Стандарт Проверки безопасности Приложений 4.0.2

Последнее обновление

Октябрь 2020 года

Оглавление

Фронтиспис

О стандарте

Стандарт проверки безопасности приложений-это список требований к безопасности приложений или тестов, которые могут использоваться архитекторами, разработчиками, тестировщиками, специалистами по безопасности, поставщиками инструментов и потребителями для определения, создания, тестирования и проверки безопасных приложений.

Авторское право и лицензия

Версия 4.0.2, октябрь 2020 г.

Руководители проектов

Эндрю ван дер Сток	Дэниел Катберт	Джим Манико
Джош Гроссман	Марк Бернетт	

Основные Вкладчики

Абхай Бхаргав	Benedikt Bauer	Элар Ланг
Усама Эльнаггар	Рон Перрис	Тонимир Кисасонди

Другие авторы и рецензенты

Аарон Гусман	Энтони Уимс	Barbara Schachner	Кристофер Лессл	Clément Notin
Дэн Корнелл	Daniël Geerts	Дэвид Кларк	Дэвид Йоханссон	Дэвид Куизенберри
Эрленд Офтедал	Fatih Ersinadim	Филип ван Ленен	Джефф Басквилл	Гленн тен Кейт
Грант Онджерс	привет7	Джейкоб Салласси	Джеймс Сулински	Джейсон Аксли
Джейсон Морроу	Хавьер Домингес	Джет Андерсон	Джим Ньюман	Jonathan Schnittger
Джозеф Керби	Kelby Ludwig	Ларс Хаулин	Льюис Ардерн	lyz-код
Марк Обри	Marco Schnüriger	Philippe De Ryck	Ральф Андалис	Рави Балла
Рик Митчелл	Riotaro Окада	Робин Вуд	Роган Доуз	Райан Голтри

Саджад Поурали	Serg Belkommen	Siim Puustusmaa	Ståle Pettersen	Стюарт Гюнтер
Тал Аргони	Tomasz Wrobel	Vincent De Schutter		

Если в приведенном выше списке кредитов 4.0.2 отсутствует кредит, пожалуйста, зарегистрируйте билет на GitHub, чтобы он был распознан в будущих обновлениях 4.x.

Стандарт проверки безопасности приложений построен на плечах участников от ASV 1.0 в 2008 году до 3.0 в 2016 году. Большая часть элементов структуры и проверки, которые все еще находятся в резюме на сегодняшний день, были первоначально написаны Майком Боберски, Джеффом Уильямсом и Дейвом Уичерсом, но есть еще много авторов. Спасибо всем, кто принимал участие в этом ранее. Для получения полного списка всех тех, кто внес свой вклад в более ранние версии, пожалуйста, ознакомьтесь с каждой предыдущей версией.

Предисловие

Добро пожаловать в Стандарт проверки безопасности приложений (ASV) версии 4.0. ASV-это усилия сообщества, направленные на создание системы требований и средств контроля безопасности, которые сосредоточены на определении функциональных и нефункциональных средств контроля безопасности, необходимых при проектировании, разработке и тестировании современных веб-приложений и веб-сервисов.

Версия 4.0.2-это второе незначительное исправление к версии 4.0, предназначенное для исправления орфографических ошибок и уточнения требований без внесения критических изменений, таких как существенные изменения требований или добавление/удаление требований.

ASV v4.0-это кульминация усилий сообщества и отзывов отрасли за последнее десятилетие. Мы попытались упростить внедрение ACB для различных вариантов использования на протяжении любого жизненного цикла разработки безопасного программного обеспечения.

Мы ожидаем, что, скорее всего, никогда не будет 100% - ного соглашения о содержании любого стандарта веб-приложений, включая ACB. Анализ рисков всегда в какой-то степени субъективен, что создает проблему при попытке обобщения в едином для всех стандарте. Тем не менее, мы надеемся, что последние обновления, сделанные в этой версии, являются шагом в правильном направлении и расширяют концепции, представленные в этом важном отраслевом стандарте.

Что нового в 4.0

Наиболее значительным изменением в этой версии является принятие Руководящих принципов цифровой идентификации NIST 800-63-3, вводящих современные, основанные на доказательствах и расширенные средства контроля аутентификации. Хотя мы ожидаем некоторого отката в направлении согласования с расширенным стандартом аутентификации, мы считаем, что необходимо согласовать стандарты, главным образом, когда другой хорошо зарекомендовавший себя стандарт безопасности приложений основан на фактических данных.

Стандарты информационной безопасности должны стремиться свести к минимуму количество уникальных требований, чтобы соответствующим организациям не приходилось принимать решения о конкурирующих или несовместимых средствах контроля. Топ-10 OWASP 2017 года, а теперь Стандарт проверки безопасности приложений OWASP теперь согласован с NIST 800-63 для аутентификации и управления сеансами. Мы призываем другие органы, устанавливающие стандарты, сотрудничать с нами, NIST и другими, чтобы прийти к общепринятому набору средств контроля безопасности приложений для обеспечения максимальной безопасности и минимизации затрат на соблюдение.

Номер ASV 4.0 был полностью перенумерован от начала до конца. Новая схема нумерации позволила нам устранить пробелы в давно исчезнувших главах и сегментировать более длинные главы, чтобы свести к минимуму количество элементов управления, которые разработчик или команда должны соблюдать. Например, если приложение не использует JWT, весь раздел по JWT в управлении сеансами неприменим.

Новинка в версии 4.0-это комплексное сопоставление с Общим перечислением слабостей (CWE), одним из наиболее часто используемых запросов на функции, которые у нас были за последнее десятилетие. CWE-сопоставление позволяет производителям инструментов и тем, кто использует программное обеспечение для управления уязвимостями, сопоставлять результаты других инструментов и предыдущих версий ASV с 4.0 и более поздними версиями. Чтобы освободить место для записи CWE, нам пришлось удалить столбец "С тех пор", который, поскольку мы полностью перенумеровали, имеет меньше смысла, чем в предыдущих версиях ASV. Не каждый элемент в ASV имеет соответствующий CWE, и, поскольку CWE имеет большое дублирование, мы попытались использовать наиболее часто используемые, а не обязательно самые близкие совпадения. Средства контроля за проверкой не всегда сопоставимы с эквивалентными слабостями. Мы приветствуем продолжающуюся дискуссию с сообществом CWE и областью информационной безопасности в целом по устранению этого разрыва.

Мы работали над тем, чтобы всесторонне соответствовать и превосходить требования для решения задач OWASP Top 10 2017 года и проактивных средств управления OWASP 2018. Поскольку OWASP Top 10 2017-это абсолютный минимум, позволяющий избежать небрежности, мы намеренно включили все, кроме конкретных требований к Топ-10 для ведения журнала, элементы управления 1-го уровня, что облегчает пользователям OWASP Top 10 переход на реальный стандарт безопасности.

Мы намерены обеспечить, чтобы ASV 4.0 уровня 1 представлял собой комплексный набор разделов 6.5 PCI DSS 3.2.1 для проектирования приложений, кодирования, тестирования, проверки защищенности кода и тестов на проникновение. Это потребовало покрытия переполнения буфера и небезопасных операций с памятью в версии 5 и флагов небезопасной компиляции, связанных с памятью, в версии 14, в дополнение к существующим ведущим в отрасли требованиям к проверке приложений и веб-служб.

Мы завершили переход ACB от монолитных серверных элементов управления только к обеспечению контроля безопасности для всех современных приложений и API. Во времена функционального программирования, безсерверного API, мобильных устройств, облаков, контейнеров, CI/CD и DevSecOps, федерации и многого другого мы не можем продолжать игнорировать современную архитектуру приложений. Современные приложения разработаны совсем не так, как те, которые были созданы, когда оригинальная ASVS была выпущена в 2009 году. Разработчики ASV всегда должны смотреть далеко в будущее, чтобы мы давали разумные советы нашей основной аудитории - разработчикам. Мы уточнили или удалили любое требование, которое предполагает, что приложения выполняются в системах, принадлежащих одной организации.

В связи с размером ASV 4.0, а также нашим желанием стать базовым ASV для всех других проектов ASV, мы удалили мобильный раздел в пользу Стандарта проверки безопасности мобильных приложений (MASV). Приложение "Интернет вещей" появится в будущем IoT ASV по уходу за проектом OWASP "Интернет вещей". Мы включили ранний предварительный просмотр ASV IoT в Приложение C. Мы благодарим как мобильную команду OWASP, так и проектную группу OWASP IoT за поддержку ASV и надеемся на сотрудничество с ними в будущем для обеспечения дополнительных стандартов.

Наконец, мы избавились от обмана и удалили менее эффективные средства контроля. Со временем ASV стали представлять собой полный набор элементов управления, но не все элементы управления одинаковы при создании безопасного программного обеспечения. Эти усилия по устранению предметов с низким уровнем воздействия могли бы пойти дальше. В будущем издании ASV Система оценки общей слабости (CWSS) поможет дополнительно расставить приоритеты в отношении тех элементов управления, которые действительно важны, и тех, которые должны быть отменены.

Начиная с версии 4.0, ASV будут ориентированы исключительно на то, чтобы быть ведущим стандартом веб-приложений и услуг, охватывающим традиционную и современную архитектуру приложений, а также гибкие методы обеспечения безопасности и культуру DevSecOps.

Использование ASV

ASVS преследует две основные цели:

- чтобы помочь организациям разрабатывать и поддерживать безопасные приложения.
- чтобы поставщики услуг безопасности, поставщики средств безопасности и потребители могли согласовать свои требования и предложения.

Уровни Проверки Безопасности Приложений

Стандарт проверки безопасности приложений определяет три уровня проверки безопасности, с каждым уровнем углубляющийся.

- ASV уровня 1 предназначен для низких уровней надежности и полностью поддается тестированию на проникновение
- ASV уровня 2 предназначен для приложений, содержащих конфиденциальные данные, которые требуют защиты и рекомендуемый уровень для большинства приложений
- ASV уровня 3 предназначен для наиболее важных приложений - приложений, которые выполняют транзакции с высокой стоимостью, содержат конфиденциальные медицинские данные или любые приложения, требующие самого высокого уровня доверия.

Каждый уровень ASVS содержит список требований безопасности. Каждое из этих требований также может быть сопоставлено с функциями и возможностями безопасности, которые должны быть встроены разработчиками в программное обеспечение.

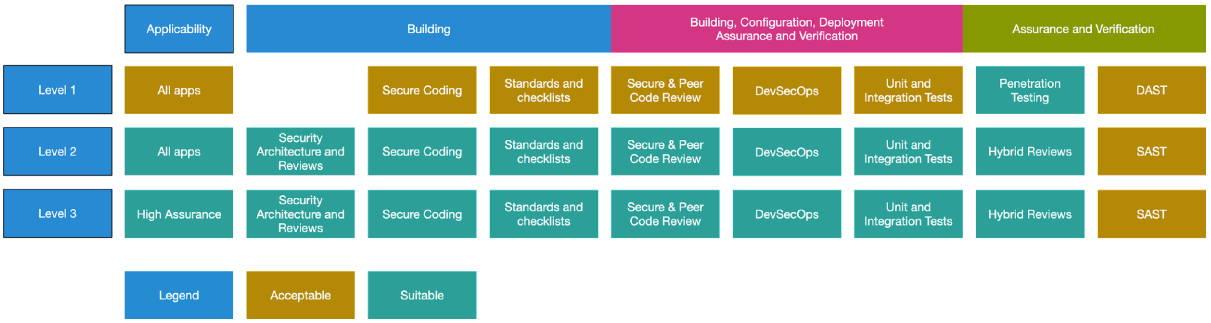

Рисунок 1 - Уровни стандарта 4.0 проверки безопасности приложений OWASP

Уровень 1 - единственный уровень, который полностью поддается проверке на проникновение с помощью человека. Все остальные требуют доступа к документации, исходному коду, конфигурации и людям, участвующим в процессе разработки. Однако, даже если L1 допускает тестирование "черного ящика" (без документации и источника), это не является эффективным мероприятием по обеспечению качества, и его следует активно поощрять. У злоумышленников много времени, большинство тестов на проникновение завершается в течение пары недель. Защитникам необходимо встраивать средства контроля безопасности, защищать, находить и устранять все слабые места, а также обнаруживать злоумышленников и реагировать на них в разумные сроки. У злоумышленников, по сути, бесконечное время, и для успеха им требуется только одна пористая защита, одна слабость или отсутствие обнаружения. Тестирование черного ящика, часто выполняемое в конце разработки, быстро или вообще не может справиться с этой асимметрией.

За последние более чем 30 лет тестирование черного ящика снова и снова доказывало, что в нем отсутствуют критические проблемы безопасности, которые непосредственно привели к еще более массовым нарушениям. Мы настоятельно рекомендуем использовать широкий спектр средств обеспечения и проверки безопасности, включая замену тестов на проникновение на основе исходного кода (гибридными) тестами на проникновение на уровне 1, с полным доступом к разработчикам и документации на протяжении всего процесса разработки. Финансовые регулирующие органы не допускают проведения внешних финансовых проверок без доступа к

бухгалтерским книгам, образцам транзакций или лицам, осуществляющим контроль. Промышленность и правительства должны требовать одинакового стандарта прозрачности в области разработки программного обеспечения.

Мы настоятельно рекомендуем использовать инструменты безопасности в рамках самого процесса разработки. Инструменты DAST и SAST могут непрерывно использоваться конвейером сборки для поиска легко обнаруживаемых проблем безопасности, которых никогда не должно быть.

Автоматизированные инструменты и онлайн-сканирование не могут выполнить более половины ACB без человеческой помощи. Если требуется комплексная автоматизация тестирования для каждой сборки, то используется комбинация пользовательских модульных и интеграционных тестов, а также онлайн-сканирование, инициированное сборкой. Недостатки бизнес-логики и тестирование контроля доступа возможны только при использовании человеческой помощи. Их следует превратить в модульные и интеграционные тесты.

Как использовать этот стандарт

Один из лучших способов использовать Стандарт проверки безопасности приложений-использовать его в качестве основы для создания контрольного списка безопасного кодирования, специфичного для вашего приложения, платформы или организации. Адаптация ASV к вашим вариантам использования позволит сосредоточить внимание на требованиях безопасности, наиболее важных для ваших проектов и сред.

Уровень 1 - Первые шаги, автоматизированный или полный просмотр портфолио

Приложение достигает уровня ASV 1, если оно адекватно защищает от уязвимостей безопасности приложений, которые легко обнаружить, и включено в Топ-10 OWASP и другие аналогичные контрольные списки.

Уровень 1-это абсолютный минимум, к которому должны стремиться все приложения. Это также полезно в качестве первого шага в многоэтапной работе или когда приложения не хранят и не обрабатывают конфиденциальные данные и, следовательно, не нуждаются в более строгих элементах управления уровня 2 или 3. Элементы управления уровня 1 могут проверяться либо автоматически с помощью инструментов, либо просто вручную без доступа к исходному коду. Мы считаем уровень 1 минимально необходимым для всех приложений.

Угрозы приложению, скорее всего, будут исходить от злоумышленников, которые используют простые и не требующие больших усилий методы для выявления легко обнаруживаемых и легко эксплуатируемых уязвимостей. Это отличается от решительного злоумышленника, который будет тратить сосредоточенную энергию специально для приложения. Если данные, обрабатываемые вашим приложением, имеют высокую ценность, вам вряд ли захочется останавливаться на проверке уровня 1.

Уровень 2 - Большинство приложений

Приложение достигает уровня ASV 2 (или стандарта), если оно адекватно защищает от большинства рисков, связанных с программным обеспечением сегодня.

Уровень 2 обеспечивает наличие, эффективность и использование элементов управления безопасностью в приложении. Уровень 2 обычно подходит для приложений, которые обрабатывают важные бизнес-транзакции, в том числе те, которые обрабатывают медицинскую информацию, реализуют критически важные для бизнеса или конфиденциальные функции или обрабатывают другие конфиденциальные активы, или в отраслях, где целостность является важным аспектом защиты их бизнеса, например, в игровой индустрии для предотвращения мошенников и взломов игр.

Угрозы для приложений уровня 2, как правило, представляют собой квалифицированных и мотивированных злоумышленников, нацеленных на конкретные цели, использующих инструменты и методы, которые хорошо отработаны и эффективны при обнаружении и использовании слабых мест в приложениях.

Уровень 3 - Высокая ценность, высокая надежность и высокая безопасность

Уровень 3 ASVS-это самый высокий уровень проверки в рамках ASVS. Этот уровень обычно зарезервирован для приложений, требующих значительных уровней проверки безопасности, таких как те, которые могут быть найдены в областях вооруженных сил, здравоохранения и безопасности, критической инфраструктуры и т.д.

Организациям может потребоваться уровень ASV 3 для приложений, выполняющих критически важные функции, где сбой может существенно повлиять на работу организации и даже на ее живучесть. Пример руководства по применению ASV уровня 3 приведен ниже. Приложение достигает уровня ASVS 3 (или продвинутого), если оно надлежащим образом защищено от продвинутых уязвимостей безопасности приложений, а также демонстрирует принципы надлежащего проектирования безопасности.

Приложение на уровне 3 ASVS требует более глубокого анализа архитектуры, кодирования и тестирования, чем все остальные уровни. Безопасное приложение модулируется значимым образом (для обеспечения устойчивости, масштабируемости и, прежде всего, уровней безопасности), и каждый модуль (разделенный сетевым подключением и/или физическим экземпляром) выполняет свои собственные обязанности по обеспечению безопасности (углубленная защита), которые необходимо надлежащим образом документировать. Обязанности включают средства контроля для обеспечения конфиденциальности (например, шифрование), целостности (например, транзакции, проверка ввода), доступности (например, корректная обработка нагрузки), аутентификации (в том числе между системами), отказа, авторизации и аудита (ведение журнала).

Применение ASV на практике

Разные угрозы имеют разную мотивацию. Некоторые отрасли обладают уникальными информационными и технологическими активами и требованиями к соблюдению нормативных требований в конкретных областях.

Организациям настоятельно рекомендуется внимательно изучить свои уникальные характеристики рисков, основанные на характере их бизнеса, и на основе этих рисков и бизнес - требований определить соответствующий уровень ASVS.

Как ссылаться на требования ASVS

Каждое требование имеет идентификатор в формате <глава>.<глава><раздел>.<требование> где каждый элемент представляет собой число, например: 1.11.3.<требование>

- Значение <chapter> соответствует главе, из которой исходит требование, например: все 1.#.# требования взяты из главы об архитектуре.<глава>
- Значение <раздел> соответствует разделу в этой главе, в котором отображается требование, например: все требования 1.11.# находятся в разделе Требования к архитектуре бизнес-логики главы Архитектура.<раздел>
- Значение <требование> определяет конкретное требование в главе и разделе, например: 1.1.1.3, которое по состоянию на версию 4.0.2 настоящего стандарта является:<требование>

Убедитесь, что все потоки бизнес-логики с высокой стоимостью, включая проверку подлинности, управление сеансами и управление доступом, потокобезопасны и устойчивы к условиям гонки времени проверки и времени использования.

Идентификаторы могут меняться в разных версиях стандарта, поэтому предпочтительно, чтобы другие документы, отчеты и инструменты использовали формат: v<версия>-<версия><глава>.<глава><раздел>.<требование>, где: "версия" - это тег версии ASVS. Например: v4.0.2-1.11.3 будет пониматься как конкретно 3-е требование в разделе "Требования к архитектуре бизнес-логики "главы" Архитектура " версии 4.0.2. (Это может быть обобщенно как v<версия> - <версия><requirement_identifier>.)<requirement_identifier>

Примечание: Буква v, предшествующая части версии, должна быть строчной.

Если идентификаторы используются без включения элемента v<версия>, то следует предположить, что они относятся к последнему стандарту проверки безопасности приложений. Очевидно, что по мере роста и изменения стандарта это становится проблематичным, поэтому авторы и разработчики должны включать элемент версии.

Списки требований ASVS доступны в формате CSV, JSON и других форматах, которые могут быть полезны для справочного или программного использования.

Оценка и сертификация

Позиция OWASP в отношении сертификатов ASV и знаков доверия

OWASP, как некоммерческая организация, не зависящая от поставщиков, в настоящее время не сертифицирует никаких поставщиков, верификаторов или программного обеспечения.

Все такие утверждения о гарантиях, знаки доверия или сертификаты официально не проверяются, не регистрируются и не сертифицируются OWASP, поэтому организации, полагающейся на такое представление, необходимо проявлять осторожность в отношении доверия, оказанного любой третьей стороне, или знака доверия, претендующего на сертификацию ASVS.

Это не должно препятствовать организациям предлагать такие услуги по обеспечению качества, если они не претендуют на официальную сертификацию OWASP.

Руководство для Сертифицирующих организаций

Приложение проверки безопасности стандарт может быть использован в качестве открытой книгой проверки заявки, в том числе открытый и беспрепятственный доступ к ключевым ресурсам, таким, как архитекторам и разработчикам, проектной документации, исходных кодов авторизованного доступа к тест-систем (включая доступ к одному или нескольким счетам в каждой роли), в частности, для L2 и L3 проверок.

Исторически сложилось так, что тестирование на проникновение и обзоры защищенного кода включали проблемы “в порядке исключения” - то есть в итоговом отчете отображаются только неудачные тесты. Сертифицирующая организация должна включать в любой отчет объем проверки (особенно если ключевой компонент выходит за рамки, например, проверка подлинности единого входа), краткое изложение результатов проверки, включая пройденные и неудачные тесты, с четкими указаниями о том, как разрешить неудачные тесты.

Некоторые требования к проверке могут быть неприменимы к тестируемому приложению. Например, если вы предоставляете своим клиентам API уровня службы без состояния без реализации клиента, многие требования в Управлении сеансами V3 не применимы напрямую. В таких случаях сертифицирующая организация все равно может заявить о полном соответствии ASVS, но должна четко указать в любом отчете причину неприменимости таких исключенных требований к проверке.

Хранение подробных рабочих документов, скриншотов или фильмов, сценариев для надежного и многократного использования проблемы, а также электронных записей тестирования, таких как перехват журналов прокси-серверов и связанных с ними заметок, таких как список очистки, считается стандартной отраслевой практикой и может быть действительно полезным в качестве доказательств результатов для самых сомнительных разработчиков. Недостаточно просто запустить инструмент и сообщить о сбоях; это (вообще) не дает достаточных доказательств того, что все проблемы на уровне сертификации были тщательно протестированы и проверены. В случае спора должно быть достаточно доказательств, подтверждающих, что каждое проверенное требование действительно было проверено.

Способ испытания

Сертифицирующие организации могут свободно выбирать соответствующие методы тестирования, но должны указывать их в отчете.

В зависимости от тестируемого приложения и требований к проверке для получения одинаковой уверенности в результатах могут использоваться различные методы тестирования. Например, проверка эффективности механизмов проверки входных данных приложения может быть проанализирована либо с помощью ручного теста на проникновение, либо с помощью анализа исходного кода.

Роль автоматизированных средств тестирования безопасности

Рекомендуется использовать автоматизированные инструменты тестирования на проникновение, чтобы обеспечить как можно больший охват.

Невозможно полностью завершить проверку ASV, используя только автоматизированные инструменты тестирования на проникновение. В то время как подавляющее большинство требований в L1 может быть выполнено с помощью автоматизированных тестов, общее большинство требований не поддается автоматизированному тестированию на проникновение.

Пожалуйста, обратите внимание, что границы между автоматическим и ручным тестированием стираются по мере развития индустрии безопасности приложений. Автоматизированные инструменты часто настраиваются вручную экспертами, а ручные тестировщики часто используют широкий спектр автоматизированных инструментов.

Роль тестирования на проникновение

В версии 4.0 мы решили сделать L1 полностью тестируемым на проникновение без доступа к исходному коду, документации или разработчикам. Два элемента ведения журнала, которые необходимы для соответствия требованиям OWASP Top 10 2017 A10, потребуют интервью, скриншотов или сбора других доказательств, как и в OWASP Top 10 2017. Однако тестирование без доступа к необходимой информации не является идеальным методом проверки безопасности, поскольку оно упускает возможность проверки источника, выявления угроз и отсутствующих элементов управления, а также выполнения гораздо более тщательного тестирования в более короткие сроки.

Там, где это возможно, при выполнении оценки L2 или L3 требуется доступ к разработчикам, документации, коду и доступу к тестовому приложению с непроизводственными данными. Тестирование на проникновение, проводимое на этих уровнях, требует такого уровня доступа, который мы называем "гибридными проверками" или "гибридными тестами на проникновение".

Другие варианты использования ASV

Помимо использования для оценки безопасности приложения, мы определили ряд других потенциальных применений ASV.

В Качестве Подробного Руководства По Архитектуре Безопасности

Одно из наиболее распространенных применений Стандарта проверки безопасности приложений-это ресурс для архитекторов безопасности. В архитектуре прикладной бизнес-безопасности Sherwood (SABSA) отсутствует большое количество информации, необходимой для завершения тщательного анализа архитектуры безопасности приложений. ASV можно использовать для заполнения этих пробелов, позволяя архитекторам безопасности выбирать более эффективные средства управления для решения распространенных проблем, таких как шаблоны защиты данных и стратегии проверки входных данных.

В качестве замены готовых контрольных списков безопасного кодирования

Многие организации могут извлечь выгоду из внедрения ASV, выбрав один из трех уровней или разветвив ASV и изменив то, что требуется для каждого уровня риска приложения, в зависимости от домена. Мы поощряем этот тип разветвления до тех пор, пока поддерживается прослеживаемость, так что, если приложение прошло требование 4.1, это означает то же самое для разветвленных копий, что и стандарт по мере его развития.

В качестве руководства для автоматизированных модульных и интеграционных тестов

ASVS разработан таким образом, чтобы его можно было легко протестировать, за исключением требований к архитектуре и вредоносному коду. Создавая модульные и интеграционные тесты, которые проверяют наличие конкретных и соответствующих случаев ошибок и злоупотреблений, приложение становится почти самоподтверждающимся при каждой сборке. Например, для набора тестов для контроллера входа в систему можно создать дополнительные тесты, включая проверку параметра имени пользователя для общих имен пользователей по умолчанию, перечисление учетных записей, перебор, LDAP и SQL-инъекции, а также XSS. Аналогично, проверка

параметра пароля должна включать общие пароли, длину пароля, ввод нулевого байта, удаление параметра, XSS и многое другое.

Для Обучения Безопасному Развитию

ASV также можно использовать для определения характеристик защищенного программного обеспечения. Многие курсы “безопасного кодирования” - это просто курсы этического взлома с легким намеком на советы по кодированию. Это может не обязательно помочь разработчикам писать более безопасный код. Вместо этого на курсах безопасной разработки можно использовать ASV с особым упором на превентивные меры контроля, содержащиеся в ASV, а не на Топ-10 негативных вещей, которые не следует делать.

В качестве драйвера гибкой безопасности приложений

ASV могут использоваться в процессе гибкой разработки в качестве основы для определения конкретных задач, которые должны быть реализованы командой для создания безопасного продукта. Один из подходов может заключаться в следующем: начиная с уровня 1, проверьте конкретное приложение или систему в соответствии с требованиями ASVS для указанного уровня, найдите, какие элементы управления отсутствуют, и добавьте конкретные заявки/задачи в список невыполненных. Это помогает в определении приоритетов конкретных задач (или уходе) и делает безопасность заметной в гибком процессе. Это также может быть использовано для определения приоритетов задач аудита и проверки в организации, где конкретное требование ASV может быть движущей силой для проверки, рефакторинга или аудита для конкретного члена команды и отображаться как "задолженность" в невыполненной работе, которую необходимо в конечном итоге выполнить.

В качестве основы для руководства Закупкой защищенного программного обеспечения

ASVS-отличная платформа для обеспечения безопасной закупки программного обеспечения или предоставления услуг по разработке на заказ. Покупатель может просто установить требование о том, что программное обеспечение, которое он хочет приобрести, должно быть разработано на уровне ASVS X, и потребовать, чтобы продавец доказал, что программное обеспечение соответствует уровню ASVS X. Это хорошо работает в сочетании с приложением Контракта на безопасное программное обеспечение OWASP

V1: Требования к архитектуре, дизайну и моделированию угроз

Цель Контроля

Архитектура безопасности почти стала утраченным искусством во многих организациях. Дни архитектора предприятия прошли в эпоху DevSecOps. Область безопасности приложений должна догнать и принять гибкие принципы безопасности, одновременно заново внедряя ведущие принципы архитектуры безопасности для специалистов-разработчиков программного обеспечения. Архитектура-это не реализация, а способ мышления о проблеме, которая потенциально имеет множество различных ответов, и ни одного "правильного" ответа. Слишком часто безопасность рассматривается как негибкая и требующая, чтобы разработчики исправляли код определенным образом, в то время как разработчики могут знать гораздо лучший способ решения проблемы. Не существует единого, простого решения для архитектуры, и делать вид, что это не так, - это плохая позиция для области разработки программного обеспечения.

Конкретная реализация веб-приложения, скорее всего, будет постоянно пересматриваться на протяжении всего срока его службы, но общая архитектура, скорее всего, будет редко меняться, а развиваться медленно. Архитектура безопасности не изменчива - нам нужна аутентификация сегодня, нам потребуется аутентификация завтра, и она понадобится нам через пять лет. Если мы примем обоснованные решения сегодня, мы сможем сэкономить много усилий, времени и денег, если будем выбирать и повторно использовать архитектурно совместимые решения. Например, десять лет назад многофакторная аутентификация редко применялась.

Если бы разработчики вложили средства в единую модель поставщика безопасных удостоверений, такую как SAML federated identity, поставщик удостоверений мог бы быть обновлен с учетом новых требований, таких как соответствие требованиям NIST 800-63, без изменения интерфейсов исходного приложения. Если многие приложения используют одну и ту же архитектуру безопасности и, следовательно, один и тот же компонент, все они сразу получат выгоду от этого обновления. Однако SAML не всегда будет оставаться лучшим или наиболее подходящим решением для аутентификации - возможно, его потребуется заменить на другие решения по мере изменения требований. Подобные изменения либо сложны, либо настолько дорогостоящие, что требуют полной переписки, либо совершенно невозможны без архитектуры безопасности.

В этой главе ASV охватывает основные аспекты любой надежной архитектуры безопасности: доступность, конфиденциальность, целостность обработки, отказ от ответственности и конфиденциальность. Каждый из этих принципов безопасности должен быть встроен и быть присущим всем приложениям. Крайне важно "сдвинуться влево", начиная с включения разработчиком контрольных списков безопасного кодирования, наставничества и обучения, кодирования и тестирования, создания, развертывания, настройки и операций и заканчивая последующим независимым тестированием, чтобы убедиться, что все средства контроля безопасности присутствуют и функционируют. Последним шагом было все, что мы делали как отрасль, но этого уже недостаточно, когда разработчики запускают код в производство десятки или сотни раз в день. Специалисты по безопасности приложений должны идти в ногу с гибкими методами, что означает внедрение инструментов разработчика, обучение кодированию и работу с разработчиками, а не критиковать проект через несколько месяцев после того, как все остальные продвинулись дальше.

Требования к жизненному циклу безопасной разработки программного обеспечения версии V1.1

#	Описание	L1	L2	L3	CWE

1.1.1	Убедитесь в использовании безопасного жизненного цикла разработки программного обеспечения, обеспечивающего безопасность на всех этапах разработки. (C1)	✓	✓	
1.1.2	Проверьте использование моделирования угроз для каждого изменения дизайна или планирования спринта для выявления угроз, планирования контрмер, облегчения соответствующих мер реагирования на риски и руководства по тестированию безопасности.	✓	✓	1053
1.1.3	Убедитесь, что все истории пользователей и функции содержат функциональные ограничения безопасности, такие как "Как пользователь, я должен иметь возможность просматривать и редактировать свой профиль. Я не должен иметь возможности просматривать или редактировать чей-либо профиль"	✓	✓	1110
1.1.4	Проверьте документацию и обоснование всех границ доверия приложения, компонентов и значительных потоков данных.	✓	✓	1059
1.1.5	Проверьте определение и анализ безопасности высокоуровневой архитектуры приложения и всех подключенных удаленных служб. (C1)	✓	✓	1059
1.1.6	Проверьте реализацию централизованных, простых (экономичных по дизайну), проверенных, безопасных и многократно используемых элементов управления безопасностью, чтобы избежать дублирования, отсутствия, неэффективности или небезопасных элементов управления. (C10)	✓	✓	637
1.1.7	Проверьте доступность контрольного списка безопасного кодирования, требований к безопасности, руководящих принципов или политики для всех разработчиков и тестировщиков.	✓	✓	637

Требования к архитектуре аутентификации V1.2

При разработке аутентификации не имеет значения, включена ли у вас надежная многофакторная аутентификация, если злоумышленник может сбросить учетную запись, обратившись в центр обработки вызовов и ответив на общеизвестные вопросы. При проверке подлинности все пути проверки подлинности должны иметь одинаковую силу.

#	Описание	L1	L2	L3	CWE
1.2.1	Проверьте использование уникальных или специальных учетных записей операционной системы с низкими привилегиями для всех компонентов приложений, служб и серверов. (C3)		✓	✓	250
1.2.2	Убедитесь, что связь между компонентами приложения, включая API, промежуточное ПО и слои данных, проходит проверку подлинности. Компоненты должны иметь наименее необходимые необходимые привилегии. (C3)		✓	✓	306
1.2.3	Убедитесь, что приложение использует один проверенный механизм проверки подлинности, который, как известно, является безопасным, может быть расширен, чтобы включать надежную проверку подлинности, и имеет достаточное ведение журнала и мониторинг для обнаружения злоупотреблений или нарушений учетной записи.		✓	✓	306
1.2.4	Убедитесь, что все пути проверки подлинности и API управления удостоверениями реализуют согласованную степень контроля безопасности проверки подлинности, чтобы не было более слабых альтернатив в зависимости от риска приложения.		✓	✓	306

Требования к архитектуре управления сеансами V1.3

Это заполнитель для будущих архитектурных требований.

V1.4 Архитектурные Требования к Управлению Доступом

#	Описание	L1	L2	L3	CWE
1.4.1	Убедитесь, что надежные точки обеспечения, такие как шлюзы управления доступом, серверы и бессерверные функции, обеспечивают контроль доступа. Никогда не применяйте контроль доступа к клиенту.		✓	✓	602
1.4.2	Убедитесь, что выбранное решение управления доступом является достаточно гибким для удовлетворения потребностей приложения.		✓	✓	284
1.4.3	Проверьте соблюдение принципа наименьших привилегий в функциях, файлах данных, URL-адресах, контроллерах, службах и других ресурсах. Это подразумевает защиту от подмены и повышения привилегий.		✓	✓	272
1.4.4	Убедитесь, что приложение использует единый и хорошо проверенный механизм контроля доступа для доступа к защищенным данным и ресурсам. Все запросы должны проходить через этот единый механизм, чтобы избежать копирования и вставки или небезопасных альтернативных путей. (C7)		✓	✓	284
1.4.5	Убедитесь, что используется управление доступом на основе атрибутов или функций, при котором код проверяет авторизацию пользователя для элемента функции/данных, а не только их роль. Разрешения по-прежнему должны распределяться с использованием ролей. (C7)		✓	✓	275

V1.5 Архитектурные требования к входу и выходу

В версии 4.0 мы отошли от термина "на стороне сервера" как от загруженного термина границы доверия. Граница доверия все еще остается нерешенной - принятие решений о ненадежных браузерах или клиентских устройствах можно обойти. Однако в современных основных архитектурных развертываниях точка обеспечения доверия кардинально изменилась. Поэтому, когда в ASV используется термин "уровень доверенных служб", мы подразумеваем любую надежную точку принудительного исполнения, независимо от местоположения, такую как микросервис, бессерверный API, серверная часть, доверенный API на клиентском устройстве с безопасной загрузкой, партнерский или внешний API и так далее.

Термин "ненадежный клиент" здесь относится к технологиям на стороне клиента, которые отображают уровень представления, обычно называемый "интерфейсными" технологиями. Термин "сериализация" здесь относится не только к отправке данных по проводу, таких как массив значений или получение и чтение структуры JSON, но и к передаче сложных объектов, которые могут содержать логику.

#	Описание	L1	L2	L3	CWE
1.5.1	Убедитесь, что требования к вводу и выводу четко определяют, как обрабатывать и обрабатывать данные в зависимости от типа, содержания и применимых законов, правил и других требований политики.		✓	✓	1029
1.5.2	Убедитесь, что сериализация не используется при общении с ненадежными клиентами. Если это невозможно, убедитесь, что применяются надлежащие средства контроля целостности (и,		✓	✓	502

	возможно, шифрования, если передаются конфиденциальные данные), чтобы предотвратить атаки десериализации, включая внедрение объектов.			
1.5.3	Убедитесь, что проверка ввода выполняется на уровне доверенного сервиса. (C5)	✓	✓	602
1.5.4	Убедитесь, что кодировка вывода выполняется близко к интерпретатору или интерпретатором, для которого она предназначена. (C4)	✓	✓	116

V1.6 Требования к Криптографической архитектуре

Приложения должны быть разработаны с использованием надежной криптографической архитектуры для защиты активов данных в соответствии с их классификацией. Шифрование всего-это расточительство, а не шифрование ничего-это юридическая халатность. Баланс должен быть достигнут, как правило, во время архитектурного или высокоуровневого проектирования, дизайнерских спринтов или архитектурных пиков. Разработка криптографии по ходу работы или ее модернизация неизбежно будут стоить намного дороже, чем ее безопасное внедрение с самого начала.

Архитектурные требования присущи всей кодовой базе, и поэтому их трудно объединить или интегрировать в тест. Требования к архитектуре требуют учета в стандартах кодирования на протяжении всего этапа кодирования и должны быть рассмотрены во время архитектуры безопасности, экспертных обзоров и обзоров кода или ретроспектив.

#	Описание	L1	L2	L3	CWE
1.6.1	Убедитесь, что существует явная политика управления криптографическими ключами и что жизненный цикл криптографических ключей соответствует стандарту управления ключами, такому как NIST SP 800-57.		✓	✓	320
1.6.2	Убедитесь, что потребители криптографических служб защищают ключевой материал и другие секреты с помощью хранилищ ключей или альтернатив на основе API.		✓	✓	320
1.6.3	Убедитесь, что все ключи и пароли являются заменяемыми и являются частью четко определенного процесса повторного шифрования конфиденциальных данных.		✓	✓	320
1.6.4	Убедитесь, что архитектура рассматривает секреты на стороне клиента, такие как симметричные ключи, пароли или токены API, как небезопасные и никогда не использует их для защиты конфиденциальных данных или доступа к ним.		✓	✓	320

1.17 Ошибки, Требования к архитектуре ведения журнала и аудита

#	Описание	L1	L2	L3	CWE
1.7.1	Убедитесь, что в системе используется общий формат и подход ведения журнала. (C9)		✓	✓	1009
1.7.2	Убедитесь, что журналы безопасно передаются в предпочтительно удаленную систему для анализа, обнаружения, оповещения и эскалации. (C9)		✓	✓	

1.1.8 Требования к архитектуре защиты данных и конфиденциальности

#	Описание	L1	L2	L3	CWE
1.8.1	Убедитесь, что все конфиденциальные данные идентифицированы и классифицированы по уровням защиты.		✓	✓	
1.8.2	Убедитесь, что все уровни защиты имеют связанный набор требований к защите, таких как требования к шифрованию, требования к целостности, хранению, конфиденциальности и другие требования к конфиденциальности, и что они применяются в архитектуре.		✓	✓	

1.1.9 Требования к архитектуре Коммуникаций

#	Описание	L1	L2	L3	CWE
1.9.1	Убедитесь, что приложение шифрует связь между компонентами, особенно если эти компоненты находятся в разных контейнерах, системах, сайтах или облачных провайдерах. (C3)		✓	✓	319
1.9.2	Убедитесь, что компоненты приложения проверяют подлинность каждой стороны в линии связи, чтобы предотвратить атаки "человек посередине". Например, компоненты приложения должны проверять сертификаты и цепочки TLS.		✓	✓	295

V1.10 Требования к Архитектуре Вредоносных Программ

#	Описание	L1	L2	L3	CWE
1.10.1	Убедитесь, что используется система управления исходным кодом с процедурами, обеспечивающими, чтобы регистрация сопровождалась проблемами или изменениями билетов. Система управления исходным кодом должна иметь контроль доступа и идентифицируемых пользователей, чтобы можно было отслеживать любые изменения.		✓	✓	284

V1.11 Требования к архитектуре бизнес-логики

#	Описание	L1	L2	L3	CWE
1.11.1	Проверьте определение и документацию всех компонентов приложения с точки зрения бизнес-функций или функций безопасности, которые они предоставляют.		✓	✓	1059
1.11.2	Убедитесь, что все важные потоки бизнес-логики, включая проверку подлинности, управление сеансами и контроль доступа, не имеют общего несинхронизированного состояния.		✓	✓	362
1.11.3	Убедитесь, что все потоки бизнес-логики высокой ценности, включая аутентификацию, управление сеансами и контроль доступа, являются потокобезопасными и устойчивыми к условиям времени проверки и времени использования.			✓	367

V1.12 Архитектурные требования К Безопасной Загрузке Файлов

#	Описание	L1	L2	L3	CWE

1.12.1	Убедитесь, что загруженные пользователем файлы хранятся за пределами корневого каталога веб-сайта.		✓	✓	552
1.12.2	Убедитесь, что загруженные пользователем файлы-если требуется, чтобы они отображались или загружались из приложения - обслуживаются либо потоковыми загрузками октетов, либо из несвязанного домена, такого как хранилище облачных файлов. Внедрите подходящую Политику безопасности контента (CSP), чтобы снизить риск от XSS-векторов или других атак из загруженного файла.		✓	✓	646

V1.13 Требования к архитектуре API

Это заполнитель для будущих архитектурных требований.

V1.14 Требования к архитектуре конфигурации

#	Описание	L1	L2	L3	CWE
1.14.1	Проверьте разделение компонентов с различными уровнями доверия с помощью четко определенных средств управления безопасностью, правил брандмауэра, шлюзов API, обратных прокси-серверов, облачных групп безопасности или аналогичных механизмов.		✓	✓	923
1.14.2	Убедитесь, что двоичные подписи, надежные соединения и проверенные конечные точки используются для развертывания двоичных файлов на удаленных устройствах.		✓	✓	494
1.14.3	Убедитесь, что конвейер сборки предупреждает об устаревших или небезопасных компонентах и предпринимает соответствующие действия.		✓	✓	1104
1.14.4	Убедитесь, что конвейер сборки содержит этап сборки для автоматической сборки и проверки безопасного развертывания приложения, особенно если инфраструктура приложения определена программным обеспечением, например сценарии сборки облачной среды.		✓	✓	
1.14.5	Убедитесь, что развертывания приложений надлежащим образом изолированы, изолированы и/или изолированы на сетевом уровне, чтобы задержать и удержать злоумышленников от атаки на другие приложения, особенно когда они выполняют конфиденциальные или опасные действия, такие как десериализация. (C5)		✓	✓	265
1.14.6	Убедитесь, что приложение не использует неподдерживаемые, небезопасные или устаревшие клиентские технологии, такие как плагины NSAPI, Flash, Shockwave, ActiveX, Silverlight, NACL или клиентские Java-апплеты.		✓	✓	477

Ссылки

Для получения дополнительной информации см. также:

- Шпаргалка по моделированию угроз OWASP
- Шпаргалка по анализу поверхности атаки OWASP
- Моделирование угроз OWASP
- Проект модели зрелости OWASP Software Assurance
- Microsoft SDL

- NIST SP 800-57

V2: Требования к Проверке подлинности

Цель Контроля

Аутентификация-это акт установления или подтверждения подлинности кого-либо (или чего-либо) и того, что утверждения, сделанные человеком или устройством, являются правильными, устойчивыми к олицетворению и предотвращают восстановление или перехват паролей.

Когда ASVS был впервые выпущен, имя пользователя + пароль были наиболее распространенной формой аутентификации за пределами систем высокой безопасности. Многофакторная аутентификация (MFA) была широко принята в кругах безопасности, но редко требовалась в других местах. По мере увеличения числа нарушений паролей идея о том, что имена пользователей каким-то образом являются конфиденциальными, а пароли неизвестны, сделала многие элементы управления безопасностью несостоятельными. Например, NIST 800-63 рассматривает имена пользователей и аутентификацию на основе знаний (KBA) как общедоступную информацию, SMS-и электронные уведомления как "ограниченные" типы средств проверки подлинности , а пароли как предварительно нарушенные. Эта реальность делает бесполезными аутентификаторы на основе знаний, восстановление SMS и электронной почты, история паролей, сложность и элементы управления сменой. Эти элементы управления всегда были менее чем полезны, часто заставляя пользователей придумывать слабые пароли каждые несколько месяцев, но с появлением более 5 миллиардов нарушений имени пользователя и пароля пришло время двигаться дальше.

Из всех разделов в ASV главы "Аутентификация" и "Управление сеансами" больше всего изменились. Внедрение эффективной, основанной на фактических данных передовой практики будет сложной задачей для многих, и это совершенно нормально. Мы должны начать переход к будущему после пароля уже сейчас.

NIST 800-63 - Современный стандарт аутентификации, основанный на фактических данных

NIST 800-63b является современным стандартом, основанным на фактических данных, и представляет собой наилучшие доступные рекомендации, независимо от их применимости. Стандарт полезен для всех организаций по всему миру, но особенно актуален для агентств США и тех, кто имеет дело с агентствами США.

Терминология NIST 800-63 поначалу может быть немного запутанной, особенно если вы привыкли использовать только аутентификацию по имени пользователя и паролю. Необходимы достижения в современной аутентификации, поэтому мы должны ввести терминологию, которая станет обыденностью в будущем, но мы понимаем сложность понимания, пока отрасль не определится с этими новыми терминами. В помощь мы подготовили глоссарий в конце этой главы. Мы перефразировали многие требования, чтобы удовлетворить цель требования, а не букву требования. Например, ASV использует термин "пароль", когда NIST использует "запомненный секрет" во всем этом стандарте.

Аутентификация ASVS V2, Управление сеансами V3 и, в меньшей степени, элементы управления доступом V4 были адаптированы для обеспечения соответствия подмножеству выбранных элементов управления NIST 800-63b, ориентированных на общие угрозы и часто используемые недостатки аутентификации. Если требуется полное соответствие NIST 800-63, пожалуйста, проконсультируйтесь с NIST 800-63.

Выбор соответствующего уровня NIST AAL

Стандарт проверки безопасности приложений попытался сопоставить ASV L1 с требованиями NIST AAL1, L2 с AAL2 и L3 с AAL3. Однако подход ASV уровня 1 в качестве "основных" элементов управления может не обязательно быть правильным уровнем AAL для проверки приложения или API. Например, если приложение является приложением уровня 3 или имеет нормативные требования к AAL3, Уровень 3 следует выбрать в разделах Управление сеансами V2 и V3. Выбор

уровня подтверждения проверки подлинности, совместимого с NIST (AAL), должен выполняться в соответствии с рекомендациями NIST 800-63b, изложенными в *разделе 6.2 Выбора AAL* в NIST 800-63b.

Легенда

Приложения всегда могут превышать требования текущего уровня, особенно если современная аутентификация включена в дорожную карту приложения. Ранее ACB требовало обязательного МИД. NIST не требует обязательного МИД. Поэтому мы использовали дополнительное обозначение в этой главе, чтобы указать, где ASV поощряет, но не требует контроля. В этом стандарте используются следующие ключи:

Мар к	Описание
	Не требуется
o	Рекомендуется, но не требуется
✓	Требуется

Требования к безопасности пароля V2.1

Пароли, называемые NIST 800-63 "Запоминаемыми секретами", включают пароли, пин-коды, шаблоны разблокировки, выбор правильного котенка или другого элемента изображения и парольные фразы. Они обычно считаются "чем-то, что вы знаете", и часто используются в качестве однофакторных аутентификаторов. Существуют значительные проблемы с продолжающимся использованием однофакторной аутентификации, включая миллиарды действительных имен пользователей и паролей, раскрытых в Интернете, пароли по умолчанию или слабые пароли, радужные таблицы и упорядоченные словари наиболее распространенных паролей.

Приложения должны настоятельно рекомендовать пользователям регистрироваться для многофакторной аутентификации и разрешать пользователям повторно использовать маркеры, которыми они уже обладают, такие как маркеры FIDO или U2F, или ссылаться на поставщика услуг по предоставлению учетных данных, который обеспечивает многофакторную аутентификацию.

Поставщики услуг предоставления учетных данных (CSP) предоставляют федеративную идентификацию для пользователей. Facebook, Twitter, Google или WeChat-это всего лишь несколько распространенных альтернатив, с помощью которых пользователи часто будут иметь несколько удостоверений с несколькими поставщиками услуг, таких как корпоративное удостоверение с использованием Azure AD, удостоверение Okta, Ping или Google, или удостоверение потребителя с использованием Facebook, Twitter, Google или WeChat. Этот список не является одобрением этих компаний или услуг, а просто побуждает разработчиков учитывать реальность того, что у многих пользователей есть множество установленных личностей. Организации должны рассмотреть возможность интеграции с существующими удостоверениями пользователей в соответствии с профилем рисков, характеризующим степень защиты удостоверений CSP. Например, маловероятно, что правительственная организация примет удостоверение в социальных сетях в качестве логина для конфиденциальных систем, поскольку легко создавать поддельные или выбрасывать удостоверения, в то время как компании мобильных игр вполне может потребоваться интеграция с основными платформами социальных сетей для расширения их активной базы игроков.

#	Описание	L1	L2	L3	CWE	NIST §

2.1.1	Убедитесь, что установленные пользователем пароли имеют длину не менее 12 символов (после объединения нескольких пробелов). (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Убедитесь, что пароли длиной 64 символа и более разрешены, но могут содержать не более 128 символов. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Убедитесь, что усечение пароля не выполняется. Однако последовательные несколько пробелов могут быть заменены одним пробелом. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Убедитесь, что в паролях запрещены любые печатные символы Юникода, включая символы, нейтральные к языку, такие как пробелы и смайлики.	✓	✓	✓	521	5.1.1.2
2.1.5	Убедитесь, что пользователи могут изменить свой пароль.	✓	✓	✓	620	5.1.1.2
2.1.6	Убедитесь, что для функции смены пароля требуется текущий и новый пароль пользователя.	✓	✓	✓	620	5.1.1.2
2.1.7	Убедитесь, что пароли, отправленные при регистрации учетной записи, входе в систему и смене пароля, сопоставляются с набором взломанных паролей либо локально (например, 1000 или 10000 наиболее распространенных паролей, соответствующих политике паролей системы), либо с помощью внешнего API. При использовании API следует использовать доказательство с нулевым знанием или другой механизм, чтобы гарантировать, что пароль в виде обычного текста не будет отправлен или использован для проверки статуса нарушения пароля. Если пароль взломан, приложение должно потребовать от пользователя установить новый пароль без взлома. (C6)	✓	✓	✓	521	5.1.1.2
2.1.8	Убедитесь, что предусмотрен измеритель силы пароля, чтобы помочь пользователям установить более надежный пароль.	✓	✓	✓	521	5.1.1.2
2.1.9	Убедитесь, что нет правил составления пароля, ограничивающих допустимый тип символов. Не должно быть никаких требований в отношении верхнего или нижнего регистра, цифр или специальных символов. (C6)	✓	✓	✓	521	5.1.1.2
2.1.10	Убедитесь, что нет периодической ротации учетных данных или требований к истории паролей.	✓	✓	✓	263	5.1.1.2
2.1.11	Убедитесь, что разрешены функции "вставки", помощники паролей браузера и внешние менеджеры паролей.	✓	✓	✓	521	5.1.1.2
2.1.12	Убедитесь, что пользователь может либо временно просмотреть весь скрытый пароль, либо временно просмотреть последний введенный символ пароля на платформах, которые не имеют этого в качестве встроенной функции.	✓	✓	✓	521	5.1.1.2

Примечание. Цель предоставления пользователю возможности временно просмотреть свой пароль или увидеть последний символ заключается в повышении удобства ввода учетных данных, особенно при использовании более длинных паролей, парольных фраз и диспетчеров паролей. Другая причина включения этого требования заключается в том, чтобы предотвратить или предотвратить ненужные отчеты о тестировании, требующие от организаций

переопределения поведения поля пароля встроенной платформы, чтобы удалить этот современный удобный интерфейс безопасности.

V2.2 Общие требования к Аутентификатору

Гибкость средства проверки подлинности имеет важное значение для приложений, защищенных в будущем. Средства проверки приложений рефакторинга, позволяющие использовать дополнительные средства проверки подлинности в соответствии с пользовательскими предпочтениями, а также упорядоченно удалять устаревшие или небезопасные средства проверки подлинности.

NIST рассматривает электронную почту и SMS как "ограниченные" типы средств проверки подлинности, и они, вероятно, будут удалены из NIST 800-63 и, следовательно, из ASV в какой-то момент в будущем. Приложения должны планировать дорожную карту, которая не требует использования электронной почты или SMS.

#	Описание	L1	L2	L3	CWE	NIST §
2.2.1	Убедитесь, что средства защиты от автоматизации эффективны для смягчения атак на проверку учетных данных с нарушением, грубую силу и блокировку учетных записей. Такие меры контроля включают блокировку наиболее распространенных взломанных паролей, мягкие блокировки, ограничение скорости, капчу, постоянно увеличивающиеся задержки между попытками, ограничения IP-адреса или ограничения, основанные на риске, такие как местоположение, первый вход на устройство, недавние попытки разблокировать учетную запись или аналогичные. Убедитесь, что в одной учетной записи возможно не более 100 неудачных попыток в час.	✓	✓	✓	307	5.2.2 / 5.1.1.2 / 5.1.4.2 / 5.1.5.2
2.2.2	Убедитесь, что использование слабых аутентификаторов (таких как SMS и электронная почта) ограничено вторичной проверкой и одобрением транзакции, а не заменой более безопасных методов аутентификации. Убедитесь, что более сильные методы предлагаются раньше, чем слабые, пользователи осведомлены о рисках или что приняты надлежащие меры для ограничения рисков компрометации учетной записи.	✓	✓	✓	304	5.2.10
2.2.3	Убедитесь, что безопасные уведомления отправляются пользователям после обновления сведений для проверки подлинности, таких как сброс учетных данных, изменения электронной почты или адреса, вход в систему из неизвестных или опасных мест. Предпочтительным является использование push - уведомлений, а не SMS или электронной почты, но в отсутствие push - уведомлений SMS или электронная почта приемлемы, если в уведомлении не раскрывается конфиденциальная информация.	✓	✓	✓	620	
2.2.4	Проверьте устойчивость олицетворения к фишингу, например, при использовании многофакторной аутентификации, криптографических устройств с намерением (таких как подключенные ключи с принудительной проверкой подлинности) или на более высоких уровнях AAL, сертификатов на стороне клиента.			✓	308	5.2.5

2.2.5	Убедитесь, что там, где Поставщик службы учетных данных (CSP) и приложение, проверяющее проверку подлинности, разделены, между двумя конечными точками действует протокол TLS с взаимной проверкой подлинности.		✓		319	5.2.6
2.2.6	Проверьте устойчивость к воспроизведению с помощью разрешенного использования устройств с одноразовыми паролями (OTP), криптографических аутентификаторов или кодов поиска.		✓		308	5.2.8
2.2.7	Проверьте намерение для аутентификации, потребовав ввода токена OTP или действия, инициированного пользователем, например, нажатия кнопки на аппаратной клавише FIDO.		✓		308	5.2.9

Требования к жизненному циклу аутентификатора версии 2.3

Аутентификаторы-это пароли, программные токены, аппаратные токены и биометрические устройства. Жизненный цикл средств проверки подлинности имеет решающее значение для безопасности приложения - если кто-либо может самостоятельно зарегистрировать учетную запись без подтверждения личности, утверждение об удостоверении может вызывать мало доверия. Для сайтов социальных сетей, таких как Reddit, это совершенно нормально. Для банковских систем большое внимание уделяется регистрации и выдаче учетных данных и устройств, что имеет решающее значение для безопасности приложения.

Примечание. Пароли не должны иметь максимальный срок действия или подлежать смене паролей. Пароли следует проверять на нарушение, а не на регулярную замену.

#	Описание	L1	L2	L3	CWE	NIST §
2.3.1	Убедитесь, что сгенерированные системой исходные пароли или коды активации ДОЛЖНЫ быть безопасно сгенерированы случайным образом, ДОЛЖНЫ содержать не менее 6 символов, МОГУТ содержать буквы и цифры и истекать через короткий промежуток времени. Эти первоначальные секреты не должны быть разрешены, чтобы стать долгосрочным паролем.	✓	✓	✓	330	5.1.1.2 / A. 3
2.3.2	Убедитесь, что регистрация и использование устройств аутентификации, предоставляемых подписчиком, поддерживаются, например, токены U2F или FIDO.		✓	✓	308	6.1.3
2.3.3	Убедитесь, что инструкции по обновлению отправляются с достаточным временем для обновления аутентификаторов с привязкой ко времени.		✓	✓	287	6.1.4

V2.4 Требования к хранению учетных данных

Архитекторы и разработчики должны придерживаться этого раздела при создании или рефакторинге кода. Этот раздел может быть полностью проверен только с помощью проверки исходного кода или с помощью безопасных модульных и интеграционных тестов. Тестирование на проникновение не может выявить ни одну из этих проблем.

Список утвержденных функций одностороннего вывода ключей подробно описан в NIST 800-63 B, раздел 5.1.1.2, и в BSI Kryptographische Verfahren: Empfehlungen und Schlussellängen (2018). Вместо этих вариантов можно выбрать новейший национальный или региональный алгоритм и стандарты длины ключа.

Этот раздел не может быть проверен на проникновение, поэтому элементы управления не помечены как L1. Тем не менее, этот раздел имеет жизненно важное значение для безопасности

учетных данных в случае их кражи, поэтому, если вы разветвляете ASV для руководства по архитектуре или кодированию или контрольного списка проверки исходного кода, пожалуйста, верните эти элементы управления в L1 в вашей частной версии.

#	Описание	L1	L2	L3	CWE	NIST §
2.4.1	Убедитесь, что пароли хранятся в форме, устойчивой к атакам в автономном режиме. Пароли ДОЛЖНЫ быть засолены и хэшированы с использованием одобренной функции одностороннего получения ключей или хэширования паролей. Функции получения ключей и хэширования паролей используют пароль, соль и коэффициент затрат в качестве входных данных при создании хэша пароля. (C6)		✓	✓	916	5.1.1.2
2.4.2	Убедитесь, что длина соли составляет не менее 32 бит и выбирается произвольно, чтобы свести к минимуму коллизии значений соли между сохраненными хэшами. Для каждой учетной записи ДОЛЖНО храниться уникальное значение соли и полученный хэш. (C6)		✓	✓	916	5.1.1.2
2.4.3	Убедитесь, что при использовании PBKDF2 количество итераций ДОЛЖНО быть настолько большим, насколько позволяет производительность сервера проверки, обычно не менее 100 000 итераций. (C6)		✓	✓	916	5.1.1.2
2.4.4	Убедитесь, что при использовании bcrypt коэффициент работы ДОЛЖЕН быть настолько большим, насколько позволяет производительность сервера проверки, обычно не менее 13. (C6)		✓	✓	916	5.1.1.2
2.4.5	Убедитесь, что выполняется дополнительная итерация функции вывода ключа, используя значение соли, которое является секретным и известно только проверяющему. Сгенерируйте значение соли с помощью одобренного генератора случайных битов [SP 800-90Ar1] и обеспечьте, по крайней мере, минимальный уровень безопасности, указанный в последней редакции SP 800-131A. Значение секретной соли ДОЛЖНО храниться отдельно от хэшированных паролей (например, в специализированном устройстве, таком как аппаратный модуль безопасности).		✓	✓	916	5.1.1.2

Там, где упоминаются стандарты США, вместо стандарта США или в дополнение к нему может быть использован региональный или местный стандарт по мере необходимости.

V2.5 Требования к восстановлению учетных данных

#	Описание	L1	L2	L3	CWE	NIST §
2.5.1	Убедитесь, что секрет начальной активации или восстановления, созданный системой, не отправляется пользователю открытым текстом. (C6)	✓	✓	✓	640	5.1.1.2
2.5.2	Подсказки для проверки пароля или проверка подлинности на основе знаний (так называемые "секретные вопросы") отсутствуют.	✓	✓	✓	640	5.1.1.2
2.5.3	Проверка пароля восстановление учетных данных никак не раскрывает текущий пароль. (C6)	✓	✓	✓	640	5.1.1.2

2.5.4	Убедитесь, что общие учетные записи или учетные записи по умолчанию отсутствуют (например, "root", "admin" или "sa").	✓	✓	✓	16	5.1.1.2 / A. 3
2.5.5	Убедитесь, что при изменении или замене фактора проверки подлинности пользователь уведомлен об этом событии.	✓	✓	✓	304	6.1.2.3
2.5.6	Проверьте забытый пароль, и другие пути восстановления используют безопасный механизм восстановления, такой как OTP на основе времени (TOTP) или другой мягкий маркер, мобильный push-запрос или другой механизм автономного восстановления. (C6)	✓	✓	✓	640	5.1.1.2
2.5.7	Убедитесь, что в случае потери факторов OTP или многофакторной аутентификации подтверждение проверки подлинности выполняется на том же уровне, что и при регистрации.		✓	✓	308	6.1.2.3

V2.6 Требования к секретному верификатору поиска

Секреты поиска-это предварительно сгенерированные списки секретных кодов, подобные номерам авторизации транзакций (TAN), кодам восстановления в социальных сетях или сетке, содержащей набор случайных значений. Они безопасно распространяются среди пользователей. Эти коды поиска используются один раз, и после их использования секретный список поиска удаляется. Этот тип аутентификатора считается "тем, что у вас есть".

#	Описание	L1	L2	L3	CWE	NIST §
2.6.1	Убедитесь, что секреты поиска можно использовать только один раз.		✓	✓	308	5.1.2.2
2.6.2	Убедитесь, что секреты поиска обладают достаточной случайностью (112 бит энтропии) или, если энтропия меньше 112 бит, засолены уникальной и случайной 32-разрядной солью и хэшированы с помощью одобренного одностороннего хэша.		✓	✓	330	5.1.2.2
2.6.3	Убедитесь, что секреты поиска устойчивы к автономным атакам, таким как предсказуемые значения.		✓	✓	310	5.1.2.2

V2.7 Требования к внешнему верификатору

В прошлом обычным внешним средством проверки было бы электронное письмо или SMS, содержащее ссылку на сброс пароля. Злоумышленники используют этот слабый механизм для сброса учетных записей, которые они еще не контролируют, например, завладев учетной записью электронной почты пользователя и повторно используя любые обнаруженные ссылки для сброса. Есть лучшие способы обработки внеплановой проверки.

Защищенные внешние аутентификаторы-это физические устройства, которые могут взаимодействовать с верификатором по защищенному вторичному каналу. Примеры включают в себя push-уведомления на мобильные устройства. Этот тип аутентификатора считается "тем, что у вас есть". Когда пользователь желает пройти аутентификацию, проверяющее приложение отправляет сообщение внешнему аутентификатору через соединение с аутентификатором напрямую или косвенно через стороннюю службу. Сообщение содержит код аутентификации (как правило, случайное шестизначное число или диалоговое окно модального утверждения). Проверяющее приложение ожидает получения кода аутентификации по основному каналу и сравнивает хэш полученного значения с хэшем исходного кода аутентификации. Если они совпадают, внеполосный верификатор может предположить, что пользователь прошел аутентификацию.

ASV предполагает, что только несколько разработчиков будут разрабатывать новые внешние средства проверки подлинности, такие как push-уведомления, и, таким образом, следующие элементы управления ASV применяются к проверяющим, таким как API проверки подлинности, приложениям и реализациям единого входа. При разработке нового внешнего средства аутентификации, пожалуйста, обратитесь к NIST 800-63B § 5.1.3.1.

Не допускаются небезопасные внешние средства проверки подлинности, такие как электронная почта и VOIP. В настоящее время аутентификация по ТСОП и SMS" ограничена " NIST и должна быть отменена в пользу push-уведомлений или аналогичных. Если вам необходимо использовать внешнюю аутентификацию по телефону или SMS, пожалуйста, обратитесь к § 5.1.3.3.

#	Описание	L1	L2	L3	CWE	NIST §
2.7.1	Убедитесь, что по умолчанию не предлагаются аутентификаторы внеполосного открытого текста (NIST "с ограниченным доступом"), такие как SMS или PSTN, и сначала предлагаются более эффективные альтернативы, такие как push-уведомления.	✓	✓	✓	287	5.1.3.2
2.7.2	Убедитесь, что срок действия внешнего средства проверки подлинности, кодов или маркеров истекает через 10 минут.	✓	✓	✓	287	5.1.3.2
2.7.3	Убедитесь, что запросы внешней проверки подлинности, коды или маркеры могут использоваться только один раз и только для исходного запроса проверки подлинности.	✓	✓	✓	287	5.1.3.2
2.7.4	Убедитесь, что внеполосный аутентификатор и верификатор обмениваются данными по защищенному независимому каналу.	✓	✓	✓	523	5.1.3.2
2.7.5	Убедитесь, что внеполосный верификатор сохраняет только хэшированную версию кода аутентификации.		✓	✓	256	5.1.3.2
2.7.6	Убедитесь, что исходный код аутентификации генерируется безопасным генератором случайных чисел, содержащим не менее 20 бит энтропии (обычно достаточно шести цифровых случайных чисел).		✓	✓	310	5.1.3.2

V2.8 Требования к однофакторному или многофакторному однократному верификатору

Однофакторные одноразовые пароли (OTP)-это физические или программные токены, которые отображают постоянно меняющийся псевдослучайный одноразовый вызов. Эти устройства затрудняют фишинг (олицетворение), но не делают его невозможным. Этот тип аутентификатора считается "тем, что у вас есть". Многофакторные токены похожи на однофакторные OTP, но для создания окончательной OTP требуется ввести действительный PIN-код, биометрическую разблокировку, вставку USB или сопряжение NFC или какое-либо дополнительное значение (например, калькуляторы подписи транзакций).

#	Описание	L1	L2	L3	CWE	NIST §
2.8.1	Убедитесь, что срок службы OTP, основанный на времени, имеет определенный срок службы до истечения срока действия.	✓	✓	✓	613	5.1.4.2 / 5.1.5.2
2.8.2	Убедитесь, что симметричные ключи, используемые для проверки отправленных OTP, имеют высокую степень защиты, например, с помощью аппаратного модуля		✓	✓	320	5.1.4.2 / 5.1.5.2

	безопасности или безопасного хранилища ключей на основе операционной системы.				
2.8.3	Убедитесь, что одобренные криптографические алгоритмы используются при генерации, заполнении и проверке OTP.	✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	Убедитесь, что OTP, основанный на времени, может быть использован только один раз в течение срока действия.	✓	✓	287	5.1.4.2 / 5.1.5.2
2.8.5	Убедитесь, что если многофакторный OTP-токен, основанный на времени, повторно используется в течение срока действия, он регистрируется и отклоняется, а владельцу устройства отправляются безопасные уведомления.	✓	✓	287	5.1.5.2
2.8.6	Убедитесь, что физический однофакторный генератор OTP может быть отозван в случае кражи или другой потери. Убедитесь, что отзыв немедленно вступает в силу во всех сеансах входа, независимо от местоположения.	✓	✓	613	5.2.1
2.8.7	Убедитесь, что биометрические аутентификаторы можно использовать только в качестве второстепенных факторов в сочетании с тем, что у вас есть, или с тем, что вы знаете.	o	✓	308	5.2.3

Требования к Верификатору криптографического программного обеспечения и устройств версии V2.9

Криптографические ключи безопасности-это смарт-карты или ключи FIDO, где пользователь должен подключить или подключить криптографическое устройство к компьютеру для завершения аутентификации. Проверяющие отправляют одноразовый вызов криптографическим устройствам или программному обеспечению, и устройство или программное обеспечение вычисляет ответ на основе надежно сохраненного криптографического ключа.

Требования к однофакторным криптографическим устройствам и программному обеспечению, а также к многофакторным криптографическим устройствам и программному обеспечению одинаковы, поскольку проверка криптографического аутентификатора подтверждает наличие фактора аутентификации.

#	Описание	L1	L2	L3	CWE	NIST §
2.9.1	Убедитесь, что криптографические ключи, используемые при проверке, хранятся надежно и защищены от раскрытия, например, с помощью доверенного платформенного модуля (TPM) или Аппаратного модуля безопасности (HSM) или службы ОС, которая может использовать это безопасное хранилище.		✓	✓	320	5.1.7.2
2.9.2	Убедитесь, что длина вызова не меньше 64 бит и статистически уникальна или уникальна в течение всего срока службы криптографического устройства.		✓	✓	330	5.1.7.2
2.9.3	Убедитесь, что утвержденные криптографические алгоритмы используются при создании, заполнении и проверке.		✓	✓	327	5.1.7.2

V2.10 Требования к Проверке подлинности Службы

Этот раздел не поддается тестированию на проникновение, поэтому не содержит требований L1. Однако при использовании в обзоре архитектуры, кодирования или безопасного кода, пожалуйста, предположите, что программное обеспечение (так же, как хранилище ключей Java) является минимальным требованием по L1. Хранение секретов открытым текстом недопустимо ни при каких обстоятельствах.

#	Описание	L1	L2	L3	CWE	NIST §
2.10.1	Убедитесь, что внутрисервисные секреты не зависят от неизменных учетных данных, таких как пароли, ключи API или общие учетные записи с привилегированным доступом.		Поддержка ОС	HSM	287	5.1.1.1
2.10.2	Убедитесь, что, если для проверки подлинности службы требуются пароли, используемая учетная запись службы не является учетными данными по умолчанию (например, в некоторых службах во время установки по умолчанию используются учетные данные root/root или admin/admin).		Поддержка ОС	HSM	255	5.1.1.1
2.10.3	Убедитесь, что пароли хранятся с достаточной защитой для предотвращения атак на восстановление в автономном режиме, включая доступ к локальной системе.		Поддержка ОС	HSM	522	5.1.1.1
2.10.4	Проверка паролей, интеграция с базами данных и сторонними системами, исходные данные и внутренние секреты, а также ключи API надежно управляются и не включаются в исходный код или хранятся в репозиториях исходного кода. Такое хранилище ДОЛЖНО противостоять атакам в автономном режиме. Для хранения паролей рекомендуется использовать защищенное хранилище программных ключей (L1), аппаратный доверенный платформенный модуль или HSM (L3).		Поддержка ОС	HSM	798	

Дополнительные требования Агентства США

Агентства США имеют обязательные требования, касающиеся NIST 800-63. Стандарт проверки безопасности приложений всегда касался 80% элементов управления, применимых почти к 100% приложений, а не последних 20% расширенных элементов управления или тех, которые имеют ограниченную применимость. Таким образом, ASV является строгим подмножеством NIST 800-63, особенно для классификаций IAL1/2 и AAL1/2, но не является достаточно полным, особенно в отношении классификаций IAL3/AAL3.

Мы настоятельно призываем правительственные учреждения США пересмотреть и внедрить NIST 800-63 в полном объеме.

Словарь терминов

Срок	Значение
CSP	Поставщик услуг по предоставлению учетных данных также называется Поставщиком удостоверений личности
Аутентификатор	Код, который проверяет подлинность пароля, токена, MFA, федеративного утверждения и так далее.

Верификатор	"Организация, которая проверяет личность заявителя путем проверки владения заявителем одним или двумя аутентификаторами и контроля над ними с использованием протокола аутентификации. Для этого верификатору также может потребоваться проверить учетные данные, которые связывают аутентификатора (- ов) с идентификатором подписчика, и проверить их статус"
OTP	Одноразовый пароль
SFA	Однофакторные аутентификаторы, такие как то, что вы знаете (заученные секреты, пароли, парольные фразы, ПИН-коды), то, чем вы являетесь (биометрия, отпечатки пальцев, сканирование лица), или то, что у вас есть (токены OTP, криптографическое устройство, такое как смарт-карта),
МИД	Многофакторная аутентификация, которая включает в себя два или более отдельных фактора

Ссылки

Для получения дополнительной информации см. также:

- NIST 800-63 - Руководство по цифровой идентификации
- NIST 800-63 A - Регистрация и проверка личности
- NIST 800-63 B - Аутентификация и управление жизненным циклом
- NIST 800-63 C - Федерация и утверждения
- NIST 800-63 ЧАСТО ЗАДАВАЕМЫЕ ВОПРОСЫ
- Руководство по тестированию OWASP 4.0: Тестирование на проверку подлинности
- Шпаргалка OWASP - Хранение паролей
- Шпаргалка OWASP - Забыли пароль
- Шпаргалка OWASP - Выбор и использование вопросов безопасности

V3: Требования к Проверке Управления Сеансами

Цель Управления

Одним из основных компонентов любого веб-приложения или API с отслеживанием состояния является механизм, с помощью которого оно контролирует и поддерживает состояние пользователя или устройства, взаимодействующего с ним. Управление сеансами изменяет протокол без состояния на протокол с сохранением состояния, что имеет решающее значение для разграничения различных пользователей или устройств.

Убедитесь, что проверенное приложение удовлетворяет следующим требованиям к управлению сеансами высокого уровня:

- Сеансы уникальны для каждого человека и не могут быть угаданы или разделены.
- Сеансы становятся недействительными, когда они больше не требуются, и истекают во время периодов бездействия.

Как отмечалось ранее, эти требования были адаптированы для соответствия подмножеству выбранных элементов управления NIST 800-63b, ориентированных на общие угрозы и часто используемые недостатки аутентификации. Предыдущие требования к проверке были отменены, отменены или в большинстве случаев адаптированы для строгого соответствия целям обязательных требований NIST 800-63b.

Требования к Проверке безопасности

V3.1 Основные Требования К Управлению Сеансами

#	Описание	L 1	L 2	L 3	CWE	NIST §
3.1.1	Убедитесь, что приложение никогда не показывает маркеры сеанса в параметрах URL.	✓	✓	✓	598	

V3.2 Требования к привязке Сеанса

#	Описание	L 1	L 2	L 3	CWE	NIST §
3.2.1	Убедитесь, что приложение создает новый токен сеанса при аутентификации пользователя. (C6)	✓	✓	✓	384	7.1
3.2.2	Убедитесь, что токены сеанса обладают по крайней мере 64 битами энтропии. (C6)	✓	✓	✓	331	7.1
3.2.3	Убедитесь, что приложение хранит токены сеанса только в браузере, используя безопасные методы, такие как надлежащим образом защищенные файлы cookie (см. раздел 3.4) или хранилище сеансов HTML 5.	✓	✓	✓	539	7.1
3.2.4	Убедитесь, что маркер сеанса сгенерирован с использованием утвержденных криптографических алгоритмов. (C6)		✓	✓	331	7.1

TLS или другой безопасный транспортный канал является обязательным для управления сеансами. Это описано в главе Безопасность связи.

V3.3 Требования к выходу из сеанса и времени ожидания

Время ожидания сеанса было согласовано с NIST 800-63, что позволяет значительно увеличить время ожидания сеанса по сравнению с традиционно разрешенными стандартами безопасности. Организации должны ознакомиться с приведенной ниже таблицей, и если желательно более длительное время ожидания, основанное на риске приложения, значением NIST должно быть верхнее значение времени ожидания простоя сеанса.

L1 в этом контексте является IAL1/AAL1, L2-IAL2/AAL3, L3-IAL3/AAL3. Для IAL2/AAL2 и IAL3/AAL3, чем меньше время простоя, тем меньше время простоя для выхода из системы или повторной аутентификации для возобновления сеанса.

#	Описание	L1	L2	L3	CWE	NIST §
3.3.1	Убедитесь, что выход из системы и истечение срока действия делают недействительным маркер сеанса, так что кнопка "Назад" или нижестоящая проверяющая сторона не возобновляет аутентифицированный сеанс, в том числе между проверяющими сторонами. (C6)	✓	✓	✓	613	7.1
3.3.2	Если аутентификаторы разрешают пользователям оставаться в системе, убедитесь, что повторная аутентификация периодически выполняется как при активном использовании, так и после периода простоя. (C6)	30 дней	12 часов или 30 минут бездействия, 2FA по желанию	12 часов или 15 минут бездействия, с 2FA	613	7.2
3.3.3	Убедитесь, что приложение предоставляет возможность завершить все другие активные сеансы после успешной смены пароля (включая изменение с помощью сброса/восстановления пароля), и что это эффективно для приложения, объединенного входа в систему (если имеется) и любых доверяющих сторон.		✓	✓	613	
3.3.4	Убедитесь, что пользователи могут просматривать и (повторно введя учетные данные для входа) выходить из любых или всех активных в данный момент сеансов и устройств.		✓	✓	613	7.1

V3.4 Управление сеансами на основе файлов cookie

#	Описание	L1	L2	L3	CWE	NIST §
3.4.1	Убедитесь, что маркеры сеанса на основе файлов cookie имеют атрибут "Безопасный". (C6)	✓	✓	✓	614	7.1.1
3.4.2	Убедитесь, что маркеры сеанса на основе файлов cookie имеют атрибут "HttpOnly". (C6)	✓	✓	✓	1004	7.1.1
3.4.3	Убедитесь, что маркеры сеанса на основе файлов cookie используют атрибут "SameSite", чтобы ограничить подверженность атакам подделки межсайтовых запросов. (C6)	✓	✓	✓	16	7.1.1
3.4.4	Убедитесь, что маркеры сеанса на основе файлов cookie используют префикс "__Хост-" (см. Ссылки) для обеспечения конфиденциальности файлов cookie сеанса.	✓	✓	✓	16	7.1.1

3.4.5	Убедитесь, что, если приложение опубликовано под доменным именем с другими приложениями, которые устанавливают или используют файлы cookie сеанса, которые могут переопределять или раскрывать файлы cookie сеанса, установите атрибут пути в маркерах сеанса на основе файлов cookie, используя максимально точный возможный путь. (C6)	✓	✓	✓	16	7.1.1

V3.5 Управление сеансами на основе токенов

Управление сеансами на основе токенов включает в себя ключи JWT, OAuth, SAML и API. Из них ключи API, как известно, являются слабыми и не должны использоваться в новом коде.

#	Описание	L1	L2	L3	CWE	NIST §
3.5.1	Проверка приложения позволяет пользователям отзывать токены OAuth, которые формируют доверительные отношения со связанными приложениями.		✓	✓	290	7.1.2
3.5.2	Убедитесь, что приложение использует маркеры сеанса, а не статические секреты и ключи API, за исключением устаревших реализаций.		✓	✓	798	
3.5.3	Убедитесь, что маркеры сеанса без состояния используют цифровые подписи, шифрование и другие контрмеры для защиты от взлома, оболочки, воспроизведения, нулевого шифра и атак с заменой ключей.		✓	✓	345	

V3.6 Повторная аутентификация из Федерации или Утверждения

Этот раздел относится к тем, кто пишет код Проверяющей стороны (RP) или Поставщика услуг учетных данных (CSP). Если вы полагаетесь на код, реализующий эти функции, убедитесь, что эти проблемы решены правильно.

#	Описание	L1	L2	L3	CWE	NIST §
3.6.1	Убедитесь, что проверяющие стороны указывают максимальное время аутентификации Поставщикам услуг аутентификации (CSP) и что CSP повторно аутентифицируют подписчика, если они не использовали сеанс в течение этого периода.			✓	613	7.2.1
3.6.2	Убедитесь, что Поставщики услуг учетных данных (CSP) сообщают Проверяющим сторонам (RP) о последнем событии аутентификации, чтобы RP могли определить, нужно ли им повторно аутентифицировать пользователя.			✓	613	7.2.1

V3.7 Защита От Эксплойтов Управления Сеансами

Существует небольшое количество атак по управлению сеансами, некоторые из которых связаны с пользовательским интерфейсом (UX) сеансов. Ранее, основываясь на требованиях ISO 27002, ASVS требовал блокировки нескольких одновременных сеансов. Блокировка одновременных сеансов больше неуместна не только потому, что у современных пользователей много устройств или приложение представляет собой API без сеанса браузера, но и в большинстве этих реализаций побеждает последний аутентификатор, которым часто является злоумышленник. В этом разделе приведены основные рекомендации по предотвращению, задержке и обнаружению атак управления сеансами с использованием кода.

Описание полуоткрытой атаки

В начале 2018 года несколько финансовых учреждений были скомпрометированы с помощью того, что злоумышленники назвали "полуоткрытыми атаками". Этот термин прижился в отрасли.

Злоумышленники нанесли удары по нескольким учреждениям с разными проприетарными кодовыми базами, и действительно, похоже, что в одних и тех же учреждениях разные кодовые базы. Полуоткрытая атака использует недостаток шаблона проектирования, обычно встречающийся во многих существующих системах аутентификации, управления сеансами и контроля доступа.

Злоумышленники начинают полуоткрытую атаку, пытаясь заблокировать, сбросить или восстановить учетные данные. Популярный шаблон проектирования управления сеансами повторно использует объекты/модели сеанса профиля пользователя между не прошедшими проверку подлинности, полуавторизованными (сброс пароля, забытое имя пользователя) и полностью аутентифицированным кодом. Этот шаблон проектирования заполняет допустимый объект сеанса или маркер, содержащий профиль жертвы, включая хэши паролей и роли. Если при проверке контроля доступа в контроллерах или маршрутизаторах неверно проверяется, что пользователь полностью вошел в систему, злоумышленник сможет действовать как пользователь. Атаки могут включать изменение пароля пользователя на известное значение, обновление адреса электронной почты для выполнения действительного сброса пароля, отключение многофакторной аутентификации или регистрацию нового устройства MFA, раскрытие или изменение ключей API и так далее.

#	Описание	L1	L2	L3	CWE	NIST §
3.7.1	Убедитесь, что приложение обеспечивает полный, действительный сеанс входа в систему или требует повторной аутентификации или дополнительной проверки, прежде чем разрешать какие-либо конфиденциальные транзакции или изменения учетной записи.	✓	✓	✓	306	

Ссылки

Для получения дополнительной информации см. также:

- Руководство по тестированию OWASP 4.0: Тестирование управления сеансами
- Шпаргалка по управлению сеансами OWASP
- Set-Cookie Сведения о префиксе хоста

V4: Требования к Проверке Контроля Доступа

Цель Контроля

Авторизация-это концепция предоставления доступа к ресурсам только тем, кому разрешено их использовать. Убедитесь, что проверенное приложение удовлетворяет следующим требованиям высокого уровня:

- Лица, получающие доступ к ресурсам, имеют для этого действительные учетные данные.
- Пользователи связаны с четко определенным набором ролей и привилегий.
- Метаданные ролей и разрешений защищены от воспроизведения или изменения.

Требования к Проверке Безопасности

V4.1 Общая Конструкция Управления Доступом

#	Описание	L1	L2	L3	CWE
4.1.1	Убедитесь, что приложение применяет правила управления доступом на уровне доверенной службы, особенно если присутствует управление доступом на стороне клиента, которое можно обойти.	✓	✓	✓	602
4.1.2	Убедитесь, что конечные пользователи не могут управлять всеми атрибутами пользователей и данных, а также информацией о политике, используемой элементами управления доступом, без специальной авторизации.	✓	✓	✓	639
4.1.3	Убедитесь, что существует принцип наименьших привилегий - пользователи должны иметь возможность получать доступ к функциям, файлам данных, URL-адресам, контроллерам, службам и другим ресурсам только в том случае, если у них есть специальное разрешение. Это подразумевает защиту от подмены и повышения привилегий. (C7)	✓	✓	✓	285
4.1.4	Убедитесь, что по умолчанию действует принцип запрета, согласно которому новые пользователи/роли запускаются с минимальными разрешениями или без них, а пользователи/роли не получают доступ к новым функциям до тех пор, пока доступ явно не будет назначен. (C7)	✓	✓	✓	276
4.1.5	Убедитесь, что элементы управления доступом надежно отказывают, в том числе при возникновении исключения. (C10)	✓	✓	✓	285

V4.2 Управление доступом На Рабочем Уровне

#	Описание	L1	L2	L3	CWE
4.2.1	Убедитесь, что конфиденциальные данные и API защищены от небезопасных атак Прямой ссылки на объекты (IDOR), направленных на создание, чтение, обновление и удаление записей, таких как создание или обновление чужой записи, просмотр всех записей или удаление всех записей.	✓	✓	✓	639
4.2.2	Убедитесь, что приложение или платформа применяет сильный механизм защиты от CSRF для защиты функций, прошедших проверку подлинности, а эффективные средства защиты от автоматизации или защиты от CSRF защищают функции, не прошедшие проверку подлинности.	✓	✓	✓	352

V4.3 Другие Соображения По Контролю Доступа

#	Описание	L 1	L 2	L 3	CWE
4.3.1	Убедитесь, что административные интерфейсы используют соответствующую многофакторную аутентификацию для предотвращения несанкционированного использования.	✓	✓	✓	419
4.3.2	Убедитесь, что просмотр каталогов отключен, если это не требуется преднамеренно. Кроме того, приложения не должны разрешать обнаружение или раскрытие метаданных файлов или каталогов, таких как папки Thumbs.db, .DS_Store, .git или .svn.	✓	✓	✓	548
4.3.3	Убедитесь, что приложение имеет дополнительную авторизацию (например, повышенную или адаптивную аутентификацию) для систем с более низкой стоимостью и / или разделение обязанностей для приложений с высокой стоимостью для обеспечения контроля за противодействием мошенничеству в соответствии с риском применения и прошлых мошенничеств.		✓	✓	732

Ссылки

Для получения дополнительной информации см. также:

- Руководство по тестированию OWASP 4.0: Авторизация
- Шпаргалка OWASP: Контроль доступа
- Шпаргалка OWASP CSRF
- Шпаргалка OWASP REST

V5: Требования к валидации, санитарной обработке и проверке кодирования

Цель Контроля

Наиболее распространенным недостатком безопасности веб-приложений является неспособность правильно проверить входные данные, поступающие от клиента или среды, прежде чем непосредственно использовать их без какой-либо кодировки вывода. Эта слабость приводит почти ко всем существенным уязвимостям в веб-приложениях, таким как межсайтовые сценарии (XSS), внедрение SQL-кода, внедрение интерпретатора, атаки на языковой стандарт/Юникод, атаки на файловую систему и переполнение буфера.

Убедитесь, что проверенное приложение удовлетворяет следующим требованиям высокого уровня:

- Архитектура проверки ввода и кодирования вывода имеет согласованный конвейер для предотвращения инъекционных атак.
- Входные данные строго типизированы, проверены, проверены по диапазону или длине или, в худшем случае, очищены или отфильтрованы.
- Выходные данные кодируются или экранируются в соответствии с контекстом данных как можно ближе к интерпретатору.

С современной архитектурой веб - приложений кодирование вывода становится более важным, чем когда-либо. В некоторых сценариях трудно обеспечить надежную проверку ввода, поэтому использование более безопасного API, такого как параметризованные запросы, платформы шаблонов с автоматическим экранированием или тщательно выбранная кодировка вывода, имеет решающее значение для безопасности приложения.

V5.1 Требования к Проверке входных данных

Правильно реализованные средства проверки ввода, использующие положительные списки разрешений и строгий ввод данных, могут устранить более 90% всех инъекционных атак. Проверка длины и диапазона может еще больше уменьшить это. Построение безопасной проверки ввода требуется во время архитектуры приложений, спринтов проектирования, кодирования, а также модульного и интеграционного тестирования. Хотя многие из этих элементов невозможно найти в тестах на проникновение, результаты их невыполнения обычно содержатся в требованиях к кодированию вывода V5.3 и предотвращению инъекций. Разработчикам и экспертам по безопасному коду рекомендуется относиться к этому разделу так, как будто для предотвращения инъекций всем элементам требуется значение L1.

#	Описание	L1	L2	L3	CWE
5.1.1	Убедитесь, что приложение обладает защитой от атак с использованием параметров HTTP, особенно если среда приложения не делает различий в источнике параметров запроса (GET, POST, файлы cookie, заголовки или переменные среды).	✓	✓	✓	235
5.1.2	Убедитесь, что фреймворки защищают от атак массового присвоения параметров или что приложение имеет контрмеры для защиты от небезопасного присвоения параметров, такие как пометка полей закрытыми или аналогичными. (C5)	✓	✓	✓	915
5.1.3	Убедитесь, что все входные данные (поля HTML-формы, запросы REST, параметры URL, HTTP-заголовки, файлы cookie, пакетные файлы, RSS-каналы и т.д.) Проверены с помощью положительной проверки (списки разрешений). (C5)	✓	✓	✓	20

5.1.4	Убедитесь, что структурированные данные строго типизированы и проверены в соответствии с определенной схемой, включая разрешенные символы, длину и шаблон (например, номера кредитных карт или телефонов или проверка соответствия двух связанных полей, например, проверка соответствия пригорода и почтового индекса/почтового индекса). (C5)	✓	✓	✓	20
5.1.5	Убедитесь, что для перенаправления и пересылки URL-адресов разрешены только пункты назначения, которые указаны в списке разрешенных, или отображается предупреждение при перенаправлении на потенциально ненадежный контент.	✓	✓	✓	601

V5.2 Требования к санитарной обработке и упаковке в песочницу

#	Описание	L 1	L 2	L 3	CWE
5.2.1	Убедитесь, что все ненадежные HTML-данные, введенные редакторами WYSIWYG или аналогичными, правильно обработаны с помощью библиотеки средств очистки HTML или функции платформы. (C5)	✓	✓	✓	116
5.2.2	Убедитесь, что неструктурированные данные очищены для обеспечения соблюдения мер безопасности, таких как допустимые символы и длина.	✓	✓	✓	138
5.2.3	Убедитесь, что приложение очищает вводимые пользователем данные перед передачей в почтовые системы для защиты от внедрения SMTP или IMAP.	✓	✓	✓	147
5.2.4	Убедитесь, что приложение избегает использования функции eval() или других функций выполнения динамического кода. Там, где альтернативы нет, любой вводимый пользователем ввод должен быть очищен или изолирован перед выполнением.	✓	✓	✓	95
5.2.5	Убедитесь, что приложение защищает от атак с внедрением шаблонов, убедившись, что любой вводимый пользователем ввод очищен или изолирован.	✓	✓	✓	94
5.2.6	Убедитесь, что приложение защищает от атак SSRF путем проверки или очистки ненадежных данных или метаданных файлов HTTP, таких как имена файлов и поля ввода URL-адресов, и использует разрешенные списки протоколов, доменов, путей и портов.	✓	✓	✓	918
5.2.7	Убедитесь, что приложение очищает, отключает или помещает в безопасные среды предоставляемое пользователем содержимое сценариев масштабируемой векторной графики (SVG), особенно если они относятся к XSS, полученным в результате встроенных сценариев, и к внешнему объекту.	✓	✓	✓	159
5.2.8	Убедитесь, что приложение очищает, отключает или помещает в безопасные среды предоставленное пользователем содержимое на языке сценариев или шаблонов выражений, например таблицы стилей Markdown, CSS или XSL, BBCode или аналогичные.	✓	✓	✓	94

V5.3 Требования к кодированию вывода и предотвращению инъекций

Выходная кодировка, расположенная близко или рядом с используемым интерпретатором, имеет решающее значение для безопасности любого приложения. Как правило, кодировка вывода не сохраняется, но используется для обеспечения безопасности вывода в соответствующем контексте вывода для немедленного использования. Неспособность вывести кодирование приведет к небезопасному, инъекционному и небезопасному применению.

#	Описание	L 1	L 2	L 3	CWE
5.3.1	Убедитесь, что кодировка вывода соответствует интерпретатору и требуемому контексту. Например, используйте кодеры специально для значений HTML, атрибутов HTML, JavaScript, параметров URL, заголовков HTTP, SMTP и других, необходимых для контекста, особенно из ненадежных входных данных (например, имена с Юникодом или апострофами, такие как ねこ и л и O'X a 04). (	✓	✓	✓	116

5.3.2	Убедитесь, что кодировка вывода сохраняет выбранный пользователем набор символов и языковой стандарт, чтобы любая точка символа Юникода была допустимой и безопасно обрабатывалась. (C4)	✓	✓	✓	176
5.3.3	Убедитесь, что контекстно-зависимый, предпочтительно автоматический - или, в худшем случае, ручной - выход, экранирующий, защищает от отраженных, сохраненных и основанных на DOM XSS. (C4)	✓	✓	✓	79
5.3.4	Убедитесь, что выбор данных или запросы к базе данных (например, SQL, HQL, ORM, NoSQL) используют параметризованные запросы, ORM, структуры сущностей или иным образом защищены от атак внедрения базы данных. (C3)	✓	✓	✓	89
5.3.5	Убедитесь, что там, где отсутствуют параметризованные или более безопасные механизмы, для защиты от инъекционных атак используется контекстно-зависимая кодировка вывода, например, использование экранирования SQL для защиты от инъекции SQL. (C3, C4)	✓	✓	✓	89
5.3.6	Убедитесь, что приложение защищает от атак с внедрением JavaScript или JSON, в том числе для атак eval, удаленных включений JavaScript, обхода политики безопасности контента (CSP), DOM XSS и оценки выражений JavaScript. (C4)	✓	✓	✓	830
5.3.7	Убедитесь, что приложение защищает от уязвимостей внедрения LDAP или что реализованы специальные средства безопасности для предотвращения внедрения LDAP. (C4)	✓	✓	✓	90
5.3.8	Убедитесь, что приложение защищает от внедрения команд операционной системы и что в вызовах операционной системы используются параметризованные запросы операционной системы или используется контекстная кодировка вывода командной строки. (C4)	✓	✓	✓	78
5.3.9	Убедитесь, что приложение защищает от атак локального включения файлов (LFI) или удаленного включения файлов (RFI).	✓	✓	✓	829
5.3.10	Убедитесь, что приложение защищает от инъекций XPath или атак с использованием XML. (C4)	✓	✓	✓	643

Примечание.Использование параметризованных запросов или экранирования SQL не всегда достаточно; имена таблиц и столбцов, УПОРЯДОЧЕННЫЕ и т. Д., Не могут быть экранированы. Включение сбежавших данных, предоставленных пользователем, в эти поля приводит к ошибочным запросам или внедрению SQL.

Примечание: Формат SVG явно допускает сценарий ECMA почти во всех контекстах, поэтому может оказаться невозможным полностью заблокировать все векторы SVG XSS. Если требуется загрузка SVG, мы настоятельно рекомендуем либо предоставлять эти загруженные файлы в виде текста/обычного текста, либо использовать отдельный домен содержимого, предоставленный пользователем, чтобы предотвратить передачу приложения успешной XSS.

V5.4 Требования к памяти, строке и неуправляемому коду

Следующие требования будут применяться только в том случае, если приложение использует системный язык или неуправляемый код.

#	Описание	L1	L2	L3	CWE

5.4.1	Убедитесь, что приложение использует безопасную для памяти строку, более безопасную копию памяти и арифметику указателей для обнаружения или предотвращения переполнения стека, буфера или кучи.		✓	✓	120
5.4.2	Убедитесь, что строки формата не принимают потенциально враждебные входные данные и являются постоянными.		✓	✓	134
5.4.3	Убедитесь, что методы проверки знака, диапазона и ввода используются для предотвращения переполнения целых чисел.		✓	✓	190

V5.5 Требования к Предотвращению десериализации

#	Описание	L1	L2	L3	CWE
5.5.1	Убедитесь, что сериализованные объекты используют проверку целостности или зашифрованы для предотвращения создания враждебных объектов или взлома данных. (C5)	✓	✓	✓	502
5.5.2	Убедитесь, что приложение правильно ограничивает синтаксические анализаторы XML, чтобы использовать только максимально ограничительную конфигурацию, и убедитесь, что небезопасные функции, такие как разрешение внешних объектов, отключены для предотвращения атак внешних объектов XML (XXE).	✓	✓	✓	611
5.5.3	Убедитесь, что десериализация ненадежных данных предотвращена или защищена как в пользовательском коде, так и в библиотеках сторонних производителей (таких как синтаксические анализаторы JSON, XML и YAML).	✓	✓	✓	502
5.5.4	Убедитесь, что при анализе JSON в браузерах или бэкэндах на основе JavaScript для анализа документа JSON используется JSON.parse. Не используйте eval() для анализа JSON.	✓	✓	✓	95

Ссылки

Для получения дополнительной информации см. также:

- Руководство по тестированию OWASP 4.0: Проверка ввода
- Шпаргалка OWASP: Проверка входных данных
- Руководство по тестированию OWASP 4.0: Тестирование на загрязнение параметров HTTP
- Шпаргалка по внедрению OWASP LDAP
- Руководство по тестированию OWASP 4.0: Тестирование на стороне клиента
- Шпаргалка по предотвращению межсайтовых сценариев OWASP
- Шпаргалка по предотвращению межсайтовых сценариев на основе OWASP DOM на основе OWASP
- Проект кодирования OWASP Java
- Шпаргалка по предотвращению массовых назначений OWASP
- DOMPurify - Библиотека очистки HTML на стороне клиента
- Шпаргалка по предотвращению внешних сущностей XML (XXE)

Для получения дополнительной информации об автоматическом экранировании, пожалуйста, смотрите:

- Сокращение XSS за счет автоматического контекстно-зависимого экранирования в системах шаблонов
- AngularJS Строгое контекстное экранирование
- AngularJS привязать
- Угловая дезинфекция
- Безопасность Углового Шаблона
- Реагируй на побег
- Неправильно управляемая модификация Динамически определяемых атрибутов объектов

Для получения дополнительной информации о десериализации см.:

- Шпаргалка по десериализации OWASP
- Руководство по десериализации OWASP ненадежных данных

V6: Сохраненные Требования К Проверке Криптографии

Цель Управления

Убедитесь, что проверенное приложение удовлетворяет следующим требованиям высокого уровня:

- Все криптографические модули выходят из строя безопасным образом, и ошибки обрабатываются правильно.
- Используется подходящий генератор случайных чисел.
- Доступ к ключам надежно управляется.

V6.1 Классификация Данных

Наиболее важным активом являются данные, обрабатываемые, хранимые или передаваемые приложением. Всегда выполняйте оценку влияния на конфиденциальность, чтобы правильно классифицировать потребности в защите любых сохраненных данных.

#	Описание	L1	L2	L3	CWE
6.1.1	Убедитесь, что регулируемые личные данные хранятся в зашифрованном виде во время отдыха, такие как Информация, позволяющая установить личность (PII), конфиденциальная личная информация или данные, которые, по оценкам, могут подпадать под действие GDPR EC.		✓	✓	311
6.1.2	Убедитесь, что регулируемые медицинские данные хранятся в зашифрованном виде в состоянии покоя, такие как медицинские записи, сведения о медицинском устройстве или обезличенные записи исследований.		✓	✓	311
6.1.3	Убедитесь, что регулируемые финансовые данные хранятся в зашифрованном виде во время отдыха, такие как финансовые счета, данные о дефолтах или кредитной истории, налоговые отчеты, история платежей, бенефициары или обезличенные данные о рынке или исследованиях.		✓	✓	311

V6.2 Алгоритмы

Последние достижения в области криптографии означают, что ранее безопасные алгоритмы и длины ключей больше не являются безопасными или достаточными для защиты данных. Следовательно, должна быть возможность изменять алгоритмы.

Хотя этот раздел нелегко проверить на проникновение, разработчикам следует считать весь этот раздел обязательным, даже если в большинстве пунктов отсутствует L1.

#	Описание	L1	L2	L3	CWE
6.2.1	Убедитесь, что все криптографические модули надежно выходят из строя, а ошибки обрабатываются таким образом, чтобы не допускать повторных атак Oracle.	✓	✓	✓	310
6.2.2	Убедитесь, что используются проверенные в отрасли или одобренные правительством криптографические алгоритмы, режимы и библиотеки, а не специально закодированная криптография. (C8)		✓	✓	327
6.2.3	Убедитесь, что вектор инициализации шифрования, конфигурация шифра и режимы блокировки настроены надежно, используя последние рекомендации.		✓	✓	326

6.2.4	Убедитесь, что случайные числа, алгоритмы шифрования и хеширования, длины ключей, раунды, шифры или режимы могут быть перенастроены, обновлены или заменены в любое время для защиты от криптографических взломов. (C8)		✓	✓	326
6.2.5	Убедитесь что известно небезопасный режимы блока (т. е. ЕЦБ, и т. д.), режимы заполнения (т. е. формат PKCS#1 версии v1.5 и т. д.), шифры с малыми размерами блока (т. е. тройной des, Blowfish и др.), и слабые алгоритмы хеширования (например, MD5 и SHA1 и т. д.) не использовать, если это не требуется для обратной совместимости.		✓	✓	326
6.2.6	Убедитесь, что номера, векторы инициализации и другие одноразовые числа не должны использоваться более одного раза с данным ключом шифрования. Метод генерации должен соответствовать используемому алгоритму.		✓	✓	326
6.2.7	Убедитесь, что зашифрованные данные аутентифицируются с помощью подписей, режимов шифрования с проверкой подлинности или HMAC, чтобы убедиться, что зашифрованный текст не изменен неавторизованной стороной.			✓	326
6.2.8	Убедитесь, что все криптографические операции выполняются постоянно, без операций "короткого замыкания" в сравнениях, вычислениях или возвратах, чтобы избежать утечки информации.			✓	385

V6.3 Случайные Величины

Истинное генерирование псевдослучайных чисел (PRNG) невероятно трудно исправить. Как правило, хорошие источники энтропии в системе быстро истощаются, если будут использоваться слишком часто, но источники с меньшей случайностью могут привести к предсказуемым ключам и секретам.

#	Описание	L1	L2	L3	CWE
6.3.1	Убедитесь, что все случайные числа, случайные имена файлов, случайные идентификаторы GUID и случайные строки генерируются с помощью одобренного криптографическим модулем генератора криптографически безопасных случайных чисел, когда эти случайные значения предназначены для того, чтобы злоумышленник не мог угадать их.		✓	✓	338
6.3.2	Убедитесь, что случайные идентификаторы GUID создаются с помощью алгоритма GUID v4 и криптографически защищенного генератора псевдослучайных чисел (CSPRNG). Идентификаторы GUID, созданные с помощью других генераторов псевдослучайных чисел, могут быть предсказуемыми.		✓	✓	338
6.3.3	Убедитесь, что случайные числа создаются с надлежащей энтропией, даже когда приложение находится под большой нагрузкой, или что приложение изящно ухудшается в таких обстоятельствах.			✓	338

V6.4 Секретное управление

Хотя этот раздел нелегко проверить на проникновение, разработчикам следует считать весь этот раздел обязательным, даже если в большинстве пунктов отсутствует L1.

#	Описание	L1	L2	L3	CWE

6.4.1	Убедитесь, что решение для управления секретами, такое как хранилище ключей, используется для безопасного создания, хранения, контроля доступа к секретам и их уничтожения. (C8)	✓	✓	798
6.4.2	Убедитесь, что ключевой материал не предоставляется приложению, а вместо этого использует изолированный модуль безопасности, такой как хранилище для криптографических операций. (C8)	✓	✓	320

Ссылки

Для получения дополнительной информации см. также:

- Руководство по тестированию OWASP 4.0: Тестирование на слабую криптографию
- Шпаргалка OWASP: Криптографическое хранилище
- FIPS 140-2

V7: Требования к обработке ошибок и проверке ведения журнала

Цель Управления

Основной целью обработки ошибок и ведения журнала является предоставление полезной информации для пользователей, администраторов и групп реагирования на инциденты. Цель состоит не в том, чтобы создать огромное количество журналов, а в том, чтобы создавать журналы высокого качества, с большим количеством сигналов, чем отброшенный шум.

Высококачественные журналы часто содержат конфиденциальные данные и должны быть защищены в соответствии с местными законами или директивами о конфиденциальности данных. Это должно включать:

- Не собирать и не регистрировать конфиденциальную информацию, если это специально не требуется.
- Обеспечение безопасной и защищенной обработки всей регистрируемой информации в соответствии с ее классификацией данных.
- Убедитесь, что журналы хранятся не вечно, а имеют абсолютный срок службы, который как можно короче.

Если журналы содержат личные или конфиденциальные данные, определение которых варьируется от страны к стране, журналы становятся одной из наиболее конфиденциальных сведений, хранящихся приложением, и, таким образом, сами по себе очень привлекательны для злоумышленников.

Также важно убедиться, что приложение безопасно выходит из строя и что ошибки не раскрывают ненужную информацию.

V7.1 Требования к содержимому журнала

Регистрация конфиденциальной информации опасна - журналы сами становятся секретными, что означает, что они должны быть зашифрованы, подпадать под действие политик хранения и должны раскрываться в аудитах безопасности. Убедитесь, что в журналах хранится только необходимая информация и, конечно, никаких платежей, учетных данных (включая токены сеанса), конфиденциальной или идентифицирующей личность информации.

V7.1 охватывает Топ-10 OWASP 2017:A10. Как 2017:A10 и этот раздел не поддаются тестированию на проникновение, это важно для:

- Разработчикам для обеспечения полного соответствия этому разделу, как если бы все элементы были помечены как L1
- Тестеры на проникновение для проверки полного соответствия всех элементов версии 7.1 с помощью интервью, скриншотов или утверждения

#	Описание	L1	L2	L3	CWE
7.1.1	Убедитесь, что приложение не регистрирует учетные данные или платежные реквизиты. Маркеры сеанса должны храниться в журналах только в необратимой хэшированной форме. (C9, C10)	✓	✓	✓	532
7.1.2	Убедитесь, что приложение не регистрирует другие конфиденциальные данные, как это определено в местных законах о конфиденциальности или соответствующей политике безопасности. (C9)	✓	✓	✓	532
7.1.3	Убедитесь, что приложение регистрирует соответствующие события безопасности, включая успешные и неудачные события проверки		✓	✓	778

подлинности, сбои управления доступом, сбои десериализации и сбои проверки ввода. (C5, C7)

7.1.4	Убедитесь, что каждое событие журнала содержит необходимую информацию, которая позволила бы подробно изучить временную шкалу, когда происходит событие. (C9)		✓	✓	778

V7.2 Требования К Обработке журналов

Своевременное ведение журнала имеет решающее значение для событий аудита, сортировки и эскалации. Убедитесь, что журналы приложения понятны и их можно легко отслеживать и анализировать локально или отправлять в удаленную систему мониторинга.

V7.2 охватывает Топ-10 OWASP 2017:A10. Как 2017:A10 и этот раздел не поддаются тестированию на проникновение, это важно для:

- Разработчикам для обеспечения полного соответствия этому разделу, как если бы все элементы были помечены как L1
- Тестеры на проникновение для проверки полного соответствия всех элементов версии 7.2 с помощью интервью, скриншотов или утверждений

#	Описание	L1	L2	L3	CWE
7.2.1	Убедитесь, что все решения по проверке подлинности зарегистрированы без сохранения конфиденциальных токенов сеанса или паролей. Это должно включать запросы с соответствующими метаданными, необходимыми для расследований в области безопасности.		✓	✓	778
7.2.2	Убедитесь, что все решения по контролю доступа могут быть зарегистрированы, а все неудачные решения зарегистрированы. Это должно включать запросы с соответствующими метаданными, необходимыми для расследований в области безопасности.		✓	✓	285

Требования к защите журналов V7.3

Журналы, которые могут быть тривиально изменены или удалены, бесполезны для расследований и судебных преследований. Раскрытие журналов может раскрыть внутреннюю информацию о приложении или содержащихся в нем данных. Необходимо соблюдать осторожность при защите журналов от несанкционированного раскрытия, изменения или удаления.

#	Описание	L1	L2	L3	CWE
7.3.1	Убедитесь, что приложение соответствующим образом кодирует предоставленные пользователем данные, чтобы предотвратить ввод журнала. (C9)		✓	✓	117
7.3.2	Убедитесь, что все события защищены от внедрения при просмотре в программном обеспечении для просмотра журналов. (C9)		✓	✓	117
7.3.3	Убедитесь, что журналы безопасности защищены от несанкционированного доступа и изменения. (C9)		✓	✓	200
7.3.4	Убедитесь, что источники времени синхронизированы с правильным временем и часовым поясом. Настоятельно рекомендуется регистрироваться только в UTC, если системы являются глобальными для оказания помощи в криминалистическом анализе после инцидента. (C9)		✓	✓	

Примечание: Кодирование журнала (7.3.1) сложно протестировать и просмотреть с помощью автоматизированных динамических инструментов и тестов на проникновение, но архитекторы, разработчики и рецензенты исходного кода должны учитывать это требование L1.

V7.4 Обработка ошибок

Цель обработки ошибок состоит в том, чтобы позволить приложению предоставлять события, связанные с безопасностью, для мониторинга, сортировки и эскалации. Цель состоит не в том, чтобы создавать журналы. При регистрации событий, связанных с безопасностью, убедитесь, что в журнале есть цель и что его можно отличить с помощью SIEM или аналитического программного обеспечения.

#	Описание	L1	L2	L3	CWE
7.4.1	Убедитесь, что при возникновении непредвиденной ошибки или ошибки, связанной с безопасностью, отображается общее сообщение, потенциально с уникальным идентификатором, который персонал службы поддержки может использовать для расследования. (C10)	✓	✓	✓	210
7.4.2	Убедитесь, что обработка исключений (или функциональный эквивалент) используется во всей базе кода для учета ожидаемых и неожиданных условий ошибок. (C10)		✓	✓	544
7.4.3	Убедитесь, что определен обработчик ошибок "последнего средства", который будет перехватывать все необработанные исключения. (C10)		✓	✓	431

Примечание. Некоторые языки, такие как Swift и Go - и в соответствии с общепринятой практикой проектирования - многие функциональные языки, не поддерживают исключения или обработчики событий в крайнем случае. В этом случае архитекторы и разработчики должны использовать удобный шаблон, язык или платформу, чтобы гарантировать, что приложения могут безопасно обрабатывать исключительные, неожиданные или связанные с безопасностью события.

Ссылки

Для получения дополнительной информации см. также:

- Содержание Руководства по тестированию OWASP 4.0: Тестирование на обработку ошибок
- Раздел Шпаргалки проверки подлинности OWASP о сообщениях об ошибках

V8: Требования К Проверке Защиты Данных

Цель Контроля

Существует три ключевых элемента надежной защиты данных: Конфиденциальность, Целостность и Доступность (ЦРУ). Этот стандарт предполагает, что защита данных обеспечивается в надежной системе, такой как сервер, которая была защищена и имеет достаточную защиту.

Приложения должны предполагать, что все пользовательские устройства каким-то образом скомпрометированы. Если приложение передает или хранит конфиденциальную информацию на небезопасных устройствах, таких как общие компьютеры, телефоны и планшеты, приложение несет ответственность за обеспечение того, чтобы данные, хранящиеся на этих устройствах, были зашифрованы и не могли быть легко незаконно получены, изменены или раскрыты.

Убедитесь, что проверенное приложение удовлетворяет следующим требованиям к защите данных высокого уровня:

- Конфиденциальность: Данные должны быть защищены от несанкционированного наблюдения или разглашения как при передаче, так и при хранении.
- Целостность: Данные должны быть защищены от злонамеренного создания, изменения или удаления неавторизованными злоумышленниками.
- Доступность: Данные должны быть доступны авторизованным пользователям по мере необходимости.

V8.1 Общая защита Данных

#	Описание	L1	L2	L3	CWE
8.1.1	Убедитесь, что приложение защищает конфиденциальные данные от кэширования в серверных компонентах, таких как балансировщики нагрузки и кэш приложений.		✓	✓	524
8.1.2	Убедитесь, что все кэшированные или временные копии конфиденциальных данных, хранящиеся на сервере, защищены от несанкционированного доступа или удалены/признаны недействительными после того, как авторизованный пользователь получит доступ к конфиденциальным данным.		✓	✓	524
8.1.3	Убедитесь, что приложение минимизирует количество параметров в запросе, таких как скрытые поля, переменные Ajax, файлы cookie и значения заголовков.		✓	✓	233
8.1.4	Убедитесь, что приложение может обнаруживать и предупреждать об аномальном количестве запросов, например, по IP-адресу, пользователю, в общей сложности за час или день или все, что имеет смысл для приложения.		✓	✓	770
8.1.5	Убедитесь, что выполняются регулярные резервные копии важных данных и что выполняется тестовое восстановление данных.			✓	19
8.1.6	Убедитесь, что резервные копии хранятся надежно, чтобы предотвратить кражу или повреждение данных.			✓	19

V8.2 Защита данных на стороне клиента

#	Описание	L1	L2	L3	CWE

8.2.1	Убедитесь, что приложение устанавливает достаточные заголовки для защиты от кэширования, чтобы конфиденциальные данные не кэшировались в современных браузерах.	✓	✓	✓	525
8.2.2	Убедитесь, что данные, хранящиеся в хранилище браузера (например, локальное хранилище HTML5, хранилище сеансов, индексируемая база данных или файлы cookie), не содержат конфиденциальных данных или PII.	✓	✓	✓	922
8.2.3	Убедитесь, что аутентифицированные данные удалены из клиентского хранилища, такого как домен браузера, после завершения работы клиента или сеанса.	✓	✓	✓	922

V8.3 Конфиденциальные Личные Данные

Этот раздел помогает защитить конфиденциальные данные от создания, чтения, обновления или удаления без авторизации, особенно в массовых количествах.

Соблюдение положений настоящего раздела подразумевает соблюдение требований V4 По контролю доступа, в частности V4.2. Например, для защиты от несанкционированного обновления или раскрытия конфиденциальной личной информации требуется соблюдение требований V4.2.1. Пожалуйста, соблюдайте положения настоящего раздела и V4 для полного охвата.

Примечание. Правила и законы о конфиденциальности, такие как Австралийские принципы конфиденциальности APP-11 или GDPR, напрямую влияют на то, как приложения должны подходить к хранению, использованию и передаче конфиденциальной личной информации. Это варьируется от суровых наказаний до простых советов. Пожалуйста, ознакомьтесь с местными законами и правилами, а также, по мере необходимости, проконсультируйтесь с квалифицированным специалистом по вопросам конфиденциальности или адвокатом.

#	Описание	L1	L2	L3	CWE
8.3.1	Убедитесь, что конфиденциальные данные отправляются на сервер в теле или заголовках HTTP-сообщения и что параметры строки запроса из любой команды HTTP не содержат конфиденциальных данных.	✓	✓	✓	319
8.3.2	Убедитесь, что у пользователей есть способ удалить или экспортировать свои данные по требованию.	✓	✓	✓	212
8.3.3	Убедитесь, что пользователям предоставлена четкая формулировка относительно сбора и использования предоставленной личной информации и что пользователи дали согласие на использование этих данных, прежде чем они будут использованы каким-либо образом.	✓	✓	✓	285
8.3.4	Убедитесь, что все конфиденциальные данные, созданные и обработанные приложением, были идентифицированы, и убедитесь, что действует политика обращения с конфиденциальными данными. (C8)	✓	✓	✓	200
8.3.5	Убедитесь, что доступ к конфиденциальным данным проверяется (без регистрации самих конфиденциальных данных), если данные собираются в соответствии с соответствующими директивами по защите данных или когда требуется регистрация доступа.		✓	✓	532
8.3.6	Убедитесь, что конфиденциальная информация, содержащаяся в памяти, перезаписывается, как только больше не требуется для смягчения атак сброса памяти, используя нули или случайные данные.		✓	✓	226
8.3.7	Убедитесь, что конфиденциальная или частная информация, которая должна быть зашифрована, зашифрована с использованием		✓	✓	327

	утвержденных алгоритмов, обеспечивающих конфиденциальность и целостность. (C8)			
8.3.8	Убедитесь, что конфиденциальная личная информация подлежит классификации хранения данных, например, что старые или устаревшие данные удаляются автоматически, по расписанию или в зависимости от ситуации.	✓	✓	285

При рассмотрении вопроса о защите данных основное внимание должно уделяться массовому извлечению, изменению или чрезмерному использованию. Например, многие системы социальных сетей позволяют пользователям добавлять только 100 новых друзей в день, но из какой системы поступали эти запросы, не важно. Банковская платформа может пожелать блокировать более 5 транзакций в час, переводя более 1000 евро средств внешним учреждениям. Требования к каждой системе, скорее всего, будут очень разными, поэтому при принятии решения о "ненормальности" необходимо учитывать модель угроз и бизнес-риски. Важными критериями являются способность обнаруживать, сдерживать или, предпочтительно, блокировать такие аномальные массовые действия.

Ссылки

Для получения дополнительной информации см. также:

- Рассмотрите возможность использования веб-сайта заголовков безопасности для проверки заголовков безопасности и защиты от кэширования
- Проект защищенных заголовков OWASP
- Проект Рисков конфиденциальности OWASP
- Шпаргалка по защите конфиденциальности пользователей OWASP
- Обзор Общих правил защиты данных Европейского Союза (GDPR)
- Супервизор по Защите Данных Европейского Союза - Инженерная сеть Конфиденциальности в Интернете

V9: Требования к проверке Связи

Цель Контроля

Убедитесь, что проверенное приложение удовлетворяет следующим требованиям высокого уровня:

- Всегда используется протокол TLS или надежное шифрование, независимо от чувствительности передаваемых данных
- Самые последние, ведущие рекомендации по настройке используются для включения и упорядочения предпочтительных алгоритмов и шифров
- Слабые или скоро устаревшие алгоритмы и шифры заказываются в качестве последнего средства
- Устаревшие или известные небезопасные алгоритмы и шифры отключены.

Ведущие отраслевые консультации по вопросам безопасной конфигурации TLS часто меняются, часто из-за катастрофических сбоев в существующих алгоритмах и шифрах. Всегда используйте самые последние версии средств проверки конфигурации TLS (например, SSLyze или другие сканеры TLS) для настройки предпочтительного порядка и выбора алгоритма. Конфигурацию следует периодически проверять, чтобы убедиться, что конфигурация безопасной связи всегда присутствует и эффективна.

V9.1 Требования к безопасности Связи с Клиентами

Все клиентские коммуникации должны осуществляться только по зашифрованным каналам связи. В частности, использование TLS 1.2 или более поздней версии, по сути, все, что требуется современным браузерам и поисковым системам. Конфигурацию следует регулярно пересматривать с помощью интерактивных инструментов, чтобы убедиться, что налицо последние передовые методы.

#	Описание	L1	L2	L3	CWE
9.1.1	Убедитесь, что защищённый протокол TLS используется для всех клиентских подключений и не переходит на небезопасные или незашифрованные протоколы. (C8)	✓	✓	✓	319
9.1.2	Проверьте с помощью онлайновых или современных средств тестирования TLS, что включены только надежные алгоритмы, шифры и протоколы, при этом предпочтительными являются самые надежные алгоритмы и шифры.	✓	✓	✓	326
9.1.3	Убедитесь, что старые версии протоколов SSL и TLS, алгоритмов, шифров и конфигурации отключены, такие как SSLv2, SSLv3 или TLS 1.0 и TLS 1.1. Предпочтительным набором шифров должна быть последняя версия TLS.	✓	✓	✓	326

V9.2 Требования к Безопасности Связи с Сервером

Связь с сервером-это нечто большее, чем просто HTTP. Должны быть установлены безопасные соединения с другими системами и с ними, такими как системы мониторинга, средства управления, удаленный доступ и ssh, промежуточное программное обеспечение, база данных, мэйнфреймы, партнерские системы или системы с внешними источниками. Все они должны быть зашифрованы, чтобы предотвратить "жесткий внешний, тривиально простой перехват внутри".

#	Описание	L1	L2	L3	CWE

9.2.1	Убедитесь, что соединения с сервером и с сервера используют надежные сертификаты TLS. Если используются внутренние или самозаверяющие сертификаты, сервер должен быть настроен таким образом, чтобы доверять только определенным внутренним центрам сертификации и определенным самозаверяющим сертификатам. Все остальные должны быть отвергнуты.	✓	✓	295
9.2.2	Убедитесь, что зашифрованные сообщения, такие как TLS, используются для всех входящих и исходящих подключений, включая порты управления, мониторинг, аутентификацию, API или вызовы веб-служб, базы данных, облачные, бессерверные, мэйнфреймы, внешние и партнерские подключения. Сервер не должен возвращаться к небезопасным или незашифрованным протоколам.	✓	✓	319
9.2.3	Убедитесь, что все зашифрованные соединения с внешними системами, связанные с конфиденциальной информацией или функциями, аутентифицированы.	✓	✓	287
9.2.4	Убедитесь, что включено и настроено надлежащее аннулирование сертификации, например сшивание по протоколу OCSP (Online Certificate Status Protocol).	✓	✓	299
9.2.5	Убедитесь, что в журнале регистрируются сбои подключения серверного протокола TLS.		✓	544

Ссылки

Для получения дополнительной информации см. также:

- Шпаргалка OWASP – TLS
- Направляющая для закрепления с помощью OWASP
- Примечания по “Утвержденным режимам TLS". В прошлом ACB ссылались на американский стандарт FIPS 140-2, но как глобальный стандарт применение стандартов США может быть сложным, противоречивым или запутанным. Лучшим способом достижения соответствия требованиям 9.1.3 было бы ознакомиться с руководствами, такими как TLS на стороне сервера Mozilla, или создать известные хорошие конфигурации, а также использовать известные инструменты оценки TLS, такие как sslyze, различные сканеры уязвимостей или надежные онлайн-службы оценки TLS для получения требуемого уровня безопасности. В целом, мы видим несоответствие для этого раздела, заключающееся в использовании устаревших или небезопасных шифров и алгоритмов, отсутствии полной секретности пересылки, устаревших или небезопасных протоколов SSL, слабых предпочтительных шифров и т. Д.

V10: Требования к проверке Вредоносного Кода

Цель Контроля

Убедитесь, что код удовлетворяет следующим требованиям высокого уровня:

- Вредоносная деятельность обрабатывается надежно и должным образом, чтобы не повлиять на остальную часть приложения.
- Нет бомб замедленного действия или других атак, основанных на времени.
- Не "звоните домой" по вредоносным или несанкционированным адресам.
- Не содержит задних дверей, пасхальных яиц, атак салями, руткитов или несанкционированного кода, которыми может управлять злоумышленник.

Обнаружение вредоносного кода является доказательством негатива, который невозможно полностью проверить. Следует приложить все усилия для обеспечения того, чтобы код не содержал врожденного вредоносного кода или нежелательной функциональности.

V10.1 Контроль целостности Кода

Лучшая защита от вредоносного кода - "доверяй, но проверяй". Введение несанкционированного или вредоносного кода в кодекс часто является уголовным преступлением во многих юрисдикциях. Политика и процедуры должны четко определять санкции в отношении вредоносного кода.

Ведущие разработчики должны регулярно проверять проверки кода, особенно те, которые могут иметь доступ ко времени, вводу-выводу или сетевым функциям.

#	Описание	L1	L2	L3	CWE
10.1.1	Убедитесь, что используется средство анализа кода, которое может обнаруживать потенциально вредоносный код, например функции времени, небезопасные операции с файлами и сетевые подключения.			✓	749

V10.2 Поиск вредоносного кода

Вредоносный код встречается крайне редко и его трудно обнаружить. Ручной пошаговый обзор кода может помочь в поиске логических бомб, но даже самому опытному обозревателю кода будет трудно найти вредоносный код, даже если он знает о его существовании.

Соблюдение этого раздела невозможно без полного доступа к исходному коду, включая библиотеки сторонних производителей.

#	Описание	L1	L2	L3	CWE
10.2.1	Убедитесь, что исходный код приложения и библиотеки сторонних разработчиков не содержат несанкционированных функций домашнего телефона или сбора данных. В тех случаях, когда такая функциональность существует, прежде чем собирать какие-либо данные, получите разрешение пользователя на ее работу.		✓	✓	359
10.2.2	Убедитесь, что приложение не запрашивает ненужных или чрезмерных разрешений для функций или датчиков, связанных с конфиденциальностью, таких как контакты, камеры, микрофоны или местоположение.		✓	✓	272
10.2.3	Убедитесь, что исходный код приложения и библиотеки сторонних, не содержат в себе лазейки, например, жестко или дополнительных недокументированных аккаунты или ключи, код запутывания, без			✓	507

	документов бинарных блобов, руткитов и анти-отладки, неуверенный в себе функции отладки, или иначе устарели, неуверенным в себе, или скрытые функциональные возможности, которые могут быть использованы злонамеренно, если обнаружен.				
10.2.4	Убедитесь, что исходный код приложения и библиотеки сторонних разработчиков не содержат бомб замедленного действия, выполнив поиск функций, связанных с датой и временем.			✓	511
10.2.5	Убедитесь, что исходный код приложения и библиотеки сторонних разработчиков не содержат вредоносного кода, такого как атаки с салями, логические обходы или логические бомбы.			✓	511
10.2.6	Убедитесь, что исходный код приложения и библиотеки сторонних разработчиков не содержат пасхальных яиц или других потенциально нежелательных функций.			✓	507

V10.3 Развернутые Средства Контроля Целостности Приложений

После развертывания приложения вредоносный код все еще может быть вставлен. Приложениям необходимо защитить себя от распространенных атак, таких как выполнение неподписанного кода из ненадежных источников и поглощение поддоменов.

Соблюдение этого раздела, скорее всего, будет оперативным и непрерывным.

#	Описание	L1	L2	L3	CWE
10.3.1	Убедитесь, что, если приложение имеет функцию автоматического обновления клиента или сервера, обновления должны быть получены по защищенным каналам и подписаны цифровой подписью. Код обновления должен подтвердить цифровую подпись обновления перед установкой или выполнением обновления.	✓	✓	✓	16
10.3.2	Убедитесь, что приложение использует средства защиты целостности, такие как подпись кода или целостность под источника. Приложение не должно загружать или выполнять код из ненадежных источников, таких как загрузка включений, модулей, плагинов, кода или библиотек из ненадежных источников или Интернета.	✓	✓	✓	353
10.3.3	Убедитесь, что приложение защищено от поглощения поддоменов, если приложение использует записи DNS или поддомены DNS, такие как устаревшие доменные имена, устаревшие указатели DNS или имена CNAME, устаревшие проекты в репозиториях общедоступных исходных кодов или временные облачные API, функции без сервера или корзины хранения (*autogen-bucket-id.cloud.example.com*) или что-то в этом роде. Защита может включать в себя обеспечение того, чтобы DNS-имена, используемые приложениями, регулярно проверялись на срок действия или изменение.	✓	✓	✓	350

Ссылки

- Враждебное Поглощение Субдоменов, Обнаружение Лабораторий
- Похищение заброшенных поддоменов часть 2. Обнаружение лабораторий

V11: Требования к проверке Бизнес-логики

Цель Управления

Убедитесь, что проверенное приложение удовлетворяет следующим требованиям высокого уровня:

- Поток бизнес-логики является последовательным, обрабатывается по порядку и не может быть обойден.
- Бизнес-логика включает ограничения на обнаружение и предотвращение автоматических атак, таких как непрерывные небольшие переводы средств или добавление миллиона друзей по одному и так далее.
- Потоки бизнес-логики с высокой стоимостью рассмотрели случаи злоупотреблений и злоумышленников и имеют защиту от подделки, подделки, отказа, раскрытия информации и несанкционированных атак.

V11.1 Требования к безопасности Бизнес-логики

Безопасность бизнес-логики настолько индивидуальна для каждого приложения, что ни один контрольный список никогда не будет применяться. Безопасность бизнес - логики должна быть разработана для защиты от возможных внешних угроз-ее нельзя добавить с помощью брандмауэров веб-приложений или защищенных коммуникаций. Мы рекомендуем использовать моделирование угроз во время спринтов проектирования, например, с помощью рога Изобилия OWASP или аналогичных инструментов.

#	Описание	L1	L2	L3	CWE
11.1.1	Убедитесь, что приложение будет обрабатывать потоки бизнес-логики только для одного и того же пользователя в последовательном порядке и без пропуска шагов.	✓	✓	✓	841
11.1.2	Убедитесь, что приложение будет обрабатывать только потоки бизнес-логики, при этом все шаги обрабатываются в реалистичном человеческом времени, т. е. транзакции не передаются слишком быстро.	✓	✓	✓	799
11.1.3	Убедитесь, что приложение имеет соответствующие ограничения для конкретных бизнес-действий или транзакций, которые правильно применяются для каждого пользователя.	✓	✓	✓	770
11.1.4	Убедитесь, что приложение имеет достаточные средства защиты от автоматизации для обнаружения и защиты от утечки данных, чрезмерных запросов бизнес-логики, чрезмерной загрузки файлов или атак типа "отказ в обслуживании".	✓	✓	✓	770
11.1.5	Убедитесь, что приложение имеет ограничения бизнес-логики или проверку для защиты от возможных бизнес-рисков или угроз, выявленных с помощью моделирования угроз или аналогичных методологий.	✓	✓	✓	841
11.1.6	Убедитесь, что приложение не страдает от проблем "Время проверки до времени использования" (TOCTOU) или других расовых условий для конфиденциальных операций.		✓	✓	367
11.1.7	Убедитесь, что приложение отслеживает необычные события или действия с точки зрения бизнес-логики. Например, попытки выполнить действия не по порядку или действия, которые обычный пользователь никогда бы не предпринял. (C9)		✓	✓	754

11.1.8	Убедитесь, что приложение имеет настраиваемое оповещение при обнаружении автоматических атак или необычной активности.	✓	✓	390

Ссылки

Для получения дополнительной информации см. также:

- Руководство по тестированию веб-безопасности OWASP 4.1: Тестирование бизнес-логики
- Защита от автоматизации может быть достигнута многими способами, включая использование OWASP AppSensor и автоматизированных угроз OWASP для веб-приложений
- Датчик приложений OWASP также может помочь в обнаружении атак и реагировании на них.
- OWASP Рог изобилия

V12: Требования к проверке файлов и ресурсов

Цель Управления

Убедитесь, что проверенное приложение удовлетворяет следующим требованиям высокого уровня:

- Ненадежные данные файлов должны обрабатываться соответствующим и безопасным образом.
- Ненадежные данные файлов, полученные из ненадежных источников, хранятся вне корневого каталога веб-сайта и с ограниченными разрешениями.

V12.1 Требования к Загрузке файлов

Хотя zip-бомбы в высшей степени поддаются проверке с использованием методов тестирования на проникновение, они считаются L2 и выше, чтобы стимулировать разработку и разработку с тщательным ручным тестированием и избегать автоматического или неквалифицированного ручного тестирования на проникновение в случае отказа в обслуживании.

#	Описание	L1	L2	L3	CWE
12.1.1	Убедитесь, что приложение не будет принимать большие файлы, которые могут заполнить хранилище или вызвать отказ в обслуживании.	✓	✓	✓	400
12.1.2	Убедитесь, что сжатые файлы проверены на наличие "zip - бомб" - небольших входных файлов, которые будут распаковываться в огромные файлы, тем самым исчерпывая ограничения на хранение файлов.		✓	✓	409
12.1.3	Убедитесь, что квота на размер файла и максимальное количество файлов для каждого пользователя установлены, чтобы один пользователь не мог заполнить хранилище слишком большим количеством файлов или файлами чрезмерного размера.		✓	✓	770

Требования к целостности файлов версии 12.2

#	Описание	L1	L2	L3	CWE
12.2.1	Убедитесь, что файлы, полученные из ненадежных источников, проверяются на соответствие ожидаемому типу на основе содержимого файла.		✓	✓	434

V12.3 Требования к Выполнению файлов

#	Описание	L1	L2	L3	CWE
12.3.1	Убедитесь, что метаданные имени файла, отправленные пользователем, не используются непосредственно системными файловыми системами или файловыми системами платформы и что API URL используется для защиты от обхода пути.	✓	✓	✓	22
12.3.2	Убедитесь, что метаданные имени файла, отправленные пользователем, проверены или проигнорированы, чтобы предотвратить раскрытие, создание, обновление или удаление локальных файлов (LFI).	✓	✓	✓	73
12.3.3	Убедитесь, что метаданные имени файла, отправленные пользователем, проверены или проигнорированы, чтобы	✓	✓	✓	98

	предотвратить раскрытие или выполнение удаленных файлов с помощью атак удаленного включения файлов (RFI) или подделки запросов на стороне сервера (SSRF).				
12.3.4	Убедитесь, что приложение защищает от загрузки файлов с отражением (RFD), проверяя или игнорируя имена файлов, отправленные пользователем, в параметре JSON, JSONP или URL.Заголовок типа содержимого ответа должен быть установлен в текст/обычный, а заголовок расположения содержимого должен иметь фиксированное имя файла.	✓	✓	✓	641
12.3.5	Убедитесь, что метаданные ненадежных файлов не используются напрямую с системным API или библиотеками для защиты от ввода команд ОС.	✓	✓	✓	78
12.3.6	Убедитесь, что приложение не включает и не выполняет функции из ненадежных источников, таких как непроверенные сети распространения контента, библиотеки JavaScript, библиотеки npm узлов или библиотеки DLL на стороне сервера.		✓	✓	829

Требования к хранению файлов версии 12.4

#	Описание	L1	L2	L3	CWE
12.4.1	Убедитесь, что файлы, полученные из ненадежных источников, хранятся вне корневого каталога сети с ограниченными разрешениями, предпочтительно с надежной проверкой.	✓	✓	✓	922
12.4.2	Убедитесь, что файлы, полученные из ненадежных источников, сканируются антивирусными сканерами для предотвращения загрузки известного вредоносного контента.	✓	✓	✓	509

Требования к загрузке файлов V12.5

#	Описание	L1	L2	L3	CWE
12.5.1	Убедитесь, что веб-уровень настроен для обслуживания только файлов с определенными расширениями файлов, чтобы предотвратить непреднамеренную утечку информации и исходного кода. Например, файлы резервных копий (например .bak), временные рабочие файлы (например .swp), сжатые файлы (.zip, .tar.gz, и т.д.) и другие расширения, обычно используемые редакторами, должны быть заблокированы без необходимости.	✓	✓	✓	552
12.5.2	Убедитесь, что прямые запросы к загруженным файлам никогда не будут выполняться в виде содержимого HTML/JavaScript.	✓	✓	✓	434

V12.6 Требования к защите SSRF

#	Описание	L1	L2	L3	CWE
12.6.1	Убедитесь, что на веб-сервере или сервере приложений настроен список разрешенных ресурсов или систем, которым сервер может отправлять запросы или загружать данные/файлы.	✓	✓	✓	918

Ссылки

Для получения дополнительной информации см. также:

- Обработка расширений файлов для конфиденциальной информации

- Отражающий файл, загруженный Ореном Хафифом
- Шпаргалка по управлению JavaScript OWASP От Третьих Лиц

V13: Требования к проверке API и веб-служб

Цель Управления

Убедитесь, что проверенное приложение, использующее доверенные API уровня обслуживания (обычно использующие JSON, XML или GraphQL), имеет:

- Надлежащая аутентификация, управление сеансами и авторизация всех веб-служб.
- Проверка ввода всех параметров, которые переходят с более низкого уровня доверия на более высокий.
- Эффективные средства контроля безопасности для всех типов API, включая облачные и бессерверные API

Пожалуйста, прочитайте эту главу в сочетании со всеми другими главами на этом же уровне; мы больше не дублируем вопросы аутентификации и управления сеансами API.

V13.1 Общие Требования К Проверке Безопасности Веб-Службы

#	Описание	L1	L2	L3	CWE
13.1.1	Убедитесь, что все компоненты приложения используют одни и те же кодировки и средства синтаксического анализа, чтобы избежать атак синтаксического анализа, использующих различные URI или поведение анализа файлов, которые могут использоваться в атаках SSRF и RFI.	✓	✓	✓	116
13.1.2	Убедитесь, что доступ к функциям администрирования и управления ограничен авторизованными администраторами.	✓	✓	✓	419
13.1.3	Убедитесь, что URL-адреса API не содержат конфиденциальной информации, такой как ключ API, токены сеанса и т.д.	✓	✓	✓	598
13.1.4	Убедитесь, что решения об авторизации принимаются как на уровне URI, что обеспечивается программной или декларативной безопасностью на контроллере или маршрутизаторе, так и на уровне ресурсов, что обеспечивается разрешениями на основе моделей.		✓	✓	285
13.1.5	Убедитесь, что запросы, содержащие неожиданные или отсутствующие типы контента, отклоняются с соответствующими заголовками (статус ответа HTTP 406 Неприемлемый или 415 Неподдерживаемый тип носителя).		✓	✓	434

V13.2 Требования к проверке веб-службы RESTful

Проверка схемы JSON находится на стадии разработки проекта стандартизации (см. Ссылки). При рассмотрении возможности использования проверки схемы JSON, что является наилучшей практикой для веб-служб RESTful, рассмотрите возможность использования этих дополнительных стратегий проверки данных в сочетании с проверкой схемы JSON:

- Проверка синтаксического анализа объекта JSON, например, на наличие отсутствующих или дополнительных элементов.
- Проверка значений объектов JSON с использованием стандартных методов проверки входных данных, таких как тип данных, формат данных, длина и т.д.
- и формальная проверка схемы JSON.

Как только стандарт проверки схемы JSON будет формализован, ASVS обновит свои рекомендации в этой области. Внимательно следите за любыми используемыми библиотеками

проверки схемы JSON, так как их необходимо будет регулярно обновлять до тех пор, пока стандарт не будет формализован и ошибки не будут устранены из стандартных реализаций.

#	Описание	L1	L2	L3	CWE
13.2.1	Убедитесь, что включенные методы HTTP RESTful являются допустимым выбором для пользователя или действия, такого как предотвращение использования обычными пользователями УДАЛЕНИЯ или ВКЛЮЧЕНИЯ защищенного API или ресурсов.	✓	✓	✓	650
13.2.2	Убедитесь, что проверка схемы JSON выполнена и проверена, прежде чем принимать входные данные.	✓	✓	✓	20
13.2.3	Убедитесь, что веб-службы RESTful, использующие файлы cookie, защищены от подделки межсайтовых запросов с помощью, по крайней мере, одного или нескольких из следующих способов: шаблон двойной отправки файлов cookie, отсутствие CSRF или проверка заголовка исходного запроса.	✓	✓	✓	352
13.2.4	Убедитесь, что в службах REST есть элементы управления для защиты от чрезмерных вызовов, особенно если API не прошел проверку подлинности.		✓	✓	770
13.2.5	Убедитесь, что службы REST явно проверяют тип входящего содержимого на ожидаемый, например application/xml или application/json.		✓	✓	436
13.2.6	Убедитесь, что заголовки сообщений и полезные данные являются надежными и не изменяются при передаче. Требование строгого шифрования для транспорта (только TLS) может быть достаточным во многих случаях, поскольку оно обеспечивает как конфиденциальность, так и защиту целостности. Цифровые подписи для каждого сообщения могут обеспечить дополнительную гарантию в дополнение к защите транспорта для приложений с высокой степенью безопасности, но влекут за собой дополнительные сложности и риски, которые сопоставляются с преимуществами.		✓	✓	345

V13.3 Требования к проверке веб-службы SOAP

#	Описание	L1	L2	L3	CWE
13.3.1	Убедитесь, что проверка схемы XSD выполнена для обеспечения правильного формирования XML-документа, а затем проверьте каждое поле ввода перед какой-либо обработкой этих данных.	✓	✓	✓	20
13.3.2	Убедитесь, что полезная нагрузка сообщения подписана с помощью WS-Security для обеспечения надежной передачи между клиентом и службой.		✓	✓	345

Примечание: Из-за проблем с атаками XXE на DTD, проверка DTD не должна использоваться, и оценка DTD платформы отключена в соответствии с требованиями, изложенными в конфигурации V14.

V13.4 GraphQL и другие требования к безопасности уровня данных веб-службы

#	Описание	L1	L2	L3	CWE

13.4.1	Убедитесь, что список разрешенных запросов или комбинация ограничения глубины и ограничения объема используются для предотвращения отказа в обслуживании (DoS) в GraphQL или выражении уровня данных в результате дорогостоящих вложенных запросов. Для более сложных сценариев следует использовать анализ затрат на запросы.	✓	✓	770
13.4.2	Убедитесь, что логика авторизации GraphQL или другого уровня данных должна быть реализована на уровне бизнес-логики, а не на уровне GraphQL.	✓	✓	285

Ссылки

Для получения дополнительной информации см. также:

- OWASP Бессерверный Топ-10
- Бессерверный проект OWASP
- Руководство по тестированию OWASP 4.0: Тестирование управления конфигурацией и развертыванием
- Шпаргалка по подделке межсайтовых запросов OWASP
- Шпаргалка по предотвращению внешних сущностей OWASP XML - Общие рекомендации
- Веб-токены JSON (и подпись)
- Шпаргалка по безопасности REST
- Схема JSON
- Атаки на объекты XML DTD
- Orange Tsai - Новая эра SSRF, Использующая Анализатор URL-адресов В современных языках программирования

V14: Требования к Проверке Конфигурации

Цель Управления

Убедитесь, что проверенное приложение содержит:

- Безопасная, воспроизводимая, автоматизируемая среда сборки.
- Надежное управление библиотеками, зависимостями и конфигурациями сторонних разработчиков, позволяющее приложению не включать устаревшие или небезопасные компоненты.
- Безопасная конфигурация по умолчанию, так что администраторам и пользователям приходится ослаблять защиту по умолчанию.

Конфигурация приложения из коробки должна быть безопасной для работы в Интернете, что означает безопасную конфигурацию из коробки.

Сборка V14.1

Конвейеры сборки являются основой для повторяемой безопасности - каждый раз, когда обнаруживается что-то небезопасное, оно может быть устранено в исходном коде, сценариях сборки или развертывания и автоматически протестировано. Мы настоятельно рекомендуем использовать конвейеры сборки с автоматическими проверками безопасности и зависимостей, которые предупреждают или прерывают сборку, чтобы предотвратить внедрение известных проблем безопасности в производство. Нерегулярное выполнение ручных действий напрямую приводит к ошибкам в системе безопасности, которых можно избежать.

По мере перехода отрасли к модели DevSecOps важно обеспечить постоянную доступность и целостность развертывания и конфигурации для достижения состояния "известного качества". В прошлом, если система была взломана, требовались дни или месяцы, чтобы доказать, что никаких дальнейших вторжений не было. Сегодня, с появлением программно-определяемой инфраструктуры, быстрым развертыванием A/B с нулевым временем простоя и автоматизированными сборками в контейнерах, можно автоматически и непрерывно создавать, укреплять и развертывать "заведомо хорошую" замену для любой скомпрометированной системы.

Если традиционные модели все еще существуют, необходимо выполнить ручные действия по укреплению и резервному копированию этой конфигурации, чтобы можно было быстро заменить скомпрометированные системы на системы с высокой целостностью и бескомпромиссностью и своевременно.

Для соблюдения требований этого раздела требуется автоматизированная система сборки и доступ к сценариям сборки и развертывания.

#	Описание	L1	L2	L3	CWE
14.1.1	Убедитесь, что процессы сборки и развертывания приложений выполняются безопасным и воспроизводимым способом, таким как автоматизация CI / CD, автоматическое управление конфигурацией и сценарии автоматического развертывания.		✓	✓	
14.1.2	Убедитесь, что флаги компилятора настроены для включения всех доступных средств защиты от переполнения буфера и предупреждений, включая рандомизацию стека, предотвращение выполнения данных, а также для прерывания сборки, если обнаружен небезопасный указатель, память, строка форматирования, целое число или строковые операции.		✓	✓	120

14.1.3	Убедитесь, что конфигурация сервера защищена в соответствии с рекомендациями используемого сервера приложений и используемых платформ.		✓	✓	16
14.1.4	Убедитесь, что приложение, конфигурация и все зависимости могут быть повторно развернуты с использованием сценариев автоматического развертывания, созданы на основе документированного и протестированного журнала выполнения в разумные сроки или своевременно восстановлены из резервных копий.		✓	✓	
14.1.5	Убедитесь, что авторизованные администраторы могут проверить целостность всех конфигураций, связанных с безопасностью, для обнаружения несанкционированного доступа.			✓	

V14.2 Зависимость

Управление зависимостями имеет решающее значение для безопасной работы любого приложения любого типа. Неспособность идти в ногу со временем устаревших или небезопасных зависимостей является основной причиной крупнейших и наиболее дорогостоящих атак на сегодняшний день.

Примечание: На уровне 1 соответствие 14.2.1 относится к наблюдениям или обнаружениям клиентских и других библиотек и компонентов, а не к более точному статическому анализу кода во время сборки или анализу зависимостей. При необходимости эти более точные методы можно было бы обнаружить с помощью опроса.

#	Описание	L1	L2	L3	CWE
14.2.1	Убедитесь, что все компоненты обновлены, предпочтительно с помощью средства проверки зависимостей во время сборки или компиляции. (C2)	✓	✓	✓	1026
14.2.2	Убедитесь, что удалены все ненужные функции, документация, образцы, конфигурации, такие как примеры приложений, документация платформы и пользователи по умолчанию или примеры пользователей.	✓	✓	✓	1002
14.2.3	Убедитесь, что если ресурсы приложения, такие как библиотеки JavaScript, CSS или веб-шрифты, размещены извне в Сети доставки контента (CDN) или у внешнего поставщика, Целостность под источника (SRI) используется для проверки целостности ресурса.	✓	✓	✓	829
14.2.4	Убедитесь, что сторонние компоненты поступают из заранее определенных, надежных и постоянно поддерживаемых репозиториев. (C2)		✓	✓	829
14.2.5	Убедитесь, что каталог инвентаризации ведется для всех используемых сторонних библиотек. (C2)		✓	✓	
14.2.6	Убедитесь, что поверхность атаки уменьшена за счет песочницы или инкапсуляции сторонних библиотек, чтобы предоставить приложению только требуемое поведение. (C2)		✓	✓	265

V14.3 Требования К Непреднамеренному Раскрытию Информации о Безопасности

Конфигурации для рабочей среды должны быть защищены от распространенных атак, таких как отладочные консоли, повышают планку для атак на межсайтовые сценарии (XSS) и Удаленное включение файлов (RFI), а также устраняют "уязвимости" для обнаружения тривиальной информации, которые являются нежелательной отличительной чертой многих отчетов о

тестировании на проникновение. Многие из этих проблем редко оцениваются как значительный риск, но они связаны вместе с другими уязвимостями. Если по умолчанию эти проблемы отсутствуют, это поднимет планку, прежде чем большинство атак смогут увенчаться успехом.

#	Описание	L1	L2	L3	CWE
14.3.1	Убедитесь, что сообщения об ошибках веб-сервера или сервера приложений и платформы настроены таким образом, чтобы предоставлять пользователям доступные для действий индивидуальные ответы для устранения любых непреднамеренных раскрытий информации о безопасности.	✓	✓	✓	209
14.3.2	Убедитесь, что в рабочей среде отключены режимы отладки веб-сервера или сервера приложений и платформы приложений, чтобы исключить функции отладки, консоли разработчиков и непреднамеренное раскрытие информации о безопасности.	✓	✓	✓	497
14.3.3	Убедитесь, что заголовки HTTP или любая часть ответа HTTP не содержат подробной информации о версии компонентов системы.	✓	✓	✓	200

Требования к заголовкам безопасности HTTP версии 14.4

#	Описание	L1	L2	L3	CWE
14.4.1	Убедитесь, что каждый HTTP-ответ содержит заголовок типа содержимого. типы содержимого text/*, /+xml и application/xml также должны указывать безопасный набор символов (например, UTF-8, ISO-8859-1).	✓	✓	✓	173
14.4.2	Убедитесь, что все ответы API содержат расположение содержимого: вложение; имя файла=заголовок"api.json" (или другое подходящее имя файла для типа содержимого).	✓	✓	✓	116
14.4.3	Убедитесь, что заголовок ответа Политики безопасности содержимого (CSP) установлен, что помогает смягчить последствия для XSS-атак, таких как уязвимости при внедрении HTML, DOM, JSON и JavaScript.	✓	✓	✓	1021
14.4.4	Убедитесь, что все ответы содержат заголовок X-Content-Type-Options: nosniff.	✓	✓	✓	116
14.4.5	Убедитесь, что заголовок Строгой транспортной безопасности включен во все ответы и для всех поддоменов, таких как Строгая транспортная безопасность: максимальный возраст = 15724800; Включает поддомены.	✓	✓	✓	523
14.4.6	Убедитесь, что включен подходящий заголовок "Политика ссылок", например "без ссылок" или "того же происхождения".	✓	✓	✓	116
14.4.7	Убедитесь, что содержимое веб-приложения по умолчанию не может быть внедрено на стороннем сайте и что встраивание точных ресурсов допускается только в случае необходимости, используя подходящие заголовки ответов Content-Security-Policy: frame-предки и X-Frame-Параметры.	✓	✓	✓	346

V14.5 Проверка требований к заголовку HTTP-запроса

#	Описание	L1	L2	L3	CWE
14.5.1	Убедитесь, что сервер приложений принимает только методы HTTP, используемые приложением/API, включая ПАРАМЕТРЫ перед полетом, и регистрирует/предупреждает о любых запросах, которые недопустимы для контекста приложения.	✓	✓	✓	749
14.5.2	Убедитесь, что предоставленный исходный заголовок не используется для аутентификации или контроля доступа, так как исходный заголовок может быть легко изменен злоумышленником.	✓	✓	✓	346
14.5.3	Убедитесь, что заголовок Управления доступом к ресурсам общего доступа (CORS), разрешающий отправку, использует строгий разрешенный список доверенных доменов и поддоменов для сопоставления и не поддерживает "нулевое" происхождение.	✓	✓	✓	346
14.5.4	Убедитесь, что заголовки HTTP, добавленные доверенным прокси-сервером или устройствами единого входа, такими как токен носителя, аутентифицируются приложением.		✓	✓	306

Ссылки

Для получения дополнительной информации см. также:

- Руководство по тестированию веб-безопасности OWASP 4.1: Тестирование на подделку HTTP-команд
- Добавление диспозиции содержимого в ответы API помогает предотвратить многие атаки, основанные на неправильном понимании типа MIME между клиентом и сервером, а параметр "имя файла" особенно помогает предотвратить Отраженные атаки загрузки файлов.
- Шпаргалка по Политике Безопасности Содержимого
- Использование неправильной конфигурации CORS для биткоинов и вознаграждений
- Руководство по тестированию веб-безопасности OWASP 4.1: Тестирование управления конфигурацией и развертыванием
- Компоненты для песочницы сторонних производителей

Приложение A: Глоссарий

- Рандомизация адресного пространства (ASLR) - метод, затрудняющий использование ошибок повреждения памяти.
- Список разрешенных - список разрешенных данных или операций, например список символов, которым разрешено выполнять проверку ввода.
- Безопасность приложений - безопасность на уровне приложений фокусируется на анализе компонентов, составляющих прикладной уровень эталонной модели взаимодействия открытых систем (модель OSI), а не на анализе, например, базовой операционной системы или подключенных сетей.
- Проверка безопасности приложения - техническая оценка приложения на соответствие OWASP ASVS.
- Отчет о проверке безопасности приложения - отчет, в котором документируются общие результаты и подтверждающий анализ, произведенный верификатором для конкретного приложения.
- Аутентификация - подтверждение заявленной личности пользователя приложения.
- Автоматическая проверка - использование автоматических инструментов (либо инструментов динамического анализа, либо инструментов статического анализа, либо того и другого), которые используют сигнатуры уязвимостей для поиска проблем.
- Тестирование черного ящика - это метод тестирования программного обеспечения, при котором проверяется функциональность приложения без изучения его внутренней структуры или работы.
- Компонент - автономная единица кода со связанными дисковыми и сетевыми интерфейсами, которая взаимодействует с другими компонентами.
- Межсайтовый скриптинг (XSS) - уязвимость системы безопасности, обычно обнаруживаемая в веб-приложениях, позволяющая внедрять клиентские скрипты в контент.
- Криптографический модуль - оборудование, программное обеспечение и / или микропрограммное обеспечение, реализующее криптографические алгоритмы и / или генерирующее криптографические ключи.
- Common Weakness Enumeration (CWE) - список общих недостатков безопасности программного обеспечения, разработанный сообществом. Он служит общим языком, мерой для инструментов безопасности программного обеспечения и базой для выявления слабых мест, смягчения и предотвращения.
- Проверка дизайна - техническая оценка архитектуры безопасности приложения.
- Динамическое тестирование безопасности приложений (DAST) - технологии предназначены для обнаружения условий, указывающих на уязвимость безопасности в приложении в его рабочем состоянии.
- Динамическая проверка - использование автоматизированных инструментов, которые используют сигнатуры уязвимостей для поиска проблем во время выполнения приложения..
- Fast IDentity Online (FIDO) - набор стандартов аутентификации, которые позволяют использовать множество различных методов аутентификации, включая биометрию, доверенные платформенные модули (TPM), токены безопасности USB и т. Д.
- Глобальный уникальный идентификатор (GUID) - уникальный ссылочный номер, используемый в качестве идентификатора в программном обеспечении.
- Протокол передачи гипертекста (HTTPS) - протокол приложения для распределенных, совместных, гипермедийных информационных систем. Это основа передачи данных во всемирной паутине.

- Жестко запрограммированные ключи - криптографические ключи, которые хранятся в файловой системе, будь то код, комментарии или файлы.
- Аппаратный модуль безопасности (HSM) - аппаратный компонент, который может хранить криптографические ключи и другие секреты в защищенном виде.
- Hibernate Query Language (HQL) - язык запросов, внешне похожий на SQL, используемый библиотекой Hibernate ORM.
- Проверка ввода - канонизация и проверка ввода ненадежных пользователей.
- Вредоносный код - код, введенный в приложение во время его разработки без ведома владельца приложения, который обходит предполагаемую политику безопасности приложения. Не то же самое, что вредоносное ПО, такое как вирус или червь!
- Вредоносное ПО - исполняемый код, который вводится в приложение во время выполнения без ведома пользователя или администратора приложения.
- Открытый проект безопасности веб-приложений (OWASP) - Открытый проект безопасности веб-приложений (OWASP) - это всемирное свободное и открытое сообщество, ориентированное на повышение безопасности прикладного программного обеспечения. Наша миссия - сделать безопасность приложений «видимой», чтобы люди и организации могли принимать обоснованные решения о рисках безопасности приложений. Видеть:
- Одноразовый пароль (OTP) - пароль, который создается уникальным образом для однократного использования.
- Объектно-реляционное сопоставление (ORM) - система, позволяющая ссылаться на реляционную / табличную базу данных и запрашивать ее внутри прикладной программы с использованием объектной модели, совместимой с приложением.
- Функция получения ключа на основе пароля 2 (PBKDF2) - специальный односторонний алгоритм, используемый для создания надежного криптографического ключа из входящего текста (например, пароля) и дополнительного случайного значения соли и, следовательно, может использоваться, чтобы усложнить взлом пароль в автономном режиме, если полученное значение сохраняется вместо исходного пароля.
- Информация, позволяющая установить личность (PII) - это информация, которая может использоваться сама по себе или вместе с другой информацией для идентификации, установления контакта или определения местонахождения отдельного человека или для идентификации человека в контексте.
- Позиционно-независимый исполняемый файл (PIE) - тело машинного кода, которое, будучи размещенным где-то в первичной памяти, выполняется правильно независимо от его абсолютного адреса.
- Инфраструктура открытых ключей (PKI) - схема, которая связывает открытые ключи с соответствующими идентификаторами объектов. Привязка устанавливается в процессе регистрации и выдачи сертификатов в центре сертификации (CA).
- Коммутируемая телефонная сеть общего пользования (PSTN) - традиционная телефонная сеть, включающая как стационарные, так и мобильные телефоны.
- Проверяющая сторона (RP) - обычно приложение, которое полагается на то, что пользователь прошел аутентификацию у отдельного провайдера аутентификации. Приложение полагается на какой-то токен или набор подписанных утверждений, предоставленных этим провайдером аутентификации, чтобы доверять тому, что пользователь является тем, кем они себя называют.
- Статическое тестирование безопасности приложений (SAST) - набор технологий, предназначенных для анализа исходного кода приложения, байтового кода и двоичных файлов для кодирования и условий разработки, которые указывают на уязвимости

безопасности. Решения SAST анализируют приложение «изнутри» в неработающем состоянии.

- Жизненный цикл разработки программного обеспечения (SDLC) - пошаговый процесс разработки программного обеспечения от начальных требований до развертывания и сопровождения.
- Архитектура безопасности - абстракция дизайна приложения, которая идентифицирует и описывает, где и как используются меры безопасности, а также идентифицирует и описывает расположение и конфиденциальность данных как пользователя, так и приложения.
- Конфигурация безопасности - конфигурация времени выполнения приложения, которая влияет на использование средств управления безопасностью.
- Контроль безопасности - функция или компонент, который выполняет проверку безопасности (например, проверку контроля доступа) или при вызове приводит к эффекту безопасности (например, создание записи аудита).
- Подделка запросов на стороне сервера (SSRF) - атака, которая злоупотребляет функциональностью сервера для чтения или обновления внутренних ресурсов путем предоставления или изменения URL-адреса, по которому код, запущенный на сервере, будет читать или отправлять данные.
- Аутентификация при едином входе (SSO) - это происходит, когда пользователь входит в одно приложение, а затем автоматически входит в другие приложения без повторной аутентификации. Например, когда вы входите в систему Google, при доступе к другим службам Google, таким как YouTube, Google Docs и Gmail, вы автоматически входите в систему.
- SQL-инъекция (SQLi) - метод внедрения кода, используемый для атаки приложения, управляемые данными, в которых вредоносные операторы SQL вставляются в точку входа.
- SVG - масштабируемая векторная графика
- OTP на основе времени - метод генерации OTP, при котором текущее время действует как часть алгоритма генерации пароля.
- Моделирование угроз - методика, состоящая в разработке все более совершенных архитектур безопасности для выявления агентов угроз, зон безопасности, средств управления безопасностью, а также важных технических и бизнес-активов.
- Безопасность транспортного уровня (TLS) - криптографические протоколы, обеспечивающие безопасность связи через сетевое соединение.
- Модуль доверенной платформы (TPM) - тип HSM, который обычно подключается к более крупному компоненту оборудования, например материнской плате, и действует как «корень доверия» для этой системы.
- Двухфакторная аутентификация (2FA) - добавляет второй уровень аутентификации для входа в учетную запись.
- Универсальный 2-й фактор (U2F) - один из стандартов, созданных FIDO специально для использования ключа безопасности USB или NFC в качестве 2-го фактора аутентификации.
- URI / URL / фрагменты URL - унифицированный идентификатор ресурса - это строка символов, используемая для идентификации имени или веб-ресурса. Унифицированный указатель ресурса часто используется как ссылка на ресурс.
- Проверяющий - человек или группа, которые проверяют приложение на соответствие требованиям OWASP ASVS.
- Что вы видите, то и получаете (WYSIWYG) - тип редактора расширенного содержимого, который показывает, как на самом деле будет выглядеть содержимое при визуализации, а не показывает кодировку, используемую для управления визуализацией.

- Сертификат X.509. Сертификат X.509 - это цифровой сертификат, в котором используется широко признанный международный стандарт инфраструктуры открытых ключей (PKI) X.509 для проверки принадлежности открытого ключа идентификатору пользователя, компьютера или службы, содержащемуся в сертификате.
- Внешняя сущность XML (XXE) - тип сущности XML, которая может получать доступ к локальному или удаленному контенту через объявленный системный идентификатор. Это может вызвать различные инъекционные атаки.

Appendix B: References

The following OWASP projects are most likely to be useful to users/adopters of this standard:

OWASP Core Projects

1. OWASP Top 10 Project: https://owasp.org/www-project-top-ten/
2. OWASP Web Security Testing Guide: https://owasp.org/www-project-web-security-testing-guide/
3. OWASP Proactive Controls: https://owasp.org/www-project-proactive-controls/
4. OWASP Security Knowledge Framework: https://owasp.org/www-project-security-knowledge-framework/
5. OWASP Software Assurance Maturity Model (SAMM): https://owasp.org/www-project-samm/

OWASP Cheat Sheet Series project

This project has a number of cheat sheets which will be relevant for different topics in the ASVS.

There is a mapping to the ASVS which can be found here: https://cheatsheetseries.owasp.org/cheatsheets/IndexASVS.html

Mobile Security Related Projects

1. OWASP Mobile Security Project: https://owasp.org/www-project-mobile-security/
2. OWASP Mobile Top 10 Risks: https://owasp.org/www-project-mobile-top-10/
3. OWASP Mobile Security Testing Guide and Mobile Application Security Verification Standard: https://owasp.org/www-project-mobile-security-testing-guide/

OWASP Internet of Things related projects

1. OWASP Internet of Things Project: https://owasp.org/www-project-internet-of-things/

OWASP Serverless projects

1. OWASP Serverless Project: https://owasp.org/www-project-serverless-top-10/

Others

Similarly, the following web sites are most likely to be useful to users/adopters of this standard

1. SecLists Github: https://github.com/danielmiessler/SecLists
2. MITRE Common Weakness Enumeration: https://cwe.mitre.org/
3. PCI Security Standards Council: https://www.pcisecuritystandards.org
4. PCI Data Security Standard (DSS) v3.2.1 Requirements and Security Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-2-1.pdf
5. PCI Software Security Framework - Secure Software Requirements and Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI-Secure-Software-Standard-v1_0.pdf
6. PCI Secure Software Lifecycle (Secure SLC) Requirements and Assessment Procedures: https://www.pcisecuritystandards.org/documents/PCI-Secure-SLC-Standard-v1_0.pdf

Приложение C: Требования к проверке Интернета вещей

Этот раздел изначально находился в основной ветке, но с учетом работы, проделанной командой OWASP IoT, нет смысла поддерживать два разных потока по этому вопросу. В версии 4.0 мы переносим это в Приложение и призываем всех, кому это необходимо, использовать основной проект OWASP IoT.

Контроль объектов

Встроенные устройства / устройства Интернета вещей должны:

- Обеспечьте тот же уровень контроля безопасности на устройстве, что и на сервере, за счет обеспечения контроля безопасности в доверенной среде.
- Конфиденциальные данные, хранящиеся на устройстве, должны выполняться безопасным образом с использованием аппаратного хранилища, такого как элементы безопасности.
- Все конфиденциальные данные, передаваемые с устройства, должны использовать безопасность транспортного уровня.

Требования к проверке безопасности

#	Описание	L1	L2	L3	Since
C.1	Убедитесь, что отладочные интерфейсы прикладного уровня, такие как USB, UART и другие последовательные варианты, отключены или защищены сложным паролем.	✓	✓	✓	4.0
C.2	Убедитесь, что криптографические ключи и сертификаты уникальны для каждого отдельного устройства.	✓	✓	✓	4.0
C.3	Убедитесь, что элементы управления защитой памяти, такие как ASLR и DEP, включены встроенной операционной системой / IoT, если применимо.	✓	✓	✓	4.0
C.4	Убедитесь, что встроенные отладочные интерфейсы, такие как JTAG или SWD, отключены или что доступный механизм защиты включен и настроен соответствующим образом.	✓	✓	✓	4.0
C.5	Убедитесь, что доверенное выполнение реализовано и включено, если оно доступно на SoC или CPU устройства.	✓	✓	✓	4.0
C.6	Убедитесь, что конфиденциальные данные, закрытые ключи и сертификаты надежно хранятся в Secure Element, TPM, TEE (Trusted Execution Environment) или защищены с помощью надежной криптографии.	✓	✓	✓	4.0
C.7	Убедитесь, что приложения микропрограмм защищают передаваемые данные с помощью безопасности транспортного уровня.	✓	✓	✓	4.0
C.8	Убедитесь, что приложения микропрограмм проверяют цифровую подпись подключений к серверу.	✓	✓	✓	4.0
C.9	Убедитесь, что беспроводная связь взаимно аутентифицирована.	✓	✓	✓	4.0
C.10	Убедитесь, что беспроводная связь передается по зашифрованному каналу.	✓	✓	✓	4.0
C.11	Убедитесь, что любое использование запрещенных функций C заменено соответствующими безопасными эквивалентными функциями.	✓	✓	✓	4.0

C.12	Убедитесь, что каждая микропрограмма поддерживает перечень программных материалов, в котором перечислены сторонние компоненты, версии и опубликованные уязвимости.	✓	✓	✓	4.0
C.13	Убедитесь, что весь код, включая сторонние двоичные файлы, библиотеки и фреймворки, проверяется на наличие жестко заданных учетных данных (бэкдоры).	✓	✓	✓	4.0
C.14	Убедитесь, что компоненты приложения и микропрограммы не подвержены внедрению команд ОС, вызвав оболочки команд оболочки, сценарии, или что меры безопасности предотвращают внедрение команд ОС.	✓	✓	✓	4.0
C.15	Убедитесь, что приложения микропрограмм закрепили цифровую подпись на доверенном сервере (ax)..		✓	✓	4.0
C.16	Убедитесь в наличии функций защиты от несанкционированного доступа и / или обнаружения несанкционированного доступа..		✓	✓	4.0
C.17	Убедитесь, что все доступные технологии защиты интеллектуальной собственности, предоставленные производителем микросхемы, включены.		✓	✓	4.0
C.18	Убедитесь, что используются средства защиты, препятствующие обратному проектированию микропрограмм (например, удаление подробных отладочных символов).		✓	✓	4.0
C.19	Перед загрузкой убедитесь, что устройство проверяет подпись загрузочного образа.		✓	✓	4.0
C.20	Убедитесь, что процесс обновления прошивки не уязвим для атак по времени проверки и времени использования.		✓	✓	4.0
C.21	Перед установкой убедитесь, что устройство использует подпись кода и проверяет файлы обновления прошивки.		✓	✓	4.0
C.22	Убедитесь, что устройство не может быть понижено до старых версий (анти-откат) действующей прошивки.		✓	✓	4.0
C.23	Проверить использование криптографически безопасного генератора псевдослучайных чисел на встроенном устройстве (например, с помощью предоставленных микросхемой генераторов случайных чисел).		✓	✓	4.0
C.24	Убедитесь, что микропрограмма может выполнять автоматические обновления микропрограммы по заранее определенному расписанию.		✓	✓	4.0
C.25	Убедитесь, что устройство стирает микропрограммное обеспечение и конфиденциальные данные при обнаружении взлома или получении недопустимого сообщения.			✓	4.0
C.26	Убедитесь, что используются только микроконтроллеры, поддерживающие отключение интерфейсов отладки (например, JTAG, SWD).			✓	4.0
C.27	Убедитесь, что используются только микроконтроллеры, обеспечивающие существенную защиту от атак по удалению ограничения и атак по побочным каналам.			✓	4.0
C.28	Убедитесь, что чувствительные следы не попадают на внешние слои печатной платы.			✓	4.0

C.29	Убедитесь, что межкристальная связь зашифрована (например, связь между материнской платой и дочерней платой).	✓	4.0
C.30	Убедитесь, что устройство использует подпись кода и проверяет код перед выполнением.	✓	4.0
C.31	Убедитесь, что конфиденциальная информация, хранящаяся в памяти, перезаписывается нулями, как только она больше не требуется.	✓	4.0
C.32	Убедитесь, что приложения микропрограмм используют контейнеры ядра для изоляции между приложениями.	✓	4.0
C.33	Убедитесь, что флаги безопасного компилятора, такие как -fPIE, -fstack-protector-all, -Wl, -z, noexecstack, -Wl, -z, noexecheap, настроены для сборок микропрограмм.	✓	4.0
C.34	Убедитесь, что микроконтроллеры настроены с защитой кода (если применимо).	✓	4.0

Использованная литература

Для получения дополнительной информации см. также:

- OWASP Internet of Things Top 10
- OWASP Embedded Application Security Project
- OWASP Internet of Things Project
- Trudy TCP Proxy Tool