апреля 2018 г.). – Новосибирск: СГУГиТ, 2018. – С. 233–236.
8. Мельников В. Г., Трифанов А. В. Методы обхода межсетевых экранов для приложений // Интерэкспо ГЕО-Сибирь-2017: XIII Междунар. науч. конгр.: Магистерская науч. сессия «Первые шаги в науке»: сборник материалов в 2 т. (Новосибирск, 17–21 апреля 2017 г.). – Новосибирск: СГУГиТ, 2017. Т. 2. – С. 113–117.
9. Суханов А. Защита данных веб-приложений от внутренних угроз [Электронный ресурс]. – Режим доступа: https://www.anti-malware.ru/practice/solutions/web-applications-internal-threats-security (дата обращения: 01.03.2019).
10. Двухфакторная аутентификация: что это и зачем оно нужно? Блог Лаборатории Касперского [Электронный ресурс]. – Режим доступа: https://www.kaspersky.ru/blog/what_is_two_factor_authenticatio/4272/ (дата обращения: 01.03.2019).