Стандарт верификации требований к безопасности приложений 4.0.3

Финальная версия

Октябрь 2021

Table of Contents

Фронтиспис **7**

О стандарте *7*

Авторские права и лицензия *7*

Руководители проекта *7*

Основные участники *7*

Другие участники и рецензенты *7*

Предисловие **9**

Что нового в версии 4.0 ..... *9*

Использование ASVS **11**

Уровни соответствия ASVS *11*

Как применять этот стандарт *12*

Уровень 1 — Первые шаги, автоматизированный или поверхностный 12

Уровень 2 — Большинство приложений 12

Уровень 3 - Высокая стоимость, надежность или безопасность 12

Применение ASVS на практике *13*

Как ссылаться на требования ASVS *13*

Оценка и сертификация **14**

Позиция OWASP в отношении сертификатов и знаков доверия ASVS *14*

Руководство для сертифицирующих организаций *14*

Методы тестирования 14

Другие варианты использования ASVS *15*

Как подробное руководство по архитектуре безопасности 15

Как чек-лист по безопасному программированию 15

Как руководство по автоматизированному модульному и интеграционному тестированию 15

Как курс по безопасной разработке 15

Как движущая сила к Agile-процессу безопасности приложений 16

Как руководство по приобретению безопасного программного обеспечения 16

V1 Архитектура, проектирование и моделирование угроз **17**

Задачи контроля *17*

V1.1 Жизненный цикл разработки безопасного ПО *17*

V1.2 Аутентификация *18*

V1.3 Управление сессиями *18*

V1.4 Контроль доступа *19*

V1.5 Входные и выходные данные *19*

V1.6 Криптографическая защита *20*

V1.7 Обработка ошибок, ведение журнала и аудит *20*

V1.8 Защита информации и конфиденциальность *20*

V1.9 Передача данных *21*
V1.10 Вредоносный код *21*
V1.11 Бизнес-логика *21*
V1.12 Безопасная загрузка файлов *21*
V1.13 Архитектура API *22*
V1.14 Безопасные конфигурации *22*
Ссылки *22*

V2 Аутентификация **23**
Задачи контроля *23*
NIST 800-63 — современный стандарт аутентификации, основанный на доказательствах *23*
Выбор подходящего уровня доверия (AAL) в NIST 23
Условные обозначения *24*
V2.1 Парольная защита *24*
V2.2 Базовые требования *26*
V2.3 Жизненный цикл аутентификатора *27*
V2.4 Хранение учётных данных *28*
V2.5 Восстановление учётных данных *29*
V2.6 Проверочные коды *29*
V2.7 Второй фактор аутентификации *30*
V2.8 OTP *31*
V2.9 Криптографический верификатор *32*
V2.10 Технические учётные записи *32*
Дополнительные требования для госучреждений США *33*
Определения и сокращения *33*
Ссылки *33*

V3 Управление сессиями **34**
Задачи контроля *34*
Верификация требований к безопасности *34*
V3.1 Базовые требования *34*
V3.2 Создание сессии *34*
V3.3 Завершение сессии *35*
V3.4 Cookie *35*
V3.5 Токены *36*
V3.6 Федеративная повторная аутентификация *36*
V3.7 Меры защиты от уязвимостей сессии *37*
Описание «полуоткрытой» атаки 37
Ссылки *37*

V4 Контроль доступа 38
Задачи контроля 38
Верификация требований к безопасности 38
V4.1 Базовые принципы 38
V4.2 Этап эксплуатации 38
V4.3 Прочие соображения 39
Ссылки 39

V5 Форматно-логический контроль, нейтрализация и кодировка 40
Задачи контроля 40
V5.1 Контроль входных данных 40
V5.2 Нейтрализация и изоляция 41
V5.3 Кодирование выходных данных и предотвращение инъекций 42
V5.4 Память, строки и неуправляемый код 43
V5.5 Предотвращение десериализации 43
Ссылки 43

V6 Хранимая криптография 45
Задачи контроля 45
V6.1 Категории информации 45
V6.2 Алгоритмы 45
V6.3 Случайные значения 46
V6.4 Управление секретами 47
Ссылки 47

V7 Обработка ошибок и ведение журнала 48
Задачи контроля 48
V7.1 Содержимое журнала 48
V7.2 Обработка журнала 49
V7.3 Защита журнала 49
V7.4 Обработка ошибок 50
Ссылки 50

V8 Защита информации 51
Задачи контроля 51
V8.1 Базовые меры 51
V8.2 Защита данных на стороне клиента 52
V8.3 Персональные данные 52
Ссылки 53

V9 Передача данных 54

Задачи контроля *54*
V9.1 Безопасность подключений со стороны клиента *54*
V9.2 Безопасность подключений со стороны сервера *55*
Ссылки *55*

V10 Вредоносный код **56**
Задачи контроля *56*
V10.1 Целостность исходного кода *56*
V10.2 Поиск вредоносного кода *56*
V10.3 Целостность приложения *57*
Ссылки *57*

V11 Бизнес-логика **58**
Задачи контроля *58*
V11.1 Безопасность бизнес-логики *58*
Ссылки *59*

V12 Файлы и ресурсы **60**
Задачи контроля *60*
V12.1 Загрузка файлов на сервер *60*
V12.2 Целостность файлов *60*
V12.3 Исполнение файла *60*
V12.4 Хранение файлов *61*
V12.5 Загрузка файлов клиентом *61*
V12.6 Защита от SSRF *61*
Ссылки *62*

V13 API и web-сервисы **63**
Задачи контроля *63*
V13.1 Базовая безопасность web-сервисов *63*
V13.2 RESTful web-сервисы *64*
V13.3 Web сервисы SOAP *64*
V13.4 GraphQL *65*
Ссылки *65*

V14 Конфигурация **66**
Задачи контроля *66*
V14.1 Сборка и развёртывание *66*
V14.2 Зависимости *67*
V14.3 Непреднамеренное раскрытие информации о безопасности *67*
V14.4 HTTP-заголовки безопасности *68*

V14.5 Проверка заголовка HTTP-запроса *68*
Ссылки *69*

Приложение A: Термины и сокращения **70**

Приложение B: Полезные ссылки **74**
Основные проекты OWASP *74*
Проект OWASP Cheat Sheet Series *74*
Проекты Mobile Security *74*
Проект OWASP Internet of Things *74*
Проект OWASP Serverless *74*
Другие *74*

Приложение C: Требования к верификации Internet of Things **75**
Задачи контроля *75*
Верификация требований к безопасности *75*
Ссылки *77*

Фронтиспис

О стандарте

Стандарт верификации требований к безопасности приложений — это перечень требований к безопасности приложений (тестов), которыми могут пользоваться архитекторы, разработчики, тестировщики, специалисты по безопасности, разработчики инструментов и конечные пользователи для проектирования, разработки, тестирования и контроля безопасных приложений.

Авторские права и лицензия

Версия 4.0.3, октябрь 2021

Copyright © 2008-2021 The OWASP Foundation. Этот документ выпущен под лицензией Creative Commons Attribution ShareAlike 3.0. При воспроизведении или распространении этого документа необходимо разъяснить условия лицензии на него.

Руководители проекта

Andrew van der Stock Daniel Cuthbert Jim Manico

Josh C Grossman Elar Lang

Основные участники

Abhay Bhargav Benedikt Bauer Osama Elnaggar

Ralph Andalis Ron Perris Sjoerd Langkemper

Tonimir Kisasondi

Другие участники и рецензенты

Aaron Guzman	Alina Vasiljeva	Andreas Kurtz	Anthony Weems	Barbara Schachner
Christian Heinrich	Christopher Loessl	Clément Notin	Dan Cornell	Daniël Geerts
David Clarke	David Johansson	David Quisenberry	Elie Saad	Erlend Oftedal
Fatih Ersinadim	Filip van Laenen	Geoff Baskwill	Glenn ten Cate	Grant Ongers
hello7s	Isaac Lewis	Jacob Salassi	James Sulinski	Jason Axley
Jason Morrow	Javier Dominguez	Jet Anderson	jeurgen	Jim Newman
Jonathan Schnittger	Joseph Kerby	Kelby Ludwig	Lars Haulin	Lewis Ardern
Liam Smit	lyz-code	Marc Aubry	Marco Schnüriger	Mark Burnett
Philippe De Ryck	Ravi Balla	Rick Mitchell	Riotaro Okada	Robin Wood
Rogan Dawes	Ryan Goltry	Sajjad Pourali	Serg Belkommen	Siim Puustusmaa
Ståle Pettersen	Stuart Gunter	Tal Argoni	Tim Hemel	Tomasz Wrobel
Vincent De Schutter	Mike Jang			

Если Вы — участник проекта, но Вашего имени нет в приведенном выше списке, пожалуйста, зарегистрируйте issue на GitHub, для признания в будущих обновлениях.

Четвертая версия стандарта верификации требований к безопасности приложений опирается на предыдущие версии ASVS, начиная с первой, вышедшей в 2008 году, и до третьей, в 2016. Большая часть разделов оглавления и пунктов требований до сих пор присутствующих в ASVS, изначально были написаны Майком Боберски, Джеффом Уильямсом и Дэйвом Уичерсом, но участников было гораздо больше - спасибо им всем. Полные списки всех, кто внес свой вклад в более ранние версии, есть в соответствующих версиях документа.

Предисловие

Добро пожаловать в Стандарт верификации требований к безопасности приложений (ASVS) версии 4.0. ASVS — это инициатива сообщества по созданию системы требований и мер безопасности, направленных на определение функциональных и нефункциональных мер безопасности, необходимых при проектировании, разработке и тестировании современных web-приложений и web-сервисов.

Версия 4.0.3 — третья редакция для версии 4.0, предназначенная для исправления орфографических ошибок и уточнений без внесения существенных изменений, таких как значительные изменения, усиление или добавление требований. Однако некоторые требования, возможно, были слегка ослаблены там, где мы сочли это целесообразным, а некоторые избыточные требования были удалены (без изменения сквозной нумерации).

ASVS v4.0 — кульминация усилий профессионального сообщества и отзывов экспертов за последнее десятилетие. Мы попытались упростить внедрение ASVS для различных вариантов его использования на протяжении жизненного цикла разработки безопасного программного обеспечения.

Мы понимаем, что, скорее всего, никогда не сможем достичь полного согласия по содержанию любого стандарта, включая ASVS. Анализ рисков всегда в какой-то степени субъективен, что создает трудности при попытке обобщить его с помощью универсального стандарта. Однако мы надеемся, что последние обновления, внесенные в эту версию, станут шагом в правильном направлении и улучшат концепции, представленные в этом важнейшем профессиональном стандарте.

Что нового в версии 4.0

Наиболее значительным изменением в этой версии является принятие Стандарта по цифровой идентификации NIST 800-63-3, в котором представлены современные, основанные на доказательствах, и продвинутые способы аутентификации. Хотя мы и ожидаем некоторого сопротивления согласованию с новым стандартом, считаем, что это согласование важно, особенно с известным стандартом безопасности приложений, который основан на доказательствах.

Стандарты информационной безопасности должны стремиться свести к минимуму количество уникальных требований, чтобы соблюдающим их организациям не приходилось принимать решения о конкурирующих или несовместимых мерах. OWASP Top 10 2017, а теперь и OWASP Application Security Verification Standard теперь соответствуют NIST 800-63 в части аутентификации и управления сессиями. Мы призываем всех, кто разрабатывает стандарты, сотрудничать с нами, NIST и другими, чтобы прийти к общепринятому набору мер безопасности приложений, для обеспечения максимальной безопасности и минимизации затрат на соблюдение требований..

Разделы и пункты ASVS 4.0 были полностью перенумерованы. Новая схема нумерации позволила нам устранить пробелы в исчезнувших и разбить на части более длинные главы, чтобы свести к минимуму количество мер, которые должны соблюдать разработчики. Например, если приложение не использует JWT, то весь раздел о JWT в управлении сессиями для него неприменим.

Новым в версии 4.0 является сопоставление с Common Weakness Enumeration (CWE), — одна из наиболее частых просьб, которые мы получали за последнее десятилетие. Сопоставление с CWE позволяет разработчикам инструментов и тем, кто использует ПО управления уязвимостями, сопоставлять результаты других инструментов и предыдущих версий ASVS с этой и более поздними версиями. Чтобы освободить место для записи CWE, нам пришлось удалить столбец «Версия», который, поскольку мы полностью перенумеровали пункты, утратил смысл. Не каждый пункт в ASVS имеет связанный CWE, но, поскольку CWE имеет большое количество дублей, мы попытались использовать наиболее часто используемые, а не наиболее близкие совпадения. Пункты требований не всегда удается сопоставить с недостатками. Мы приветствуем продолжающуюся дискуссию между сообществами CWE и информационной безопасности в целом по устранению этого пробела.

Мы старались всесторонне удовлетворить и превзойти требования стандартов OWASP Top 10 2017 и OWASP Proactive Controls 2018. Поскольку OWASP Top 10 2017 — это минимальный уровень, то во избежание небрежности, мы умышленно присвоили всем требованиям Top 10 первый уровень соответствия, упрощая приверженцам OWASP переход к данному стандарту.

Мы стремились к тому, чтобы первый уровень соответствия ASVS 4.0 полностью охватывал раздел 6.5 стандарта PCI DSS 3.2.1, в части проектирования и разработки приложений, тестирования, анализа безопасности кода и тестов на проникновение. Это потребовало включения в раздел V5 требований по переполнению буфера и небезопасным операциям с памятью, а в V14 - по небезопасным флагам компиляции, связанным с памятью, в дополнение к существующим требованиям к проверке приложений и web-сервисов.

Мы завершили переход ASVS от мер безопасности, применимых только к монолитным архитектурам, к обеспечению безопасности для всех современных приложений и API. Во времена бессерверных API, мобильных устройств, облачных сред, контейнеров, CI/CD и DevSecOps, мы не можем продолжать игнорировать современную архитектуру приложений. Современные приложения разрабатываются совсем иначе, чем те, что создавались во время выпуска первой версии ASVS в 2009 году. ASVS всегда должен ориентироваться на будущее, чтобы мы могли давать разумные советы нашей основной аудитории — разработчикам. Мы разъяснили или исключили требования, предполагающие, что приложения выполняются в системах, принадлежащих только одной организации.

Из-за размера ASVS 4.0, а также из-за нашего желания стать базовым стандартом для других инициатив ASVS, мы отказались от "мобильной" главы в пользу стандарта исследования безопасности мобильных приложений (MASVS). Приложение «Интернет вещей» появится в будущем стандарте IoT ASVS в рамках проекта OWASP Internet of Things. Мы включили предварительную версию IoT ASVS в Приложение C. Мы благодарим команду OWASP Mobile и команду проекта OWASP IoT за их поддержку ASVS и надеемся на сотрудничество с ними в будущем при разработке дополнительных стандартов.

Наконец, мы устранили дубли и убрали менее результативные меры. Со временем ASVS стал комплексным набором мер, но не все они одинаково полезны для разработки безопасного ПО. Работа по устранению неэффективных мер продолжится. В будущей версии ASVS Common Weakness Scoring System (CWSS) поможет расставить приоритеты над мерами, которые действительно важны, и от которых следует отказаться.

Начиная с версии 4.0, ASVS будет сосредоточен исключительно на том, чтобы стать ведущим стандартом для web-приложений и сервисов, охватывая традиционную и современную архитектуры приложений, а также гибкие методы обеспечения безопасности (Agile) и культуру DevSecOps.

Использование ASVS

У ASVS две основные задачи:

- помочь организациям разрабатывать и поддерживать безопасные приложения.
- согласовать между собой требования потребителей и предложения разработчиков инструментов и поставщиков услуг кибербезопасности.

Уровни соответствия ASVS

Стандарт верификации требований к безопасности приложений определяет три уровня соответствия, каждый из которых усиливает требования предыдущего.

- Первый уровень ASVS соответствует низкому уровню безопасности и полностью поддается тестированию на проникновение.
- Второй уровень ASVS предназначен для приложений, содержащих конфиденциальную информацию, и являются рекомендуемым для большинства приложений.
- Третий уровень ASVS предназначен для наиболее критичных приложений, выполняющих финансовые транзакции, содержащих медицинские данные и других приложений, требующих высочайшего уровня доверия.

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

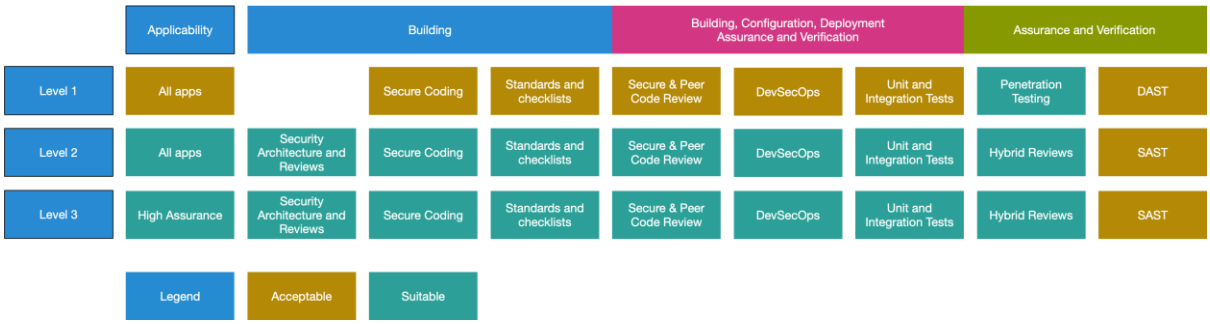

Рис. 1 — Уровни OWASP ASVS версии 4.0

Первый уровень — единственный, который можно полностью протестировать на проникновение вручную. Все остальные требуют доступа к документации, исходному коду, конфигурации и к специалистам, участвующим в процессе разработки. Однако, даже если L1 и допускает тестирование по принципу «черного ящика» (без документации и исходных кодов), он не является достаточным для достижения соответствия, и на этом не следует останавливаться. У злоумышленников огромный запас времени, а большинство тестов на проникновение проводятся в течение пары недель. Защитникам необходимо встраивать меры безопасности, защищать, находить и устранять все уязвимости, а также обнаруживать злоумышленников и реагировать на них в разумные сроки. У злоумышленники по сути бесконечное время, и для успеха им требуется только одно слабое звено в защите, одна уязвимость или недостаток обнаружения. Тестирование методом «черного ящика», часто выполняемое в конце разработки, наспех или не выполняемое вовсе, совершенно не справляется с этой асимметрией.

За последние 30 с лишним лет тестирование методом "черного ящика" снова и снова доказывало, что оно пропускает критические проблемы безопасности, которые непосредственно приводили ко все более массовым нарушениям. Мы настоятельно рекомендуем использовать широкий спектр инструментов по анализу защищённости, включая замену тестов на проникновение на основе исходного кода (гибридными) тестами на проникновение на L1 с полным доступом к разработчикам и документации на протяжении всего процесса разработки. Финансовые регуляторы не начинают внешний финансовый аудит без доступа к бухгалтерским книгам, выборкам транзакций и к людям, их

выполняющим. Государства и профессиональные сообщества должны требовать такого же уровня прозрачности и в области разработки программного обеспечения.

Мы настоятельно рекомендуем использовать инструменты анализа в самом процессе разработки. DAST и SAST могут быть встроены в конвейер сборки, чтобы налету обнаруживать проблемы безопасности, которых не должно быть.

Автоматизированные инструменты и online-сканирование не в состоянии выполнить более половины требований ASVS без помощи человека. Если требуется комплексная автоматизация тестирования для каждой сборки, то используется комбинация модульных и интеграционных тестов, а также online-сканирование, инициированное сборкой. Тестирование недостатков бизнес-логики и контроля доступа возможно только с помощью человека. Оно должно быть встроено в модульные и интеграционные тесты.

Как применять этот стандарт

Один из лучших способов применения ASVS — использовать его в качестве основы для создания чек-листа безопасной разработки для вашего приложения, платформы или организации. Адаптация ASV к вашим сценариям использования позволит уделять больше внимания требованиям безопасности, которые наиболее важны для ваших проектов и сред.

Уровень 1 — Первые шаги, автоматизированный или поверхностный

Приложение достигает первого уровня ASVS, если оно адекватно защищает от уязвимостей безопасности, которые легко обнаружить, которые включены в список OWASP Top 10 и другие подобные списки.

Первый уровень — это абсолютный минимум, к которому должны стремиться все приложения без исключений. Он также полезен в качестве первого шага в многоэтапной работе или когда приложения не хранят и не обрабатывают конфиденциальную информацию, и, следовательно, не нуждаются в более строгих мерах безопасности второго или третьего уровня. Элементы управления уровня 1 могут быть проверены либо автоматически с помощью инструментов, либо просто вручную без доступа к исходному коду. Мы считаем уровень 1 минимально необходимым для всех приложений.

Чаще всего, угрозы для приложения исходят от нарушителей, которые пользуются простыми, не требующими больших усилий методами для выявления легко обнаруживаемых и легко эксплуатируемых уязвимостей. Совсем другое дело, когда приложение атакует заинтересованный и опытный злоумышленник. Если данные, обрабатываемые вашим приложением, имеют большую ценность, вы вряд ли захотите останавливаться на первом уровне.

Уровень 2 — Большинство приложений

Приложение достигает второго уровня ASVS, если оно адекватно защищает от большинства рисков, связанных с современным программным обеспечением.

Уровень 2 гарантирует наличие и эффективность применения мер безопасности в приложении. Уровень 2 обычно подходит для приложений, которые обрабатывают важные бизнес-транзакции, в том числе те, которые обрабатывают данные о здоровье, решают критически важные для бизнеса задачи, или в отраслях, где целостность является значимым аспектом для защиты бизнеса, например, в игровой индустрии, чтобы помешать читерам и взлому игр.

Угрозы для приложений уровня 2 обычно представляют собой квалифицированных и мотивированных нарушителей, сосредоточенных на конкретных целях, использующих инструменты и методы, которые хорошо отработаны и эффективны при обнаружении и использовании слабых мест в приложениях.

Уровень 3 - Высокая стоимость, надежность или безопасность

Третий уровень ASVS — самый высокий в рамках ASVS. Этот уровень обычно необходим для приложений, требующих серьезных гарантий безопасности, например, в таких областях, как оборона, здравоохранение, критическая инфраструктура и т.д.

Организациям может потребоваться ASVS уровня 3 для приложений, выполняющих критически важные функции, где сбой может существенно повлиять на непрерывность работы и даже на ее сохранение как

организации. Ниже приводится примерное руководство по применению ASVS уровня 3. Приложение достигает третьего (продвинутого) уровня ASVS, если оно адекватно защищено от серьезных уязвимостей безопасности приложений, а также соответствует концептуальной архитектуре безопасности приложений.

Приложение на третьем уровне ASVS требует более глубокого анализа архитектуры, разработки и тестирования, чем на других уровнях. Защищенное приложение обычно имеет модульную структуру (для обеспечения отказоустойчивости, масштабируемости и, прежде всего, уровней безопасности), при этом каждый модуль (отделенный от других сетевым подключением и / или физически) выполняет свои собственные задачи по обеспечению безопасности (эшелонированная защита), которые должны быть должным образом задокументированы. Эти задачи включают меры обеспечения конфиденциальности (шифрование), целостности (например, транзакционной, форматно-логический контроль), доступности (например корректная работа под нагрузкой), аутентификация (в том числе между системами), авторизация и аудит (ведение журнала событий безопасности).

Применение ASVS на практике

Разные угрозы имеют разную мотивацию. В отдельных отраслях имеются уникальные информационные и технологические ресурсы или отраслевые регуляторные требования.

Организациям настоятельно рекомендуется внимательно изучить риски, присущие характеру их бизнеса, и на их основе и с учетом бизнес-требований определить соответствующий уровень ASVS.

Как ссылаться на требования ASVS

Каждое требование имеет идентификатор в формате <глава>.<раздел>.<требование> где каждый элемент является числом, например: 1.11.3.

- Значение <глава> соответствует главе, в которой находится это требование, например: все требования 1.#.# находятся в главе Архитектура.
- Значение <раздел> соответствует разделу в этой главе, где находится это требование, например: все требования 1.11.# находятся в разделе Бизнес-логика главы Архитектура.
- Значение <требование> указывает на конкретное требование в главе и разделе, например: 1.11.3, которым в версии 4.0.3 этого стандарта является:

Убедитесь, что все критичные потоки бизнес-логики, включая аутентификацию, управление сессиями и контроль доступа, являются безопасными с точки зрения их параллельного выполнения и устойчивыми к условиям гонки, вызванной разницей между состояниями приложения в момент проверки и в момент использования (TOCTOU).

Идентификаторы требований могут меняться от версии к версии, поэтому желательно, чтобы другие документы, отчеты или инструменты использовали формат: v<версия>-<глава>.<раздел>.<требование>, где: 'версия' это метка версии ASVS. Например: v4.0.3-1.11.3 будет означать 3-е требование в разделе 'Бизнес-логика' главы 'Архитектура' в версии 4.0.3. (Резюмируя, v<версия>-<идентификатор_требования>.)

Примечание. Буква v, предшествующая версии, должна быть в нижнем регистре.

Считается, что если идентификаторы указаны без элемента v<версия>, то они относятся к последней версии стандарта. Очевидно, что по мере развития и изменения стандарта это станет проблематичным, поэтому авторы и разработчики должны включать элемент версии.

Списки требований ASVS доступны в форматах CSV, JSON и других, которые могут оказаться полезными для справки или использования в программах.

Оценка и сертификация

Позиция OWASP в отношении сертификатов и знаков доверия ASVS

OWASP, будучи независимой от поставщиков некоммерческой организацией, в настоящее время не сертифицирует разработчиков, аудиторов или программное обеспечение.

Подобные заверения, знаки доверия или сертификаты официально не контролируются, не регистрируются и не сертифицируются OWASP, поэтому организация, полагающаяся на них, должна проявлять осторожность в отношении доверия к любой третьей стороне или знаку доверия, претендующему на сертификацию ASVS.

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

Руководство для сертифицирующих организаций

Стандарт верификации требований к безопасности приложений может использоваться в качестве открытой книги, подразумевая открытый и неограниченный доступ к ключевым ресурсам, таким как архитекторы и разработчики, проектная документация, исходный код, санкционированный доступ к тестовым системам (включая доступ к одной или нескольким учетным записям в каждой роли), особенно для проверок на L2 и L3.

Исторически так сложилось, что при тестировании на проникновение и анализе безопасности кода проблемы учитывались в виде исключения, то есть в окончательный отчет попадали только неудачные тесты. Сертифицирующая организация должна включать в отчет область исследования (особенно, если ключевой компонент, например, single sign-on, остается за ее рамками), сводку результатов тестирования, включая пройденные и непройденные тесты, с четкими указаниями, как устранить причины по непройденным тестам.

Некоторые требования стандарта могут быть неприменимы к тестируемому приложению. Например, если вы предоставляете потребителям REST API без сохранения состояния без реализации клиента, то многие требования раздела V3 "Управление сессиями" напрямую неприменимы. В таких случаях сертифицирующая организация может по-прежнему заявлять о полном соответствии ASVS, но должна чётко указывать в отчете причину неприменимости исключенных требований.

Включение подробных рабочих документов, снимков или видео экрана, сценариев эксплуатации уязвимостей и т.п. в отчет о тестировании, считается стандартной практикой и может быть действительно полезным в качестве доказательств для сомневающихся разработчиков. Недостаточно просто запустить инструмент и сообщить о несоблюдении требования; само по себе это еще не является достаточных основанием того, что все недостатки подтверждены и тщательно протестированы. В случае спора должны быть представлены достаточные аргументы, чтобы продемонстрировать, что каждое требование было действительно верифицировано.

Методы тестирования

Сертифицирующие организации свободны в выборе подходящего метода (методов) тестирования, но должны явно указать их в отчете.

В зависимости от тестируемого приложения и уровня требований разные методы могут обеспечить равную уверенность в результатах. Например, механизм форматно-логического контроля входных данных можно проверить либо с помощью теста на проникновение, либо, анализируя исходный код.

Роль инструментов автоматизированного тестирования безопасности

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

Невозможно провести проверку ASVS полностью только с помощью автоматизированных инструментов. Хотя подавляющее большинство требований в L1 может быть выполнено с помощью автоматизированных тестов, все же остаются требования, не поддающиеся автоматизации.

Пожалуйста, обратите внимание, что границы между автоматизированным и ручным тестированием размываются по мере совершенствования безопасности приложений. Автоматизированные инструменты часто настраиваются вручную экспертами, а при ручном тестировании часто используется 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 требования к ведению журнала, которые необходимы для соответствия пункту A10 в OWASP Top 10 2017, потребуют интервью, снимков экрана или других доказательств (как и в OWASP Top 10 2017). Однако тестирование без доступа к необходимой информации не является идеальным методом, поскольку при этом упускается возможность проверки исходного кода, выявления угроз и отсутствующих мер защиты, а также выполнения гораздо более тщательного тестирования в более короткие сроки.

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

Другие варианты использования ASVS

Помимо использования для оценки безопасности приложения, мы предлагаем ряд других потенциальных применений для ASVS.

Как подробное руководство по архитектуре безопасности

Одним из наиболее распространенных применений Стандарта верификации безопасности приложений является использование в качестве ресурса для архитекторов безопасности. В Sherwood Applied Business Security Architecture (SABSA) отсутствует большой пласт информации, необходимой для проведения тщательного анализа архитектуры безопасности приложений. ASVS можно использовать для восполнения этого пробела, давая возможность выбирать более эффективные меры для решения общих проблем, таких как архитектурные шаблоны защиты данных и стратегии контроля входных данных.

Как чек-лист по безопасному программированию

Многие организации могут извлечь выгоду из внедрения ASVS, выбрав один из трех уровней или адаптируя ASVS под себя, изменяя требования для каждого уровня риска приложения в зависимости от нужд своей компании. Мы поощряем эту адаптацию, если поддерживается отслеживаемость, т.е., если приложение удовлетворяет требованию 4.1 для адаптированной версии, значит оно удовлетворяет и требованию стандарта по мере его развития.

Как руководство по автоматизированному модульному и интеграционному тестированию

ASVS создавался таким, чтобы его можно было легко протестировать, за исключением требований к архитектуре и вредоносному коду. Благодаря модульным и интеграционным тестам, которые проверяют характерные случаи фаззинга и недокументированного использования, приложение с каждой сборкой становится практически самотестируемым. Например, для набора тестов для контроллера входа в систему можно разработать дополнительные тесты, проверяющие атрибут username на совпадение с наиболее распространенными именами пользователей по умолчанию, перебор учетных записей, LDAP- и SQL-инъекции, а также XSS. Аналогично, проверка атрибута password должна включать перебор самых распространенных паролей, оценку длины пароля, вставку нулевого байта, удаление этого атрибута, XSS и многое другое.

Как курс по безопасной разработке

ASVS также можно использовать для определения характеристик защищенного программного обеспечения. Многие "курсы безопасной разработки" — это просто курсы по этичному хакингу с легким намеком на советы по безопасной разработке. Не факт, что это как-то поможет разработчикам писать более безопасный код. Вместо этого курсы по безопасной разработке могут использовать ASVS с упором на упреждающие меры, описанные в ASVS, а не на Top 10 того, что делать не надо.

Как движущая сила к Agile-процессу безопасности приложений

ASVS можно использовать в процессе Agile-разработки в качестве основы для определения задач, которые должны быть реализованы, чтобы получить безопасный продукт. Один из подходов может быть следующим: начиная с первого уровня, проверить конкретное приложение или систему на соответствии с требованиями ASVS для указанного уровня, определить недостающие меры безопасности, и поставить задачи на их доработку. Это помогает расставить приоритеты конкретных задач и делает безопасность видимой в Agile-процессе. Это также может быть использовано для определения приоритетов задач анализа и аудита. Каждое требование ASVS может быть движущей силой для проведения рефакторинга или аудита конкретным участником или командой, и отображаться как его "долг" в очереди задач, который необходимо рано или поздно выполнить.

Как руководство по приобретению безопасного программного обеспечения

ASVS — это отличное руководство по приобретению безопасного программного обеспечения или заказных услуг по его разработке. Заказчику достаточно указать требование, что программное обеспечение, которое он хочет приобрести, должно быть разработано на уровне ASVS LX, и потребовать, чтобы Исполнитель доказал, что оно ему соответствует. Это неплохо сочетается с Приложением OWASP к Договору на безопасное программное обеспечение.

V1 Архитектура, проектирование и моделирование угроз

Задачи контроля

Во многих организациях архитектура безопасности сегодня стала почти забытым занятием. Слава корпоративных архитекторов в эпоху DevSecOps проходит. Область безопасности приложений должна подхватить это знамя, и, адаптируясь к принципам Agile, представить свои передовые принципы архитектуры безопасности разработчикам программного обеспечения. Архитектура — это не реализация, а способ осмысления проблемы, которая потенциально имеет много разных решений и не имеет единственного правильного ответа. Часто безопасность считается негибкой, требуя, чтобы разработчики исправляли код определённым образом, однако разработчики часто могут знать гораздо лучший способ решения. Не существует единого простого решения в архитектуре, и делать вид, что это не так, — медвежья услуга разработчикам программного обеспечения.

Конкретная реализация web-приложения, возможно, будет постоянно меняться на протяжении всего его жизненного цикла, но общая архитектура, скорее всего, будет меняться редко, и развиваться медленно. Так же и архитектура безопасности — аутентификация нужна нам сегодня, и завтра, и через пять лет. Если мы примем правильные решения сегодня, то сможем сэкономить много сил, времени и средств, выбрав и повторно используя архитектурно-совместимые решения. Например, лет десять назад многофакторная аутентификация почти не применялась.

Если бы разработчики вложились в единую защищённую модель поставщика удостоверений, такую как федеративные удостоверения SAML, её можно было бы обновить, включив только новые требования, например, из стандарта NIST 800-63, без изменения интерфейсов исходного приложения. Если множество приложений имеют общую архитектуру безопасности и, следовательно, общие компоненты, то все они получат выгоду от этого обновления одновременно. Однако SAML не всегда будет лучшим или наиболее подходящим решением для аутентификации — возможно, его придётся заменить другими решениями по мере изменения требований. Подобные изменения либо сложны, либо настолько дорогостоящи, что требуют полной переделки, либо совершенно невозможны без архитектуры безопасности.

В этой главе ASVS охватывает основные аспекты правильной архитектуры безопасности: доступность, конфиденциальность, целостность обработки, неотказуемость и конфиденциальность. Каждый из этих принципов безопасности должен быть встроен во все приложения. Крайне важно «двигаться влево», начиная с поддержки разработчиков с помощью чек-листов безопасного программирования, наставничества и обучения, разработки и тестирования, сборки, развёртывания, конфигурирования и эксплуатации, и заканчивая приёмо-сдаточным тестированием, чтобы убедиться, что все необходимые меры безопасности приняты и они работают. Раньше этот последний шаг и был всем, что мы делали. Но теперь этого недостаточно, ведь разработчики выпускают код в промышленную среду десятки или сотни раз в день. Специалисты по безопасности приложений должны идти в ногу с гибкими методами, что означает изучение инструментов разработки, обучение программированию и совместную работу с разработчиками, а не критику проекта спустя месяцы после того, как все остальные продвинулись дальше.

V1.1 Жизненный цикл разработки безопасного ПО

№	Описание	L1	L2	L3	CWE
1.1.1	Убедитесь в использовании безопасного жизненного цикла разработки программного обеспечения, который обеспечивает безопасность на всех этапах разработки. (C1)		✓	✓	
1.1.2	Убедитесь в применении моделирования угроз для каждого изменения дизайна или планирования спринта, чтобы выявлять угрозы, планировать контрмеры, способствовать надлежащему реагированию на риски и управлять тестированием безопасности.		✓	✓	1053

№	Описание	L1	L2	L3	CWE
1.1.3	Убедитесь, что все пользовательские истории и функции содержат функциональные ограничения безопасности. Например, такие: "Как пользователь, я должен иметь возможность просматривать и редактировать свой профиль, но не должен иметь возможности просматривать или редактировать чужой профиль".		✓	✓	1110
1.1.4	Убедитесь в наличии документации и обоснования для всех границ доверия приложения, его компонентов и значимых информационных взаимодействий.		✓	✓	1059
1.1.5	Проверьте описание концептуальной архитектуры приложения, связанных с ним внешних систем и анализ их безопасности. (C1)		✓	✓	1059
1.1.6	Проконтролируйте наличие централизованных, простых в реализации, проверенных, безопасных и переиспользуемых мер безопасности, чтобы избежать их дублирования или, наоборот, отсутствия, а также неэффективных или небезопасных мер. (C10)		✓	✓	637
1.1.7	Убедитесь в наличии чек-листа для безопасной разработки, требований, политик и стандартов безопасности, а также в их доступности для разработчиков и тестировщиков.		✓	✓	637

V1.2 Аутентификация

При проектировании системы аутентификации не имеет значения, насколько безопасна ваша многофакторная аутентификация, если злоумышленник может сбросить пароль учетной записи, просто позвонив в контакт-центр, и ответив на общеизвестные вопросы. При подтверждении личности все пути аутентификации должны иметь одинаковую силу.

№	Описание	L1	L2	L3	CWE
1.2.1	Убедитесь в использовании всеми компонентами приложений, служб и серверов уникальных или специальных учетных записей операционной системы с низким уровнем привилегий. (C3)		✓	✓	250
1.2.2	Убедитесь, что взаимодействие между всеми компонентами приложения, включая API, промежуточное ПО и уровень хранения данных, аутентифицировано. Компоненты должны иметь минимально необходимые привилегии. (C3)		✓	✓	306
1.2.3	Убедитесь, что приложение использует единый утвержденный механизм аутентификации, который может быть дополнен включением строгой аутентификации, а также обеспечивает адекватное журналирование и мониторинг для обнаружения нарушений политик безопасности или взлома учетной записи.		✓	✓	306
1.2.4	Убедитесь, что все пути аутентификации и API идентификации реализуют заданный уровень безопасности, и что более слабые альтернативы отсутствуют.		✓	✓	306

V1.3 Управление сессиями

Место вставки для будущих архитектурных требований.

V1.4 Контроль доступа

№	Описание	L1	L2	L3	CWE
1.4.1	Убедитесь, что контроль доступа применяется в пределах контролируемоой зоны (т.е. на шлюзах безопасности, серверах и бессерверных функциях). Нельзя применять контроль доступа на стороне клиента.		✓	✓	602
1.4.2	[УДАЛЕНО, НЕВОЗМОЖНО ПРОВЕРИТЬ]				
1.4.3	[УДАЛЕНО, ДУБЛИРУЕТ 4.1.3]				
1.4.4	Убедитесь, что приложение использует единый и тщательно протестированный 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 при управлении доступом на основе атрибутов или функций, код проверяет доступ пользователя к функции/данным, а не только его роль. Разрешения в любом случае должны выдаваться в соответствии с ролью. (C7)		✓	✓	275

V1.5 Входные и выходные данные

В версии 4.0 мы отошли от термина «серверная часть» как термина, обозначающего границу доверия. Граница доверия как термин по-прежнему вызывает сомнения — контроль доступа в недоверенных браузерах или клиентских устройствах можно обойти. Однако в современных архитектурных концепциях смысл точки обеспечения доверия кардинально изменился. Поэтому, когда в ASVS используется термин «уровень доверенного сервиса», мы подразумеваем любую доверенную точку применения политики управления доступом, независимо от ее местоположения, например, это может быть микросервис, бессерверный API, серверная часть, доверенный API на клиентском устройстве, которое имеет доверенную загрузку, партнерские или внешние API и т.д.

Термин «недоверенный клиент» здесь относится к технологиям на стороне клиента, которые отображают уровень представления, обычно называемый интерфейсным (front-end). Термин «сериализация» здесь относится не только к отправке данных по сети, например, массива значений или получению и чтению структуры JSON, но и к передаче сложных объектов, которые могут содержать логику, например, Data Transfer Objects, (DTO).

№	Описание	L1	L2	L3	CWE
1.5.1	Убедитесь, что требования к входным и выходным данным четко определяют, как обращаться с данными и обрабатывать их в зависимости от типа, содержимого и применимых законов, нормативных документов и других политик.		✓	✓	1029
1.5.2	Убедитесь, что сериализация не используется при взаимодействии с недоверенными клиентами. Если это невозможно, убедитесь, что применяются адекватные меры контроля целостности (и, при необходимости, шифрование при передаче конфиденциальных данных) для предотвращения атак десериализации, включая внедрение объектов.		✓	✓	502
1.5.3	Убедитесь, что форматно-логический контроль входных данных выполняется на уровне доверенного сервиса. (C5)		✓	✓	602
1.5.4	Убедитесь, что кодировка и/или экранирование выходных данных проводится интерпретатором, для которого они предназначены, или непосредственно перед ним. (C4)		✓	✓	116

V1.6 Криптографическая защита

Приложения должны разрабатываться в соответствии с утвержденной архитектурой криптографической защиты информационных ресурсов с учетом их категории конфиденциальности. Шифровать всё — это расточительство, а не шифровать вовсе — правовой нигилизм. Компромисс обычно находится во время архитектурного или концептуального проектирования, спринтов по дизайну. Разработка криптографии по ходу проекта или ее последующая модернизация обойдутся гораздо дороже, чем встроить ее с самого начала.

Архитектурные требования предъявляются ко всей кодовой базе, поэтому их трудно охватить модульными или интеграционными тестами. Требования к архитектуре требуют учета в стандартах разработки на протяжении всего жизненного цикла и должны учитываться на архитектуре безопасности, при оценке кода коллегами или на ретроспективах.

№	Описание	L1	L2	L3	CWE
1.6.1	Убедитесь в наличии формализованной политики управления криптографическими ключами и что жизненный цикл криптографического ключа соответствует стандарту управления ключами, например NIST SP 800-57.		✓	✓	320
1.6.2	Убедитесь, что потребители криптографических сервисов защищают ключевой материал и другие секреты с помощью хранилища ключей (vault) или альтернатив на основе API.		✓	✓	320
1.6.3	Убедитесь, что все ключи и пароли являются заменяемыми и являются частью формализованного и легко воспроизводимого процесса шифрования чувствительных данных.		✓	✓	320
1.6.4	Убедитесь, что на уровне архитектуры секреты на стороне клиента, такие как симметричные ключи, пароли или API-токены, считаются по умолчанию небезопасными, и никогда не используются для защиты или доступа к конфиденциальным данным.		✓	✓	320

V1.7 Обработка ошибок, ведение журнала и аудит

№	Описание	L1	L2	L3	CWE
1.7.1	Убедитесь, что в системе используются единый подход и формат ведения журнала событий. (C9)		✓	✓	1009
1.7.2	Убедитесь, что журналы событий безопасности надежно передаются в предпочтительно удалённую систему для их дальнейшего анализа, обнаружения аномалий, оповещения соответствующих специалистов и эскалации. (C9)		✓	✓	

V1.8 Защита информации и конфиденциальность

№	Описание	L1	L2	L3	CWE
1.8.1	Убедитесь, что все чувствительные данные идентифицированы и классифицированы по уровням защиты.		✓	✓	
1.8.2	Убедитесь, что каждому уровню защиты соответствует набор требований безопасности (целостность, конфиденциальность, сроки хранения и др.), и что эти требования удовлетворяются в архитектуре.		✓	✓	

V1.9 Передача данных

№	Описание	L1	L2	L3	CWE
1.9.1	Убедитесь, что приложение шифрует обмен данными между компонентами, особенно когда эти компоненты находятся в разных контейнерах, системах, сайтах или у разных провайдеров облачной инфраструктуры. (C3)		✓	✓	319
1.9.2	Убедитесь, что компоненты приложения аутентифицируют каждую сторону взаимодействия, чтобы предотвратить атаки "человек посередине". Например, компоненты приложения должны проверять сертификаты и цепочки доверия TLS.		✓	✓	295

V1.10 Вредоносный код

№	Описание	L1	L2	L3	CWE
1.10.1	Убедитесь, что используется система управления исходным кодом и применяются процедуры, гарантирующие, что регистрация кода сопровождается отчетами о проблемах или заявками на изменение. Система управления исходным кодом должна контролировать доступ и идентифицировать пользователей, чтобы можно было отслеживать каждое изменение кода.		✓	✓	284

V1.11 Бизнес-логика

№	Описание	L1	L2	L3	CWE
1.11.1	Проверьте определение и документацию на компоненты приложения с точки зрения бизнес-функций или функций безопасности, которые они обеспечивают.		✓	✓	1059
1.11.2	Убедитесь, что все ключевые потоки бизнес-логики, включая аутентификацию, управление сессиями и контроль доступа, не используются в несинхронизированном состоянии.		✓	✓	362
1.11.3	Убедитесь, что все критичные потоки бизнес-логики, включая аутентификацию, управление сессиями и контроль доступа, являются безопасными с точки зрения их параллельного выполнения и устойчивыми к условиям гонки, вызванной разницей между состояниями приложения в момент проверки и в момент использования (TOCTOU).			✓	367

V1.12 Безопасная загрузка файлов

№	Описание	L1	L2	L3	CWE
1.12.1	[УДАЛЕНО, ДУБЛИРУЕТ 12.4.1]				
1.12.2	Убедитесь, что загружаемые пользователем файлы — если требуется их отображение или загрузка из приложения — сохраняются либо как MIME-type: application/octet-stream, либо из несвязанного домена, например из облачного хранилища файлов, Чтобы снизить риск воздействия XSS-векторов или других атак, связанных с загружаемыми файлами, применяется соответствующая политика безопасности контента (CSP).		✓	✓	646

V1.13 Архитектура API

Место вставки для будущих архитектурных требований.

V1.14 Безопасные конфигурации

№	Описание	L1	L2	L3	CWE
1.14.1	Контролируйте разделение компонентов с разными уровнями доверия с помощью шлюзов безопасности, правил брандмауэра, шлюзов API, обратных прокси-серверов, облачных групп безопасности и т.п.		✓	✓	923
1.14.2	Убедитесь, что при развертывании бинарных файлов на удаленных устройствах контролируются их сигнатуры, используются доверенные соединения и конечные точки из списка разрешенных.		✓	✓	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 содержит этап автоматической сборки и проверки безопасного развертывания приложения, особенно если инфраструктура приложения определяется программно, например сценарием сборки в облачной среде.		✓	✓	
1.14.5	Убедитесь, что развертывание приложений проводится в «песочнице», контейнере и/или изолировано на сетевом уровне, чтобы сдержать нарушителя от атак на другие приложения, особенно когда они выполняют чувствительные или опасные действия, например, десериализацию. (C5)		✓	✓	265
1.14.6	Убедитесь, что приложение не использует неподдерживаемые, небезопасные или устаревшие клиентские технологии, такие как подключаемые модули NSAPI, Flash, Shockwave, ActiveX, Silverlight, NACL или клиентские Java-апплеты.		✓	✓	477

Ссылки

Для дополнительной информации см. также:

- OWASP Threat Modeling Cheat Sheet
- OWASP Attack Surface Analysis Cheat Sheet
- OWASP Threat modeling
- OWASP Software Assurance Maturity Model Project
- Microsoft SDL
- NIST SP 800-57

V2 Аутентификация

Задачи контроля

Аутентификация — это действие по установлению или подтверждению подлинности кого-либо (или чего-либо) и того, что утверждения, сделанные человеком или устройством, верны, защищены от подмены и предотвращают восстановление или перехват паролей.

Когда ASVS был впервые опубликован, имя пользователя и пароль были наиболее распространенной формой аутентификации (за исключением систем, содержащих сведения, составляющие государственную тайну). Многофакторная аутентификация (МФА) была широко известна в узких кругах специалистов, но редко применялась. По мере увеличения количества взломов идея о том, что имена пользователей каким-то образом конфиденциальны, а пароли неизвестны, оказалась несостоятельной (вместе с мерами безопасности, которые на ней основывались). Например, NIST 800-63 рассматривает имена пользователей и аутентификацию на основе знаний — Knowledge Based Authentication (KBA) — как общедоступную информацию; уведомления по SMS и электронной почте — как «ограниченные» типы аутентификаторов, а все пароли — как предварительно взломанные. Эта реальность делает бесполезными аутентификаторы, основанные на знаниях, восстановление по SMS и электронной почте, историю паролей, сложность и контроль ротации. Эти меры и раньше были не очень полезными, часто вынуждая пользователей каждые несколько месяцев придумывать такие же слабые пароли, но когда счет взломанных учетных записей пошёл на миллиарды, пришло время двигаться дальше.

Из всех глав ASVS больше всего изменились главы об аутентификации и управлении сессиями. Внедрение эффективной, основанной на фактических данных передовой практики для многих будет сложной задачей, и это совершенно нормально. Мы должны начать переход к пост-парольному будущему уже сейчас.

NIST 800-63 — современный стандарт аутентификации, основанный на доказательствах

NIST 800-63b — современный, основанный на фактических данных стандарт, который представляет собой наилучшие из имеющихся рекомендаций, независимо от их применимости. Стандарт полезен для всех организаций по всему миру, но особенно актуален для государственных служб в США и тех, кто работает с ними.

Поначалу терминология NIST 800-63 может поначалу немного сбивать с толку, особенно если раньше вы пользовались только аутентификацией по имени пользователя и паролю. Чтобы описать достижения в области современной аутентификации необходимо ввести новую терминологию, которая станет общепринятой в будущем. Мы осознаем сложность ее понимания, пока сообщество не примет эти новые термины, поэтому в конце этой главы мы привели глоссарий, чтобы помочь вам. Мы перефразировали многие требования, чтобы лучше донести «дух» требования, а не его «букву». Например, в ASVS используется термин «пароль», тогда как в NIST — «запоминаемый секрет».

Разделы ASVS V2 Аутентификация, V3 Управление сессиями, и, в меньшей степени, V4 Контроль доступа были адаптированы для соответствия с избранными мерами в NIST 800-63b, ориентированными на основные угрозы и часто эксплуатируемые недостатки аутентификации. Если требуется полное соответствие требованиям NIST 800-63, лучше обратиться непосредственно к NIST 800-63.

Выбор подходящего уровня доверия (AAL) в NIST

В Стандарте верификации требований к безопасности приложений мы попытались сопоставить ASVS L1 с требованиями NIST уровня доверия AAL1, L2 с AAL2 и L3 с AAL3. Однако выбор базовых мер в ASVS L1 не обязательно окажется правильным AAL для тестирования приложения или API. Например, если приложение относится к L3 или соответствует нормативным требованиям AAL3, следует выбрать L3 в главах V2 и V3 Управление сессиями. Выбор уровня доверия (AAL) для аутентификации, совместимого с NIST, должен соответствовать рекомендациям NIST 800-63b, изложенным в разделе *Выбор AAL* в NIST 800-63b раздел 6.2.

Условные обозначения

Приложения могут превосходить требования текущего уровня, особенно если современные механизмы аутентификации входят в план развития приложения. Ранее для ASVS требовалась обязательная МФА, а в NIST нет требования обязательности. Поэтому в этой главе мы использовали дополнительные обозначения, чтобы указать, где ASVS поощряет, но не требует контроля:

Знак	Описание
	Не требуется
o	Рекомендуется
✓	Обязательно

V2.1 Парольная защита

Пароли, называемые в NIST 800-63 «запоминаемыми секретами», включают в себя пароли, PIN-коды, узоры разблокировки, выбор правильного изображения из набора и парольные фразы. Они обычно считаются «чем-то, что вы знаете» и часто используются в качестве однофакторных аутентификаторов. Существуют серьезные проблемы с дальнейшим использованием однофакторной аутентификации, включая миллиарды действительных имен пользователей и паролей, раскрытых в Интернете, стандартные или слабые пароли, радужные таблицы и упорядоченные словари наиболее распространенных паролей.

Приложения должны настоятельно рекомендовать пользователям включать многофакторную аутентификацию и позволять применять токены, которые у них уже есть, такие как FIDO или U2F, или ссылаться на поставщика услуг учетных данных (Credential Service Provider, CSP), который обеспечивает многофакторную аутентификацию.

CSP предоставляют федеративную идентификацию для пользователей. У пользователей часто бывает множество цифровых удостоверений у разных CSP, например корпоративное на Azure AD, Okta или Ping Identity, и личные — на Яндекс, VK или Google, и это лишь некоторые из распространенных вариантов. Этот список не является рекомендацией этих компаний или их услуг, а просто побуждает разработчиков учитывать тот факт, что многие пользователи имеют несколько разных цифровых удостоверений. Организации должны рассмотреть возможность интеграции с существующими учётными записями пользователей в соответствии с профилем риска, который определяется доверием к удостоверениям, предоставляемым каждым CSP. Например, маловероятно, что какой-либо госорган примет учётную запись из социальной сети в качестве входа в государственную информационную систему, поскольку легко создать поддельную или одноразовую учётную запись, в то время как компания, разрабатывающая мобильные игры, вполне может интегрироваться с основными платформами социальных сетей, чтобы увеличить базу активных игроков.

№	Описание	L1	L2	L3	CWE	NIST
2.1.1	Убедитесь, что устанавливаемые пользователем пароли имеют длину не менее 12 символов (после объединения пробелов, если они идут подряд). (C6)	✓	✓	✓	521	5.1.1.2
2.1.2	Убедитесь, что разрешены пароли длиной не менее 64 символов и что пароли длиной более 128 символов запрещены. (C6)	✓	✓	✓	521	5.1.1.2
2.1.3	Убедитесь, что отсутствует усечение пароля по длине. Идущие подряд несколько пробелов могут быть заменены одним. (C6)	✓	✓	✓	521	5.1.1.2
2.1.4	Убедитесь, что в паролях разрешены любые печатные символы Unicode, включая нейтральные к языку символы, такие как пробелы и эмодзи.	✓	✓	✓	521	5.1.1.2

№	Описание	L1	L2	L3	CWE	NIST
2.1.5	Убедитесь, что пользователи могут менять свой пароль.	✓	✓	✓	620	5.1.1.2
2.1.6	Убедитесь, что при изменении пароля требуется ввести текущий и новый пароль пользователя.	✓	✓	✓	620	5.1.1.2
2.1.7	Убедитесь (локально или с помощью внешнего API), что пароли, введенные при регистрации учетной записи, входе в систему или смене пароля, проверяются на вхождение в состав скомпрометированных (например, в 1000 или 10 000 наиболее распространенных). Для API при проверке следует использовать доказательство с нулевым разглашением или другой механизм, гарантирующий, что пароль не будет отправлен открытым текстом. Если пароль скомпрометирован, то приложение должно потребовать от пользователя ввести другой. (C6)	✓	✓	✓	521	5.1.1.2
2.1.8	Убедитесь в наличии индикатора надежности пароля, который поможет пользователям установить более стойкий пароль.	✓	✓	✓	521	5.1.1.2
2.1.9	Убедитесь, что правила составления пароля не ограничивают допустимый тип символов. Не должно быть никаких требований к наличию символов в верхнем или нижнем регистре, цифр или специальным символов. (C6)	✓	✓	✓	521	5.1.1.2
2.1.10	Убедитесь, что нет требований к периодической замене учетных данных или истории паролей.	✓	✓	✓	263	5.1.1.2
2.1.11	Убедитесь, что разрешены функции вставки пароля в соответствующие поля экранных форм, а также утверждены браузерные расширения по управлению паролями и/или менеджеры паролей.	✓	✓	✓	521	5.1.1.2
2.1.12	Убедитесь, что пользователь имеет возможность временно приоткрыть весь маскированный пароль, либо последний введенный символ (если такая функция не включена по умолчанию).	✓	✓	✓	521	5.1.1.2

Примечание: Цель предоставления пользователю возможности временно просматривать свой пароль или его последний символ состоит в том, чтобы повысить удобство ввода учётных данных, особенно для длинных паролей, парольных фраз и в менеджерах паролей.

V2.2 Базовые требования

Для приложений, ориентированных на будущее необходима гибкость в отношении аутентификаторов. Для включения дополнительных средств аутентификации в соответствии с предпочтениями пользователя, а также для планомерного вывода из эксплуатации устаревших или небезопасных средств аутентификации код приложения необходимо регулярно пересматривать.

NIST рассматривает email и SMS как «ограниченные» типы аутентификаторов, и они, видимо, будут исключены из NIST 800-63, и, следовательно, из ASVS в будущих версиях. Приложения должны запланировать отказ от использования электронной почты или SMS при аутентификации.

№	Описание	L1	L2	L3	CWE	NIST
2.2.1	Убедитесь, что для снижения последствий от взлома учетных данных реализованы механизмы защиты от атак методом перебора и блокирования учетных записей. Такие механизмы включают запрет выбора наиболее распространенных скомпрометированных паролей, временную блокировку учетной записи, ограничение частоты запросов, CAPTCHA, увеличивающиеся интервалы между попытками, ограничение списка допустимых IP-адресов, или основанные на оценке рисков, например, ограничение по местоположению, времени входа и т.п. Убедитесь, что для одной учетной записи разрешено, не более ста 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 механизмов аутентификации (таких как SMS и email) ограничивается проверкой второго фактора или подтверждением транзакций, а не заменой более безопасных механизмов. Убедитесь, что сначала предлагаются более надежные методы, а затем более слабые; что пользователи осведомлены о рисках; и что приняты надлежащие меры для ограничения рисков компрометации учётной записи.	✓	✓	✓	304	5.2.10
2.2.3	Убедитесь, что после обновления учетных данных, например, сброса пароля, изменения электронной почты или адреса, входа в систему из неизвестного места, пользователям отправляются уведомления. Предпочтительно использование push-уведомлений, а не SMS или email, но при невозможности использования push-уведомлений допустимы SMS или email, при условии, что в уведомлении не раскрывается конфиденциальная информация.	✓	✓	✓	620	
2.2.4	Убедитесь в устойчивости к фишингу. Для защиты используются многофакторная аутентификация; мобильные устройства с push-аутентификацией (позволяющие подтвердить или отклонить намерение пройти аутентификацию через push-уведомления). На более высоких уровнях AAL — сертификаты на стороне клиента.			✓	308	5.2.5
2.2.5	Убедитесь, что если провайдер учётных данных (CSP) и аутентифицирующее приложение разделены, между их конечными точками используется протокол mTLS с взаимной аутентификацией.			✓	319	5.2.6

№	Описание	L1	L2	L3	CWE	NIST
2.2.6	Убедитесь в устойчивости к атакам повторного воспроизведения, требуя обязательного использования одноразовых паролей (OTP), криптографических аутентификаторов или проверочных кодов.			✓	308	5.2.8
2.2.7	Убедитесь, что намерение пройти аутентификацию, подтверждается требованием ввести OTP или инициированным пользователем действием, например, нажатием кнопки на аппаратном ключе FIDO.			✓	308	5.2.9

V2.3 Жизненный цикл аутентификатора

Аутентификаторы — это пароли, программные и аппаратные токены и биометрические устройства. Жизненный цикл аутентификаторов имеет решающее значение для безопасности приложения. Если зарегистрировать учетную запись без подтверждения ее подлинности, то она не вызовет большого доверия. Для сайтов социальных сетей такие в порядке вещей, но для безопасности банковских приложений внимание к регистрации и выдаче учетных данных и устройств имеет решающее значение.

Примечание: Пароли не должны иметь максимального срока действия или подлежать замене. Их следует регулярно проверять на предмет взлома, а не заменять.

№	Описание	L1	L2	L3	CWE	NIST
2.3.1	Убедитесь, что создаваемые системой начальные пароли или коды активации формируются криптографически случайным образом. Они ДОЛЖНЫ иметь длину не менее 6 символов и МОГУТ содержать буквы и цифры, а срок их действия истекает через короткий период времени. Нельзя допускать, чтобы начальный пароль становился долгосрочным.	✓	✓	✓	330	5.1.1.2 / A.3
2.3.2	Убедитесь, что поддерживается регистрация и использование предоставленных пользователем устройств аутентификации, таких как токены U2F или FIDO.		✓	✓	308	6.1.3
2.3.3	Убедитесь, что для обновления аутентификаторов с ограниченным сроком действия инструкции по их замене направляются заблаговременно.		✓	✓	287	6.1.4

V2.4 Хранение учётных данных

Архитекторам и разработчикам следует сверяться с этим разделом при создании или рефакторинге кода. Изложенные здесь требования можно полностью проверить, анализируя исходный код или с помощью модульных или интеграционных тестов безопасности. Тестирование на проникновение не сможет выявить ни одной из этих проблем.

Список утвержденных односторонних функций формирования ключей подробно описан в разделе 5.1.1.2 NIST 800-63 B, и в BSI Kryptographische Verfahren: Empfehlungen und Schlussellängen (2018). Вместо этих вариантов требования к алгоритмам и длине ключа можно взять из актуальных версий национальных стандартов.

Требования этого раздела нельзя проверить пентестом, поэтому они не отмечены в L1. Однако этот раздел имеет жизненно важное значение для безопасности учетных данных в случае их кражи. Поэтому, если вы применяете ASVS в качестве руководства по архитектуре, разработке или чек-листа проверки исходного кода, пожалуйста, включите эти меры в L1 в вашей собственной версии.

№	Описание	L1	L2	L3	CWE	NIST
2.4.1	Убедитесь, что пароли хранятся в форме, устойчивой к атакам в режиме offline. Пароли ДОЛЖНЫ быть «засолены» и хэшированы с использованием утвержденной односторонней функции формирования ключа или функции хеширования. При генерации хэша функция принимает на вход пароль, соль и показатель стоимости. (C6)		✓	✓	916	5.1.1.2
2.4.2	Убедитесь, что соль имеет как минимум 32 бит энтропии и выбирается произвольно, чтобы минимизировать коллизии значений соли между сохраненными хэшами. Для КАЖДОГО пароля должны хранится уникальное значение соли и результирующий хэш. (C6)		✓	✓	916	5.1.1.2
2.4.3	Убедитесь, что если используется функция PBKDF2, то количество итераций ДОЛЖНО быть настолько большим, насколько это позволяет производительность сервера верификации, но не менее 100000 итераций. (C6)		✓	✓	916	5.1.1.2
2.4.4	Убедитесь, что working factor при использовании bcrypt велик настолько, насколько это позволяет производительность сервера верификации, но не менее 10. (C6)		✓	✓	916	5.1.1.2
2.4.5	Убедитесь, что выполняется дополнительная итерация функции формирования ключа с использованием значения соли, которое является секретным и известно только верификатору. Соль формируется с помощью утвержденного генератора случайных битов в соответствии с NIST SP 800-90Ar1 и обеспечивается хотя бы минимальный уровень безопасности, указанный в последней редакции NIST SP 800-131A. Значение секретной соли ДОЛЖНО храниться отдельно от хэшированных паролей (в специализированном устройстве, например, в HSM (аппаратном модуле безопасности).		✓	✓	916	5.1.1.2

Вместо или в дополнение к приведенным в этом документе стандартам США могут использоваться национальные стандарты.

V2.5 Восстановление учётных данных

№	Описание	L1	L2	L3	CWE	NIST
2.5.1	Убедитесь, что сгенерированный системой начальный секрет активации или восстановления не отправляется пользователю в виде открытого текста. (C6)	✓	✓	✓	640	5.1.1.2
2.5.2	Убедитесь, что не используются подсказки для проверки пароля или аутентификация на основе знаний (так называемые «секретные вопросы»).	✓	✓	✓	640	5.1.1.2
2.5.3	Убедитесь, что при восстановлении забытого пароля никоим образом не раскрывается текущий пароль. (C6)	✓	✓	✓	640	5.1.1.2
2.5.4	Убедитесь, что отсутствуют неперсонифицированные учетные записи или учетные записи по умолчанию (например, root, admin, sa, guest и т.п.).	✓	✓	✓	16	5.1.1.2 / A.3
2.5.5	Убедитесь, что при изменении фактора аутентификации пользователь был уведомлен об этом событии.	✓	✓	✓	304	6.1.2.3
2.5.6	Убедитесь, что пути восстановления пароля (Забыли пароль? и др.) используют безопасный механизм, например, OTP на основе времени (TOTP), мобильное push-уведомление или иной механизм передачи по дополнительному каналу. (C6)	✓	✓	✓	640	5.1.1.2
2.5.7	Убедитесь, что в случае потери OTP или любого другого фактора аутентификация выполняется с тем же уровнем безопасности, что и во время регистрации.		✓	✓	308	6.1.2.3

V2.6 Проверочные коды

Проверочные коды – это предварительно сгенерированные списки секретных кодов, аналогичные номерам авторизации транзакций (TAN), кодам восстановления в социальных сетях или кодовым таблицам, содержащим набор случайных значений, безопасным образом распространяемые среди пользователей. Каждый код используется только один раз, и когда все они использованы, кодовая таблица становится недействительной. Этот тип аутентификаторов считается «чем-то, что у вас есть».

№	Описание	L1	L2	L3	CWE	NIST
2.6.1	Убедитесь, что проверочный код из кодовой таблицы можно использовать только один раз.		✓	✓	308	5.1.2.2
2.6.2	Убедитесь, что проверочные коды имеют достаточную случайность (112 бит энтропии). Если их энтропия меньше 112 бит, они «засолены» уникальной и случайной 32-битной солью и хэшированы с помощью утвержденной односторонней функции хэширования.		✓	✓	330	5.1.2.2
2.6.3	Убедитесь, что проверочные коды устойчивы к атакам offline. Например, что в них не используются предсказуемые значения.		✓	✓	310	5.1.2.2

V2.7 Второй фактор аутентификации

В прошлом обычным механизмом проверки второго фактора аутентификации был email или SMS, содержащие ссылку для сброса пароля. Злоумышленники используют этот слабый механизм для сброса пароля, которые они хотят контролировать, например, захватывая учетную запись электронной почты жертвы, и используя обнаруженную там ссылку для установки собственного пароля. Есть способы обработки второго фактора и получше.

Безопасный второй фактор — это физическое устройство, которое может обмениваться данными с верификатором по отдельному безопасному каналу. Пример — push-уведомления на мобильные устройства. Этот фактор считается «чем-то, что у вас есть». Когда пользователь хочет пройти аутентификацию, проверяющее приложение отправляет сообщение напрямую или косвенно через сторонний сервис. Сообщение содержит код аутентификации (обычно это случайное шестизначное число или модальное диалоговое окно подтверждения). Проверяющее приложение ожидает получения кода аутентификации по основному каналу и сравнивает значения полученного хэша с хэшем отправленного кода аутентификации. Если они совпадают, то верификатор может считать, что пользователь прошел аутентификацию.

ASVS предполагает, что мало кто будет разрабатывать с нуля новые факторы аутентификации, такие как push-уведомления, и, таким образом, следующие меры ASVS применяются к таким верификаторам, как API и приложения аутентификации, реализации единого входа. При разработке новых аутентификаторов, см. § 5.1.3.1.

Электронная почта и VOIP в качестве второго фактора недопустимы. Аутентификация через ТСОП и SMS в настоящее время «ограничивается» в NIST и должна быть заменена на push-уведомления и т.п. Если вам необходимо использовать телефон или SMS как второй фактор, см. § 5.1.3.3.

№	Описание	L1	L2	L3	CWE	NIST
2.7.1	Убедитесь, что по умолчанию не предлагается передавать второй фактор аутентификации открытым текстом через SMS или ТСОП. Т.е. сначала предлагаются более безопасные альтернативы, например, push-уведомления.	✓	✓	✓	287	5.1.3.2
2.7.2	Убедитесь, что срок действия запросов аутентификации, кодов или токенов, передаваемых по дополнительному каналу, истекает не позже, чем через 10 минут.	✓	✓	✓	287	5.1.3.2
2.7.3	Убедитесь, что запросы аутентификации, коды или токены, передаваемые по дополнительному каналу, можно использовать только один раз и только для исходного запроса аутентификации.	✓	✓	✓	287	5.1.3.2
2.7.4	Убедитесь, что проверяющая и проверяемая стороны взаимодействуют по отдельному защищенному каналу.	✓	✓	✓	523	5.1.3.2
2.7.5	Убедитесь, что проверяющая сторона сохраняет только хешированную версию кода аутентификации.		✓	✓	256	5.1.3.2
2.7.6	Убедитесь, что код аутентификации формируется криптографическим генератором случайных чисел, и содержит не менее 20 бит энтропии (обычно достаточно шестизначного случайного числа).		✓	✓	310	5.1.3.2

V2.8 OTP

Однофакторные одноразовые пароли (OTP) — это физические или программные токены, которые отображают постоянно меняющееся псевдослучайное одноразовые значение. Эти устройства делают фишинг (т.е. выдачу себя за другое лицо) трудным, но не невозможным. Этот тип аутентификатора считается «чем-то, что у вас есть». Многофакторные токены аналогичны однофакторным одноразовым паролям, но требуют ввода действующего PIN-кода, биометрической разблокировки, USB-токена, сопряжения по NFC или некоторого дополнительного значения (например, калькулятора подписи транзакций) для создания одноразового пароля.

№	Описание	L1	L2	L3	CWE	NIST
2.8.1	Убедитесь, что основанные на времени одноразовые пароли (OTP) имеют конечный срок действия.	✓	✓	✓	613	5.1.4.2 / 5.1.5.2
2.8.2	Убедитесь, что симметричные ключи, используемые для проверки отправленных одноразовых паролей, надежно защищены, например, с помощью аппаратного модуля безопасности (HSM) или безопасного хранилища ключей. предоставляемого операционной системой.		✓	✓	320	5.1.4.2 / 5.1.5.2
2.8.3	Убедитесь, что при создании, заполнении и проверке одноразовых паролей используются утвержденные криптографические алгоритмы.		✓	✓	326	5.1.4.2 / 5.1.5.2
2.8.4	Убедитесь, что одноразовый пароль на основе времени можно использовать 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 на основе времени используется повторно в течение срока действия, то это событие регистрируется, токен не принимается, а владельцу устройства направляется уведомление.		✓	✓	287	5.1.5.2
2.8.6	Убедитесь, что физический генератор однофакторных одноразовых паролей может быть отозван в случае его кражи или утери. Убедитесь, что отзыв вступает в силу немедленно во всех сессиях, в которых выполнен вход, независимо от местоположения.		✓	✓	613	5.2.1
2.8.7	Убедитесь, что биометрические факторы аутентификации можно использовать только в качестве дополнительных, в сочетании с тем, что у вас есть, или с чем-то, что вы знаете.		o	✓	308	5.2.3

V2.9 Криптографический верификатор

Криптографические ключи безопасности — это смарт-карты или ключи FIDO, к которым пользователь должен подключить или связать криптографическое устройство с компьютером для завершения аутентификации. Верификатор отправляет одноразовый запрос криптографическому устройству или приложению, которые вычисляют ответ на основе хранимого в тайне криптографического ключа.

Требования к однофакторным и многофакторным криптографическим устройствам и приложениям одинаковы, поскольку успешная проверка криптографического аутентификатора подтверждает обладание фактором аутентификации.

№	Описание	L1	L2	L3	CWE	NIST
2.9.1	Убедитесь, что криптографические ключи, используемые при проверке, хранятся в тайне и защищены от разглашения, например с помощью модуля доверенной платформы (TPM) или аппаратного модуля безопасности (HSM) или службы ОС, которая может использовать это защищенное хранилище.		✓	✓	320	5.1.7.2
2.9.2	Убедитесь, что одноразовая случайная строка (nonce) в запросе имеет длину не менее 64 бит и является уникальной статистически или в течение срока службы криптографического устройства.		✓	✓	330	5.1.7.2
2.9.3	Убедитесь, что при создании и проверке используются утвержденные криптографические алгоритмы.		✓	✓	327	5.1.7.2

V2.10 Технические учётные записи

Этот раздел не поддается тестированию на проникновение, поэтому не имеет требований L1. Однако, если у вас в архитектуре, разработке или анализе безопасного кода используются секреты, пожалуйста, рассматривайте программное хранилище (такое как доверенное хранилище ключей в Java) как минимальное требование на первом уровне. Хранение секретов в открытом виде недопустимо ни при каких обстоятельствах.

№	Описание	L1	L2	L3	CWE	NIST
2.10.1	Убедитесь, что секреты технических учетных записей не зависят от неизменных учётных данных, таких как пароли, ключи API или неперсонифицированные учетные записи с привилегированным доступом.		ОС	HSM	287	5.1.1.1
2.10.2	Убедитесь, что если для аутентификации технической учётной записи требуется пароль, то она не является учётной записью по умолчанию. Например, во время установки иногда используются root /root или admin / admin.		ОС	HSM	255	5.1.1.1
2.10.3	Убедитесь, что пароли хранятся с достаточной защитой для предотвращения атак восстановления offline, включая доступ к локальной системе.		ОС	HSM	522	5.1.1.1
2.10.4	Убедитесь, что пароли, строки подключения к базам данных и внешним системам, векторы инициализации и внутренние секреты, а также ключи API хранятся в тайне, и не включаются в исходный код, не хранятся в репозиториях исходного кода. Такое хранилище ДОЛЖНО противостоять атакам в режиме offline. Для хранения паролей рекомендуется использовать безопасное хранилище программных ключей (L1), аппаратный TPM или HSM (L3).		ОС	HSM	798	

Дополнительные требования для госучреждений США

У госучреждений в США есть обязательные требования, касающиеся NIST 800-63. Стандарт верификации требований к безопасности приложений всегда старался охватить те 80% мер, которые применимы почти ко всем приложениям, а не те 20%, которые имеют ограниченную применимость. Сам ASVS является подмножеством NIST 800-63 в части уровней IAL1/2 и AAL1/2, но не полностью заменяет его, особенно в отношении уровней IAL3/AAL3.

Мы настоятельно рекомендуем госучреждениям США изучить и внедрить у себя NIST 800-63 во всей его полноте.

Определения и сокращения

Термин	Определение
CSP	Поставщик услуг учетных данных (Credential Service Provider) также называемый поставщиком цифровых удостоверений (Identity Provider)
Аутентификатор	Код, который подтверждает подлинность пароля, токена, МФА, федеративного утверждения и т.д.
Верификатор	Сущность, которая подтверждает личность заявителя, удостоверяя, что он обладает одним или двумя аутентификаторами с использованием протокола аутентификации. Для этого верификатору также может потребоваться проверить учетные данные, которые связывают аутентификатор(ы) с идентификатором пользователя и проконтролировать их статус
OTP	Одноразовый пароль (One-time password)
SFA	Однофакторные аутентификаторы (Single-factor authenticators), например, то, что вы знаете (запомненные секреты, пароли, парольные фразы, PIN-коды), то, чем вы являетесь (биометрические данные, отпечатки пальцев, сканирование лица) или то, что у вас есть (токены OTP, криптографическое устройство, такое как смарт-карта).
МФА	Многофакторная аутентификация (Multi-factor authentication, MFA), включающая два или более одиночных фактора.

Ссылки

Для дополнительной информации см. также:

- NIST 800-63 - Digital Identity Guidelines
- NIST 800-63 A - Enrollment and Identity Proofing
- NIST 800-63 B - Authentication and Lifecycle Management
- NIST 800-63 C - Federation and Assertions
- NIST 800-63 FAQ
- OWASP Testing Guide 4.0: Testing for Authentication
- OWASP Cheat Sheet - Password storage
- OWASP Cheat Sheet - Forgot password
- OWASP Cheat Sheet - Choosing and using security questions

V3 Управление сессиями

Задачи контроля

Одним из основных компонентов любого web-приложения или API с отслеживанием состояния является механизм, с помощью которого оно контролирует и поддерживает состояние пользователя или устройства, взаимодействующего с ним. Управление сессиями превращает протокол без сохранения состояния в протокол с его сохранением, что имеет решающее значение для разграничения пользователей или устройств.

Убедитесь, что проверяемое приложение удовлетворяет следующим концептуальным требованиям к управлению сессиями:

- Сессии уникальны для каждого пользователя, их идентификаторы нельзя угадать, и ими нельзя «поделиться» с другими.
- Сессии становятся недействительными, когда они больше не нужны, т.е. при бездействии пользователя срок действия сессии истекает.

Как отмечалось ранее, эти требования были адаптированы для соответствия подмножеству мер в NIST 800-63b, ориентированных на наиболее распространенные угрозы и часто эксплуатируемые недостатки аутентификации. Из предыдущих требований были исключены дубли.

Верификация требований к безопасности

V3.1 Базовые требования

№	Описание	L1	L2	L3	CWE	NIST
3.1.1	Убедитесь, что приложение не указывает сессионные токены в параметрах URL.	✓	✓	✓	598	

V3.2 Создание сессии

№	Описание	L1	L2	L3	CWE	NIST
3.2.1	Убедитесь, что при аутентификации пользователя приложение каждый раз создает новый сессионный токен. (C6)	✓	✓	✓	384	7.1
3.2.2	Убедитесь, что сессионные токены обладают энтропией как минимум в 64-бит. (C6)	✓	✓	✓	331	7.1
3.2.3	Убедитесь, что приложение хранит сессионные токены в браузере безопасно, например, в защищенных файлах cookie (см. раздел V.3.4) или в хранилище сессий HTML5.	✓	✓	✓	539	7.1
3.2.4	Убедитесь, что сессионные токены генерируются с использованием утвержденных криптографических алгоритмов. (C6)		✓	✓	331	7.1

TLS или другой безопасный канал передачи является обязательным для управления сессиями. Об этом говорится в главе «Передача данных».

V3.3 Завершение сессии

Период ожидания сессии приведен в соответствие со стандартом NIST 800-63, который допускает гораздо более продолжительные значения, чем в других стандартах по безопасности. Организации должны проанализировать приведенную ниже таблицу, и если желателен более длительный период ожидания в зависимости от риска приложения, значение NIST должно быть верхним пределом времени простоя.

L1 в этом контексте — это IAL1/AAL1, L2 — это IAL2/AAL3, L3 — это IAL3/AAL3. Для IAL2/AAL2 и IAL3/AAL3 более короткий период ожидания является нижней границей времени для выхода из системы или повторной аутентификации для возобновления сессии.

№	Описание	L1	L2	L3	CWE	NIST
3.3.1	Убедитесь, что при выходе из системы и по истечению срока действия сессионный токен становится недействительным, таким образом, чтобы кнопка «Назад» или нижестоящая проверяющая сторона не возобновляли аутентифицированную сессию, в том числе между проверяющими сторонами. (C6)	✓	✓	✓	613	7.1
3.3.2	Если механизмы аутентификации позволяют пользователям оставаться в системе, убедитесь, что как при активном использовании, так и после периода простоя, периодически проводится повторная аутентификация. (C6)	30 дней	12 часов или 30 минут бездействия, МФА как опция	12 часов или 15 минут бездействия, МФА обязательна	613	7.2
3.3.3	Убедитесь, что после успешного изменения пароля (в т.ч. после его сброса/восстановления) приложение дает возможность завершить все активные сессии, и что это действие распространяется на приложения при федеративном входе в систему (при его наличии) и на проверяющие стороны.		✓	✓	613	
3.3.4	Убедитесь, что пользователи могут видеть все свои активные на данный момент сессии и устройства и (после повторного ввода учетных данных) выходить из любой из них или из всех сразу.		✓	✓	613	7.1

V3.4 Cookie

№	Описание	L1	L2	L3	CWE	NIST
3.4.1	Убедитесь, что для сессионных токенов на основе файлов cookie установлен атрибут Secure. (C6)	✓	✓	✓	614	7.1.1
3.4.2	Убедитесь, что для сессионных токенов на основе cookie установлен атрибут HttpOnly. (C6)	✓	✓	✓	1004	7.1.1

№	Описание	L1	L2	L3	CWE	NIST
3.4.3	Убедитесь, что сессионные токены на основе cookie используют атрибут SameSite, установленный в значение Lax или Strict, чтобы ограничить подверженность атакам по подмене межсайтовых запросов (CSRF). (C6)	✓	✓	✓	16	7.1.1
3.4.4	Убедитесь, что токены сессии на основе cookie используют префикс "__Host-". Таким образом, cookie отправляются только на тот хост, который изначально его установил, т.е. нельзя установить несколько cookie с одинаковым именем из другого поддомена или с другим значением домена или с другим значением пути.	✓	✓	✓	16	7.1.1
3.4.5	Убедитесь, что если приложение публикуется под одним доменным именем с другими приложениями, которые также устанавливают или используют свои сессионные cookie, и могут раскрывать их, то задайте атрибут Path, используя максимально точный путь. (C6)	✓	✓	✓	16	7.1.1

V3.5 Токены

Управление сессиями на основе токенов включает в себя JWT, OAuth, SAML и API-ключи. Известно, что последние являются наиболее слабыми и не должны использоваться в новых разработках.

№	Описание	L1	L2	L3	CWE	NIST
3.5.1	Убедитесь, что приложение позволяет пользователям отзывать Oauth-токены, которые формируют отношения доверия со связанными приложениями.		✓	✓	290	7.1.2
3.5.2	Убедитесь, что приложение использует сессионные токены, а не статические секреты и API-ключи, за исключением устаревших реализаций.		✓	✓	798	
3.5.3	Убедитесь, что сессионные токены без сохранения состояния используют электронную подпись, шифрование и другие контрмеры для защиты от атак подделки, обертывания, воспроизведения, нулевого шифрования и подмены ключей.		✓	✓	345	

V3.6 Федеративная повторная аутентификация

Этот раздел предназначен для тех, кто разрабатывает код для проверяющей стороны (англ.: Relying Party, RP) или поставщика учётных данных (англ.: Credential Service Provider, CSP). Если вы полагаетесь на код, реализующий эти функции, убедитесь, что следующие ниже требования удовлетворяются.

№	Описание	L1	L2	L3	CWE	NIST
3.6.1	Убедитесь, что на проверяющей стороне (RP) определено предельно допустимое время аутентификации для поставщика учетных данных (CSP), и что CSP должны требовать повторной аутентификации пользователя, в случае бездействия в течение этого периода.			✓	613	7.2.1
3.6.2	Убедитесь, что поставщик учетных данных (CSP) информирует проверяющую сторону (RP) о последнем событии аутентификации, чтобы RP могла определить, нужно ли ей повторно аутентифицировать пользователя.			✓	613	7.2.1

V3.7 Меры защиты от уязвимостей сессии

Существует несколько атак на управление сессиями, некоторые из которых связаны с пользовательским интерфейсом (UX). Ранее, в соответствии с требованиями ISO 27002, ASVS требовал блокировать параллельные сессии. Теперь такое решение неприменимо, не только потому, что у современных пользователей много устройств, или потому, что приложение может представлять собой API без сессий от браузера, но ещё и потому, что в большинстве этих реализаций действующим остается только последний аутентификатор, который чаще всего и предъявляет злоумышленник. В этом разделе представлены основные рекомендации по предотвращению, задержке и обнаружению атак на управление сессиями.

Описание «полуоткрытой» атаки

В начале 2018 года несколько финансовых учреждений были скомпрометированы с помощью так называемых «полуоткрытых атак». Полуоткрытая атака использует недостаток шаблона проектирования, часто встречающийся во многих системах аутентификации, управления сессиями и контроля доступа.

Злоумышленники начинают полуоткрытую атаку, пытаясь заблокировать, сбросить или восстановить учетные данные. В популярном шаблоне проектирования объекты профиля пользователя переиспользуются между неаутентифицированным, полуаутентифицированным (сброс или восстановление) и полностью аутентифицированным состояниями. Этот шаблон вставляет объект сессии или токен, содержащий профиль жертвы, включая хэш его пароля и имеющиеся роли. Если контроллер доступа или маршрутизатор не проверяют, что пользователь действительно полностью аутентифицирован, то злоумышленник сможет действовать от имени пользователя. Атаки могут включать изменение пароля, обновление адреса email, отключение многофакторной аутентификации или регистрацию нового устройства для МФА, раскрытие или изменение API-ключей и т.д.

№	Описание	L1	L2	L3	CWE	NIST
3.7.1	Убедитесь, что приложение обеспечивает полный, действительный сеанс входа в систему или требует повторной аутентификации или проверки второго фактора, прежде чем разрешать какие-либо конфиденциальные транзакции или изменения учетной записи.	✓	✓	✓	306	

Ссылки

Для дополнительной информации см. также:

- OWASP Testing Guide 4.0: Session Management Testing
- OWASP Session Management Cheat Sheet
- Set-Cookie __Host- prefix details

V4 Контроль доступа

Задачи контроля

Авторизация — это концепция предоставления доступа к ресурсам только тем, кому разрешено их использовать. Убедитесь, что исследуемое приложение удовлетворяет следующим концептуальным требованиям:

- Те, кто получает доступ к ресурсам, имеют действующие учётные записи.
- Пользователи имеют четко определенный набор ролей и привилегий.
- Метаданные ролей и разрешений защищены от воспроизведения или подделки.

Верификация требований к безопасности

V4.1 Базовые принципы

№	Описание	L1	L2	L3	CWE
4.1.1	Убедитесь, что приложение применяет правила контроля доступа на уровне доверенного сервиса, особенно если контроль доступа реализован на стороне клиента и может быть обойдён.	✓	✓	✓	602
4.1.2	Убедитесь, что конечные пользователи без специального на то разрешения не могут манипулировать атрибутами пользователей и данных, а также информацией о применяемой политике контроля доступа.	✓	✓	✓	639
4.1.3	Убедитесь, что применяется принцип наименьших привилегий, т.е. пользователи имеют доступ только к тем функциям, данным, URL-адресам, контроллерам, сервисам и прочим ресурсам, для которых им явно определены права. Это подразумевает защиту от спуфинга (подмены) и повышения привилегий. (C7)	✓	✓	✓	285
4.1.4	[УДАЛЕНО, ДУБЛИРУЕТ 4.1.3]				
4.1.5	Убедитесь, что механизмы управления доступом даже при сбоях ведут себя безопасным образом, в том числе при возникновении исключений. (C10)	✓	✓	✓	285

V4.2 Этап эксплуатации

№	Описание	L1	L2	L3	CWE
4.2.1	Убедитесь, что конфиденциальные данные и API защищены от атак с небезопасной прямой ссылкой на объект (IDOR), направленных на создание, чтение, обновление и удаление данных, например, регистрацию или изменение чужой учётной записи, просмотр или удаление всех учетных записей.	✓	✓	✓	639
4.2.2	Убедитесь, что приложение или платформа применяют надежный механизм защиты от CSRF-атак для неаутентифицированной зоны, а после аутентификации — защиту от автоматизированных атак и от CSRF.	✓	✓	✓	352

V4.3 Прочие соображения

№	Описание	L1	L2	L3	CWE
4.3.1	Убедитесь, что в интерфейсе администратора для предотвращения его несанкционированного использования используется многофакторная аутентификация.	✓	✓	✓	419
4.3.2	Убедитесь, что везде, где это не предусмотрено специально, отключен просмотр каталогов. Кроме того, приложения не должны разрешать обнаружение или раскрытие метаданных файлов или каталогов, таких как Thumbs.db, .DS_Store, .git или .svn.	✓	✓	✓	548
4.3.3	Убедитесь, что для обеспечения контроля над мошенничеством с учетом рисков и исторического опыта, в приложении применяются дополнительные меры контроля доступа (например, двухшаговая или адаптивная аутентификация) для менее значимых систем и/или разделение полномочий для более критичных приложений.		✓	✓	732

Ссылки

Для дополнительной информации см. также:

- OWASP Testing Guide 4.0: Authorization
- OWASP Cheat Sheet: Access Control
- OWASP CSRF Cheat Sheet
- OWASP REST Cheat Sheet

V5 Форматно-логический контроль, нейтрализация и кодировка

Задачи контроля

Наиболее распространенным недостатком безопасности web-приложений является неспособность должным образом проконтролировать входные данные, поступающие от клиента или среды, перед их непосредственным использованием без какой-либо кодировки. Это приводит почти ко всем существенным уязвимостям в web-приложениях, таким как межсайтовый скриптинг (XSS), инъекции SQL и команд, атаки, связанные с кодировкой, атаки на файловую систему и переполнение буфера.

Убедитесь, что исследуемое приложение удовлетворяет следующим концептуальным требованиям:

- Архитектура контроля входных и кодирования выходных данных имеет согласованный конвейер для предотвращения атак с использованием инъекций.
- Входные данные строго типизируются, контролируется их логическая непротиворечивость, проверяются допустимая размерность или диапазон значений, или, как минимум, они нейтрализуются или фильтруются.
- Выходные данные кодируются или экранируются в соответствии с контекстом данных как можно ближе к интерпретатору.

При современной архитектуре web-приложений кодирование выходных данных становится важным как никогда. В некоторых случаях сложно обеспечить надёжный контроль входных данных, поэтому использование более безопасных API, таких как параметризованные запросы, автоматически экранирующие шаблоны или тщательно подобранная кодировка для выходных данных, имеют решающее значение для безопасности приложения.

V5.1 Контроль входных данных

Правильно реализованные меры контроля входных данных, с использованием «белых» списков разрешений и строгой типизации данных, могут устранить более 90% атак с использованием инъекций. Контроль размера и диапазона значений могут еще сильнее снизить эти проблемы. Встраивание контроля входных данных необходимо на этапах проектирования архитектуры приложения, разработки, а также модульного и интеграционного тестирования. Хотя многих из этих требований и нет в тестах на проникновение, результаты их невыполнения обычно приводят к необходимости выполнения требований раздела V5.3 Кодирование выходной информации и предотвращение инъекций. Разработчикам и исследователям безопасности приложений рекомендуется относиться к этому разделу так, как если бы для всех пунктов требовался первый уровень (L1), чтобы предотвратить инъекции.

№	Описание	L1	L2	L3	CWE
5.1.1	Убедитесь, что в приложении применяются механизмы защиты от атак «загрязнения» параметров HTTP, особенно если платформа приложения не различает источник параметров запроса (GET, POST, файлы cookie, http-заголовки или переменные среды).	✓	✓	✓	235
5.1.2	Убедитесь, что фреймворки защищают от атак с массовым присвоением параметров, или что приложение обеспечивает меры защиты от небезопасного назначения параметров, например, помечая поля как private и т.п. (C5)	✓	✓	✓	915
5.1.3	Убедитесь, что все входные данные проходят форматно-логический контроль по «белому» списку разрешенных типов данных, значений атрибутов, параметров, заголовков, запросов, методов и т.п., включая поля HTML-формы, REST-запросы, параметры URL, HTTP-заголовки, атрибуты cookie, пакетные файлы, RSS-каналы и т.д. (C5)	✓	✓	✓	20

№	Описание	L1	L2	L3	CWE
5.1.4	Убедитесь, что структурированные данные строго типизированы и проверяются по заданной схеме, в которой указаны разрешенные символы, максимальная длина и шаблон данных (например, номера банковских карт, телефонов, адреса email. Проверяется, что связанные поля соответствуют друг другу, например, городу однозначно соответствует его почтовый индекс). (C5)	✓	✓	✓	20
5.1.5	Убедитесь, что перенаправление URL разрешено только по адресам из списка разрешенных. При перенаправлении на потенциально недоверенный адрес отображается предупреждение.	✓	✓	✓	601

V5.2 Нейтрализация и изоляция

№	Описание	L1	L2	L3	CWE
5.2.1	Убедитесь, что весь недоверенный HTML-код, из визуальных редакторов (WYSIWYG), должным образом нейтрализуется с помощью библиотеки санитизации или функционала HTML-фреймворка. (C5)	✓	✓	✓	116
5.2.2	Убедитесь, что неструктурированные данные нейтрализуются с применением таких мер, как разрешенные символы и длина строки.	✓	✓	✓	138
5.2.3	Убедитесь, что приложение нейтрализует пользовательский ввод перед передачей в почтовые системы для защиты от SMTP- или IMAP-инъекций.	✓	✓	✓	147
5.2.4	Убедитесь, что приложение избегает использования eval() или других функций динамического исполнения кода. Там, где нет альтернатив, пользовательский ввод должен нейтрализовываться или помещаться в песочницу перед исполнением.	✓	✓	✓	95
5.2.5	Убедитесь, что приложение защищено от атак с внедрением шаблонов, гарантируя, что весь пользовательский ввод нейтрализуется или исполняется в изолированной среде.	✓	✓	✓	94
5.2.6	Убедитесь, что приложение защищено от SSRF-атак, проверяя или нейтрализуя недоверенные данные или метаданные HTTP-запросов, такие как имена файлов, поля для ввода URL-адресов, а также применяет списки разрешенных протоколов, доменов, путей и портов.	✓	✓	✓	918
5.2.7	Убедитесь, что приложение нейтрализует, отключает или изолирует предоставляемый пользователем контент для сценариев масштабируемой векторной графики (SVG), особенно если они относятся к XSS, полученным из встроенных в код сценариев или foreignObject.	✓	✓	✓	159
5.2.8	Убедитесь, что приложение нейтрализует, отключает или помещает в песочницу предоставляемый пользователем контент на языках сценариев или шаблонов разметки, таких как Markdown, таблицы стилей CSS или XSL, BBCode и т.п.	✓	✓	✓	94

V5.3 Кодирование выходных данных и предотвращение инъекций

Кодирование выходных данных непосредственно перед используемым интерпретатором имеет решающее значение для безопасности приложения. Как правило, сама кодировка не хранится, а применяется для корректного отображения выходных данных в соответствующем контексте. Неспособность применить кодировку может привести к инъекциям и небезопасному приложению.

№	Описание	L1	L2	L3	CWE
5.3.1	Убедитесь, что кодировка выходных данных подходит для интерпретатора и контекста. Т.е. для значений и атрибутов HTML, JavaScript, параметров в URL, HTTP-заголовков, SMTP и др. кодировка определяется в зависимости от контекста, особенно при недоверенных входных данных, например, имен на Unicode или с апострофами, таких как ねこ или O'Xapa). (C4)	✓	✓	✓	116
5.3.2	Убедитесь, что кодировка выходных данных отображает выбранный пользователем набор символов и языковой стандарт, таким образом чтобы любой символ Unicode был допустимым и безопасно обрабатывался. (C4)	✓	✓	✓	176
5.3.3	Убедитесь, что контекстно-зависимое экранирование выходных данных защищает от XSS-атак (отражённых, сохраненных, основанных на DOM). Предпочтительно автоматизированное, или хотя бы ручное. (C4)	✓	✓	✓	79
5.3.4	Убедитесь, что запросы к базам данных (SQL, HQL, ORM, NoSQL и пр.) параметризованы посредством ORM, Entity Framework, или иным образом защищены от атак инъекции в базу данных. (C3)	✓	✓	✓	89
5.3.5	Убедитесь, что для защиты от SQL-инъекций там, где параметризованные или более безопасные механизмы отсутствуют, при выводе данных используется контекстно-зависимое кодирование, например, экранирование SQL. (C3, C4)	✓	✓	✓	89
5.3.6	Убедитесь, что приложение защищено от JSON-инъекции, атак JSON eval и интерпретации кода на JavaScript. (C4)	✓	✓	✓	830
5.3.7	Убедитесь, что приложение защищено от LDAP-инъекций или что были реализованы меры безопасности для предотвращения LDAP-инъекций. (C4)	✓	✓	✓	90
5.3.8	Убедитесь, что приложение защищено от инъекций команд операционной системы, и что для вызова команд (и передачи в них аргументов) используются параметризованные запросы к ОС или контекстное кодирование при выводе командной строки. (C4)	✓	✓	✓	78
5.3.9	Убедитесь, что приложение защищено от атак с локальным включением файлов (LFI) или удаленным включением файлов (RFI).	✓	✓	✓	829
5.3.10	Убедитесь, что приложение защищено от атак с использованием XPath- или XML-инъекций. (C4)	✓	✓	✓	643

Примечание: Использования параметризованных запросов или экранирования SQL не всегда достаточно. Наименования таблиц и столбцов, ORDER BY и т.д. экранировать не получится. Включение экранированных пользовательских данных в эти поля приводит к неудачным запросам или SQL-инъекции.

Примечание. Формат SVG явно разрешает ECMAScript почти во всех контекстах, поэтому полностью заблокировать все XSS-векторы в SVG может оказаться невозможным. Если требуется загрузка SVG, мы настоятельно рекомендуем загружать файлы либо как text/plain, либо со своего домена, чтобы предотвратить XSS-атаку на приложение.

V5.4 Память, строки и неуправляемый код

Следующие требования будут актуальны, если приложение использует язык ассемблера или неуправляемый код.

№	Описание	L1	L2	L3	CWE
5.4.1	Убедитесь, что для обнаружения или предотвращения ситуаций переполнения стека, буфера или кучи приложение учитывает размер доступной области памяти при строковых операциях, копировании в память, в арифметике указателей и т.п..		✓	✓	120
5.4.2	Убедитесь, что строка форматирования не содержит потенциально вредоносных входных данных и является неизменяемой.		✓	✓	134
5.4.3	Убедитесь, что для предотвращения целочисленного переполнения применяется контроль знака, диапазона и типа допустимых входных данных.		✓	✓	190

V5.5 Предотвращение десериализации

№	Описание	L1	L2	L3	CWE
5.5.1	Убедитесь, что для предотвращения создания вредоносных объектов или подмены данных используется контроль целостности и/или шифрование. (C5)	✓	✓	✓	502
5.5.2	Убедитесь, что приложение правильно ограничивает XML-парсер наиболее строгой конфигурацией, которая гарантирует, что небезопасные XML-функции, такие как разрешение внешних объектов, отключены для предотвращения атак XML eXternal Entity (XXE).	✓	✓	✓	611
5.5.3	Убедитесь, что десериализация недоверенных данных исключена или изолирована как в пользовательском коде, так и в сторонних библиотеках (например, в парсерах JSON, XML, YAML и пр.).	✓	✓	✓	502
5.5.4	Убедитесь, что при разборе JSON в браузерах или в серверной части на основе JavaScript для документа JSON используется JSON.parse. Не используйте eval() для разбора JSON.	✓	✓	✓	95

Ссылки

Для дополнительной информации см. также:

- OWASP Testing Guide 4.0: Input Validation Testing
- OWASP Cheat Sheet: Input Validation
- OWASP Testing Guide 4.0: Testing for HTTP Parameter Pollution
- OWASP LDAP Injection Cheat Sheet
- OWASP Testing Guide 4.0: Client Side Testing
- OWASP Cross Site Scripting Prevention Cheat Sheet
- OWASP DOM Based Cross Site Scripting Prevention Cheat Sheet
- OWASP Java Encoding Project
- OWASP Mass Assignment Prevention Cheat Sheet
- DOMPurify - Client-side HTML Sanitization Library

- XML External Entity (XXE) Prevention Cheat Sheet

Для дополнительной информации по экранированию см. также:

- Reducing XSS by way of Automatic Context-Aware Escaping in Template Systems
- AngularJS Strict Contextual Escaping
- AngularJS ngBind
- Angular Sanitization
- Angular Security
- ReactJS Escaping
- Improperly Controlled Modification of Dynamically-Determined Object Attributes

Для дополнительной информации по десериализации см. также:

- OWASP Deserialization Cheat Sheet
- OWASP Deserialization of Untrusted Data Guide

V6 Хранимая криптография

Задачи контроля

Убедитесь, что исследуемое приложение удовлетворяет следующим концептуальным требованиям:

- Криптографические модули выходят из строя безопасным образом, и ошибки корректно обрабатываются.
- Используется соответствующий генератор случайных чисел.
- Обеспечивается контроль доступа к ключам.

V6.1 Категории информации

Самым важным активом является информация, обрабатываемая, хранимая или передаваемая приложением. Проводите оценку воздействия на ее конфиденциальность, чтобы правильно классифицировать потребности в защите информации при хранении.

№	Описание	L1	L2	L3	CWE
6.1.1	Убедитесь, что в зашифрованном виде хранятся регулируемые законом персональные данные, или данные, которые могут подпадать под действие ФЗ-152, GDPR и т.п.		✓	✓	311
6.1.2	Убедитесь, что в зашифрованном виде хранятся регулируемые законом данные о здоровье, например медицинские карты, результаты анализов, диагностические данные с медицинского оборудования или необезличенные результаты клинических исследований.		✓	✓	311
6.1.3	Убедитесь, что в зашифрованном виде хранится регулируемая законом финансовая информация, например, данные о банковских картах и счетах, о банкротстве, кредитная история, налоговые декларации, история платежей, выгодоприобретатели или необезличенные данные исследований рынка.		✓	✓	311

V6.2 Алгоритмы

Недавние достижения в области криптографии означают, что ранее считавшиеся стойкими алгоритмы и достаточными длины ключей больше не являются таковыми для защиты данных. Следовательно, должна быть возможность поменять алгоритмы.

Хотя этот раздел нелегко протестировать на проникновение, разработчики должны рассматривать весь этот раздел как обязательный, даже если пометка в L1 отсутствует.

№	Описание	L1	L2	L3	CWE
6.2.1	Убедитесь, что криптографические модули при сбоях ведут себя безопасно, а ошибки обрабатываются таким образом, чтобы не допустить атаки Padding Oracle.	✓	✓	✓	310
6.2.2	Убедитесь, что используются утвержденные государственные или отраслевые стандарты на криптографические алгоритмы, режимы и библиотеки, а не криптография собственной / заказной разработки. (C8)		✓	✓	327
6.2.3	Убедитесь, что вектор инициализации, конфигурация шифра и блочные режимы настроены безопасно, используя актуальные рекомендации.		✓	✓	326

№	Описание	L1	L2	L3	CWE
6.2.4	Убедитесь, что для защиты от криптографических атак в любой момент можно перенастроить, обновить или изменить параметры алгоритмов (генерации случайных чисел, шифрования, хэширования), а также длины ключей, раунды, шифры или режимы. (C8)		✓	✓	326
6.2.5	Убедитесь, что не используются небезопасные блочные режимы (например, ECB и т.д.), режимы дополнения (например, PKCS#1 v1.5 и т.д.), шифры с небольшими размерами блоков (например, 3DES, Blowfish и т.д.) и нестойкие алгоритмы хеширования (например, MD5, SHA1 и т. д.), если это не требуется для обратной совместимости.		✓	✓	326
6.2.6	Убедитесь, что значения nonce, векторов инициализации и других случайных одноразовых строк не должны использоваться более одного раза с данным ключом шифрования. Метод их генерации должен соответствовать используемому алгоритму.		✓	✓	326
6.2.7	Убедитесь, что зашифрованные данные аутентифицированы с помощью подписей, аутентифицированных режимов шифрования или HMAC, чтобы гарантировать, что зашифрованный текст не будет изменен неуполномоченной на то стороной.			✓	326
6.2.8	Убедитесь, что все криптографические операции выполняются за постоянное время, без операций «короткого замыкания» при сравнении, вычислениях или возврате, чтобы избежать утечки информации по побочным каналам.			✓	385

V6.3 Случайные значения

Настоящий генератор псевдослучайных чисел (PRNG) невероятно сложно сделать правильно. Обычно, хорошие источники энтропии в системе быстро истощаются при чрезмерном использовании, 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 утвержденного криптографического генератора случайных чисел в криптографическом модуле.		✓	✓	338
6.3.2	Убедитесь, что случайные идентификаторы GUID создаются с использованием алгоритма GUIDv4 и криптографического генератора псевдослучайных чисел (CSPRNG). 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, и что при сбоях оно ведет себя безопасно.			✓	338

V6.4 Управление секретами

Хотя этот раздел нелегко протестировать на проникновение, разработчики должны рассматривать весь этот раздел как обязательный, даже если пометка в L1 отсутствует.

№	Описание	L1	L2	L3	CWE
6.4.1	Убедитесь, что для безопасного создания, хранения, контроля доступа и уничтожения секретов используется решение для управления секретами, например, сейф — хранилище ключей (key vault). (C8)		✓	✓	798
6.4.2	Убедитесь, что ключевой материал не доступен приложению, и для криптографических операций применяется изолированный модуль безопасности, например, сейф (vault). (C8)		✓	✓	320

Ссылки

Для дополнительной информации см. также:

- OWASP Testing Guide 4.0: Testing for weak Cryptography
- OWASP Cheat Sheet: Cryptographic Storage
- FIPS 140-2

V7 Обработка ошибок и ведение журнала

Задачи контроля

Основная цель обработки ошибок и ведения журнала — дать полезную информацию пользователям, администраторам и группам реагирования на инциденты. Цель журнала не в том, чтобы регистрировать в нем как можно больше записей, а в полезности каждой отдельной записи, и в улучшении отношения сигнал/шум.

Качественные журналы часто содержат конфиденциальную информацию и должны быть защищены в соответствии с местными законами о конфиденциальности. Меры защиты журнала должны включать:

- Не собирать и не регистрировать в журналах конфиденциальную информацию, если этого не требуется.
- Обеспечивать безопасную обработку и защиту регистрируемой в журналах информации в соответствии с ее категорией.
- Убедиться, что журналы не хранятся вечно, и имеют как можно более короткий срок хранения.

Если журналы содержат персональные или другие чувствительные данные, определение которых варьируется от страны к стране, то они становятся одной из наиболее привлекательных целей для злоумышленников в приложении.

Также важно убедиться, что и при сбоях приложение ведет себя безопасно, не раскрывая избыточной информации в тексте ошибок.

V7.1 Содержимое журнала

Регистрация конфиденциальной информации в журнале опасна — журналы сами по себе становятся засекреченными, а это означает, что они должны быть зашифрованы, на них распространяются политики хранения и их необходимо раскрывать в ходе аудитов безопасности. Убедитесь, что в журналах хранится только необходимая информация, и, конечно, никаких платежных, учетных (включая сессионные токены), конфиденциальных или персональных данных.

V7.1 покрывает OWASP Top 10 2017:A10. Поскольку 2017:A10 не поддаётся тестированию на проникновение, то необходимо:

- разработчикам — обеспечивать полное соответствие этому разделу, как если бы все требования были помечены как L1;
- пентестерам — обеспечивать контроль соответствия всем требованиям V7.1 с помощью интервью, снимков экрана и проверки утверждений.

№	Описание	L1	L2	L3	CWE
7.1.1	Убедитесь, что приложение не записывает в журнал учетные или платежные данные. Сессионные токены могут записываться в журнал только в необратимой хэшированной форме. (C9, C10)	✓	✓	✓	532
7.1.2	Убедитесь, что приложение не записывает в журнал другие конфиденциальные данные в соответствии с применимыми законами (ПДн, банковская, налоговая, коммерческая, врачебная и иные виды тайн) или действующей политике безопасности. (C9)	✓	✓	✓	532
7.1.3	Убедитесь, что приложение регистрирует в журнале события, относящиеся к безопасности, включая успешную и неудачную аутентификацию, события нарушения контроля доступа, десериализации и форматно-логического контроля входных данных. (C5, C7)		✓	✓	778

№	Описание	L1	L2	L3	CWE
7.1.4	Убедитесь, что каждое событие журнала включает необходимую и достаточную информацию, которая позволит провести его подробное исследование с учетом хронологических связей. (C9)		✓	✓	778

V7.2 Обработка журнала

Своевременная регистрация событий имеет решающее значение для их аудита, сортировки и эскалации. Убедитесь, что журналы приложения понятны и могут анализироваться либо локально, либо в удаленной системе мониторинга.

V7.2 покрывает OWASP Top 10 2017:A10. Поскольку 2017:A10 не поддаётся тестированию на проникновение, то необходимо:

- разработчикам — обеспечивать полное соответствие этому разделу, как если бы все требования были помечены как L1;
- пентестерам — обеспечивать контроль соответствия всем требованиям V7.2 с помощью интервью, снимков экрана и проверки утверждений.

№	Описание	L1	L2	L3	CWE
7.2.1	Убедитесь, что все решения по аутентификации регистрируются в журнале, без сохранения сессионных токенов, паролей и т.п., включая запросы с соответствующими метаданными, необходимыми для расследований событий безопасности.		✓	✓	778
7.2.2	Убедитесь, что все решения по контролю доступа МОГУТ, а все неудачные попытки ДОЛЖНЫ быть занесены в журнал, включая запросы с соответствующими метаданными, необходимыми для расследований событий безопасности.		✓	✓	285

V7.3 Защита журнала

Журналы, которые можно тривиально изменить или удалить, бесполезны для расследований и криминалистики. Раскрытие журнала может привести к разглашению внутренних сведений о приложении или содержащейся в нем информации. Необходимо соблюдать осторожность при защите журналов от несанкционированного раскрытия, изменения или удаления.

№	Описание	L1	L2	L3	CWE
7.3.1	Убедитесь, что все компоненты записывают события в журнал в нормализованной форме, указывая правильную кодировку, и используя экранирование символов, чтобы предотвратить инъекцию в журнал. (C9)		✓	✓	117
7.3.2	[УДАЛЕНО, ДУБЛИРУЕТ 7.3.1]				
7.3.3	Убедитесь, что журналы событий безопасности защищены от несанкционированного доступа и изменения. (C9)		✓	✓	200
7.3.4	Убедитесь, что системное время синхронизируется с утвержденными источниками времени и в правильном часовом поясе. Для географически распределенных систем настоятельно рекомендуется вести журнал в формате UTC, чтобы облегчить криминалистический анализ после инцидента. (C9)		✓	✓	

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

V7.4 Обработка ошибок

Цель обработки ошибок — предоставлять важные для безопасности приложения события для мониторинга, сортировки и эскалации. Смысл не в том, чтобы создавать много записей. При регистрации событий, связанных с безопасностью, убедитесь, что в каждой записи есть смысл, который понятен SIEM или аналитической системе.

№	Описание	L1	L2	L3	CWE
7.4.1	Убедитесь, что при возникновении непредвиденной ошибки или ошибки, связанной с безопасностью, отображается универсальное сообщение с уникальным идентификатором, который персонал службы сопровождения может использовать для расследования. (C10)	✓	✓	✓	210
7.4.2	Убедитесь, что для учета ожидаемых и непредвиденных ошибок в кодовой базе используется обработка исключений (или ее функциональный эквивалент). (C10)		✓	✓	544
7.4.3	Убедитесь, что определен обработчик ошибок «последней инстанции», который будет перехватывать все необработанные исключения. (C10)		✓	✓	431

Примечание. Некоторые языки, такие как Swift и Go, а также многие функциональные языки не поддерживают исключения или обработчики событий последней инстанции. В этом случае архитекторы и разработчики должны использовать такой шаблон, язык или фреймворк, которые обеспечат безопасную обработку исключений, а также неожиданных или связанных с безопасностью событий.

Ссылки

Для дополнительной информации см. также:

- OWASP Testing Guide 4.0: Testing for Error Handling
- OWASP Authentication Cheat Sheet

V8 Защита информации

Задачи контроля

Есть три ключевых элемента надежной защиты данных: конфиденциальность, целостность и доступность. В данном стандарте предполагается, что защита данных обеспечивается в доверенной системе, например, на сервере, укрепленном достаточным набором мер защиты.

Приложение должно считать, что все пользовательские устройства так или иначе скомпрометированы. Если приложение передает или хранит конфиденциальную информацию на небезопасных устройствах, таких как компьютеры, телефоны и планшеты, приложение несет ответственность за то, чтобы информация, хранящаяся на этих устройствах, была зашифрована и не могла быть легко получена незаконным путем, изменена или разглашена.

Убедитесь, что исследуемое приложение удовлетворяет следующим концептуальным требованиям к защите информации:

- Конфиденциальность: информация должна быть защищена от несанкционированного просмотра или разглашения как при передаче, так и при хранении.
- Целостность: информация должна быть защищена от несанкционированной регистрации, изменения или удаления неуполномоченными лицами.
- Доступность: информация должна быть доступна авторизованным пользователям по мере необходимости.

V8.1 Базовые меры

№	Описание	L1	L2	L3	CWE
8.1.1	Убедитесь, что приложение защищает конфиденциальные данные от кэширования в серверных компонентах, таких как балансировщики нагрузки и кэши приложений.		✓	✓	524
8.1.2	Убедитесь, что все кэшированные или временные копии конфиденциальных данных, хранящиеся на сервере, защищены от несанкционированного доступа или очищены/аннулированы 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 данному IP-адресу или пользователю, по общему количеству запросов в час или в день или по другому признаку в контексте приложения.		✓	✓	770
8.1.5	Убедитесь, что регулярно создаются резервные копии всех значимых данных и проводится тестирование по восстановлению этих данных.			✓	19
8.1.6	Убедитесь, что резервные копии хранятся в безопасном месте, 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 соответствующие заголовки для предотвращения кэширования, чтобы конфиденциальные данные не кэшировались в современных браузерах.	✓	✓	✓	525
8.2.2	Убедитесь, что данные, хранящиеся в хранилище браузера (например, localStorage, sessionStorage, IndexedDB или файлы cookie), не содержат конфиденциальной информации.	✓	✓	✓	922
8.2.3	Убедитесь, что данные, полученные после аутентификации клиента, удаляются из хранилища, например из DOM браузера, после завершения работы клиента или сессии.	✓	✓	✓	922

V8.3 Персональные данные

Этот раздел поможет защитить конфиденциальную информацию от её несанкционированной регистрации, чтения, изменения или удаления, особенно большими партиями.

Соблюдение требований данного раздела также подразумевает удовлетворение требованиям раздела V4 Контроля доступа, особенно в части V4.2. Например, для защиты от несанкционированного изменения или разглашения персональных данных требуется соответствие требованию V4.2.1. Для полного охвата необходимо соответствие и этому разделу и V4.

Примечание: Законы и регламенты о персональных данных, такие как Федеральный закон РФ "О персональных данных" 152-ФЗ или GDPR, напрямую влияют на то, как приложения должны подходить к хранению, использованию и передаче персональных данных. Наказания варьируются от суровых штрафов до простых советов. Ознакомьтесь с местными законами и нормативными актами, а также при необходимости проконсультируйтесь с квалифицированным специалистом по вопросам персональных данных или юристом.

№	Описание	L1	L2	L3	CWE
8.3.1	Убедитесь, что в параметрах строки HTTP-запроса не содержится конфиденциальных данных. Вместо этого они отправляются на сервер в теле или заголовках HTTP-сообщения.	✓	✓	✓	319
8.3.2	Убедитесь, что у пользователей есть способ удалить или экспортировать свои данные по запросу.	✓	✓	✓	212
8.3.3	Убедитесь, что пользователям предоставлены четкие формулировки в отношении сбора и использования их персональных данных, и что они дали добровольное согласие на обработку этих данных, прежде чем эти данные будут каким-либо образом использованы.	✓	✓	✓	285
8.3.4	Убедитесь, что определены все конфиденциальные данные, создающиеся и обрабатываемые приложением, и что на эти данные распространяется действие политики обращения с конфиденциальными данными. (C8)	✓	✓	✓	200
8.3.5	Убедитесь, что доступ к конфиденциальным данным подвергается аудиту (без регистрации в журнале самих этих данных), если данные собираются в соответствии с соответствующими законами о защите данных или если вести журнал доступа требуется по стандарту организации (отрасли).		✓	✓	532

№	Описание	L1	L2	L3	CWE
8.3.6	Убедитесь, что для предотвращения атак с дампом памяти, конфиденциальная информация, содержащаяся в памяти, после того как она больше не нужна, перезаписывается (нулями или случайными данными).		✓	✓	226
8.3.7	Убедитесь, что конфиденциальная информация, которую необходимо зашифровать, зашифрована с использованием утвержденных алгоритмов, обеспечивающих как конфиденциальность, так и целостность. (C8)		✓	✓	327
8.3.8	Убедитесь, что конфиденциальная информация подлежит классификации по сроку хранения, чтобы ненужные или устаревшие данные удалялись автоматически, по расписанию или по мере необходимости.		✓	✓	285

При рассмотрении вопроса о защите данных основное внимание следует уделять их массовому извлечению, изменению или чрезмерному использованию. Например, многие социальные сети позволяют пользователям добавлять не более 100 новых «друзей» в день. Банковская система может заблокировать переводы во внешние учреждения, если их частота более пяти в час, и сумма более $1000. Требования к каждой системе могут быть разными, поэтому, отвечая на вопрос "Что считать «аномалией»?", необходимо учитывать модель угроз и бизнес-риски. Важными критериями являются способность обнаруживать, сдерживать, и, что ещё важнее, блокировать такие аномальные массовые операции.

Ссылки

Для дополнительной информации см. также:

- Сайт Security Headers
- Проект OWASP Secure Headers
- Проект OWASP Privacy Risks
- OWASP User Privacy Protection Cheat Sheet
- Обзор European Union General Data Protection Regulation (GDPR)
- European Union Data Protection Supervisor - Internet Privacy Engineering Network

V9 Передача данных

Задачи контроля

Убедитесь, что исследуемое приложение соответствует следующим концептуальным требованиям:

- Требует TLS или шифрования, независимо от конфиденциальности обрабатываемой информации.
- Следует последним рекомендациям в части:
  - безопасных конфигураций;
  - предпочтительных алгоритмов и шифров.
- Избегает нестойких или устаревающих алгоритмов и шифров, за исключением крайних случаев.
- Отключает небезопасные алгоритмы и шифры.

В рамках этих требований:

- Будьте в курсе рекомендаций по безопасной конфигурации TLS, поскольку они часто меняется («часто» из-за катастрофических проблем, обнаруживаемых в существующих алгоритмах и шифрах).
- Используйте самые последние версии инструментов анализа конфигурации TLS, чтобы настроить предпочтительный порядок и выбор алгоритмов.
- Периодически проверяйте конфигурацию, чтобы убедиться, что данные передаются надёжно и безопасно.

V9.1 Безопасность подключений со стороны клиента

Убедитесь, что сообщения клиента передаются по зашифрованным каналам с использованием TLS 1.2 или более поздней версии. Используйте современные инструменты для регулярной проверки конфигурации клиента.

№	Описание	L1	L2	L3	CWE
9.1.1	Убедитесь, что для клиентских подключений используется протокол TLS, и при этом он не оставляет соединения без защиты (незашифрованными или неаутентифицированными). (C8)	✓	✓	✓	319
9.1.2	С помощью современных инструментов тестирования TLS убедитесь, что включены только стойкие шифронаборы, при этом наиболее стойкие из них выбраны как предпочтительные.	✓	✓	✓	326
9.1.3	Убедитесь, что включены только актуальные и рекомендуемые версии протокола TLS, такие как TLS 1.2 и TLS 1.3. Наиболее свежая версия TLS должна быть выбрана как предпочтительная.	✓	✓	✓	326

V9.2 Безопасность подключений со стороны сервера

Связь с сервером — это не только HTTP. Также необходимо обеспечить безопасные соединения с другими системами, такими как системы мониторинга, инструменты управления, удаленный доступ и ssh, промежуточное ПО, базы данных, серверы, партнерские или внешние API. Все эти соединения должны быть зашифрованы, чтобы не допустить перехвата информации по принципу «трудно снаружи, но легко внутри».

№	Описание	L1	L2	L3	CWE
9.2.1	Убедитесь, что для подключения к и от сервера используются доверенные сертификаты TLS. Если используются сгенерированные собственным УЦ или самоподписанные сертификаты, то сервер должен быть настроен так, чтобы доверять только утвержденным внутренним центрам сертификации и только указанным самоподписанным сертификатам. Все остальные должны отвергаться.		✓	✓	295
9.2.2	Убедитесь, что для всех входящих и исходящих подключений используются зашифрованные соединения, такие как TLS, в том числе для портов управления, мониторинга, аутентификации, вызовов API или web-сервисов, подключений к базам данных, облачным сервисам, бессерверным функциям, серверам, внешних и партнерских подключений. Сервер не должен возвращаться к небезопасным или незашифрованным протоколам.		✓	✓	319
9.2.3	Убедитесь, что все зашифрованные соединения с внешними системами, передающие конфиденциальную информацию или функции, аутентифицированы.		✓	✓	287
9.2.4	Убедитесь, что включена и настроена процедура отзыва сертификатов, например, Online Certificate Status Protocol (OCSP) Stapling.		✓	✓	299
9.2.5	Убедитесь, что сбои TLS-соединений со стороны сервера регистрируются в журнале.			✓	544

Ссылки

Для дополнительной информации см. также:

- OWASP – TLS Cheat Sheet
- OWASP - Pinning Guide
- Примечания по рекомендованным режимам TLS:
  - В прошлом ASVS ссылался на стандарт США FIPS 140-2, но рекомендация о применении стандартов США в качестве международных может оказаться трудной в реализации, противоречивой и нелогичной.
  - Лучшим способом достижения соответствия разделу 9.1 будет рассмотрение таких руководств, как Mozilla's Server Side TLS или generate known good configurations, и использование известных и современных инструменты анализа TLS для достижения желаемого уровня безопасности.

V10 Вредоносный код

Задачи контроля

Убедитесь, что ваш код удовлетворяет следующим концептуальным требованиям:

- Вредоносная активность надёжно и безопасно сдерживается, чтобы не повлиять на остальную часть приложения.
- Не имеет «бомб отложенного действия» или других атак, основанных на времени.
- Не «звонит домой» злонамеренным или неразрешённым адресатам.
- В нём нет лазеек, «пасхальных яиц», атак «салями», руткитов или несанкционированного кода, которым может управлять злоумышленник.

Обнаружение вредоносного кода является «доказательством от противного», в котором нельзя удостовериться полностью. Следует приложить все силы для обеспечения гарантии того, что ваш код не содержит встроенного вредоносного кода или недокументированных возможностей.

V10.1 Целостность исходного кода

Лучшая защита от вредоносного кода — принцип «доверяй, но проверяй». Включение недокументированного или вредоносного кода во многих юрисдикциях является уголовным преступлением. Регламенты и процедуры должны четко определять санкции в отношении вредоносного кода.

Ведущие разработчики должны регулярно проводить анализ регистрации нового кода, особенно там, где есть доступ к системному времени, вводу-выводу или сетевым функциям.

№	Описание	L1	L2	L3	CWE
10.1.1	Убедитесь, что используется инструмент анализа исходного кода, который может обнаруживать потенциально вредоносный код, например, манипуляции с системным временем, с файлами и сетевыми подключениями.			✓	749

V10.2 Поиск вредоносного кода

Вредоносный код встречается крайне редко, и его трудно обнаружить. Построчный просмотр кода вручную может помочь в поиске логических бомб, но даже самый опытный рецензент едва ли найдет вредоносный код, даже если знает, что он есть.

Соблюдение требований этого раздела невозможно без полного доступа к исходному коду, включая сторонние библиотеки.

№	Описание	L1	L2	L3	CWE
10.2.1	Убедитесь, что исходный код приложения и сторонние библиотеки не содержат недокументированных возможностей по сбору и отправке данных пользователя неразрешённым адресатам. Если такие функции существуют, необходимо получить разрешение на эти действия и явное согласие пользователя, прежде чем собирать какие-либо данные.		✓	✓	359
10.2.2	Убедитесь, что приложение не запрашивает ненужных или избыточных разрешений для функций, связанных с конфиденциальностью, например, доступ к контактам, сообщениям, истории просмотров и пр. или к датчикам, таким как камера, микрофон или местоположение.		✓	✓	272

№	Описание	L1	L2	L3	CWE
10.2.3	Убедитесь, что исходный код приложения и сторонних библиотек не содержит лазеек, таких как включенные в код и/или недокументированные учетные записи или ключи, обфускация кода, недокументированные бинарные объекты (blob), руткиты или средства защиты от отладки, или иные устаревшие, небезопасные или скрытые функции, которые могут быть использованы злонамеренно.			✓	507
10.2.4	Убедитесь, что исходный код приложения и сторонние библиотеки не содержат «бомб отложенного действия», выполнив поиск функций, связанных с датой и временем.			✓	511
10.2.5	Убедитесь, что исходный код приложения и сторонние библиотеки не содержат вредоносного кода, такого как атаки типа «нарезка салями», обход логики или «логические бомбы».			✓	511
10.2.6	Убедитесь, что исходный код приложения и сторонние библиотеки не содержат «пасхальных яиц» и других недокументированных функций.			✓	507

V10.3 Целостность приложения

Вредоносный код может быть включен и после развертывания приложения. Приложения должны защищать себя от распространенных атак, таких как выполнение неподписанного кода из недоверенных источников и захват доменов.

Соблюдение требований этого раздела, вероятно, будет регулярным занятием во время эксплуатации приложений.

№	Описание	L1	L2	L3	CWE
10.3.1	Убедитесь, что если в приложении есть функция автоматического обновления клиента или сервера, то обновления должны быть получены по защищенным каналам и подписаны электронной подписью. Код обновления должен проверять подпись обновления до установки или применения обновления.	✓	✓	✓	16
10.3.2	Убедитесь, что приложение использует меры защиты целостности, такие как подпись кода или включаемых ресурсов (subresource integrity). Приложение не должно загружать или исполнять код из недоверенных источников, таких как загрузка компонентов, модулей, плагинов, кода или библиотек из Интернета.	✓	✓	✓	353
10.3.3	Убедитесь, что если приложение полагается на записи DNS или субдомены DNS, то оно контролирует возможный захват субдоменов злоумышленниками, например, после истечения срока делегирования доменных имен, устаревании указателей DNS или CNAME, в необновляемых проектах, в общедоступных репозиториях исходного кода или во временных облачных API, бессерверных функциях, или хранилищах (*autogen-bucket-id*.cloud.example.com и т.п.). Меры защиты включают регулярную проверку DNS-имен, используемых приложениями, на предмет истечения срока их делегирования или изменения владельца.	✓	✓	✓	350

Ссылки

- Враждебный захват субдомена, Detectify Labs
- Угон заброшенных поддоменов, часть 2, Detectify Labs

V11 Бизнес-логика

Задачи контроля

Убедитесь, что исследуемое приложение удовлетворяет следующим концептуальным требованиям:

- Поток бизнес-логики является последовательным, обрабатывается по порядку, и его нельзя обойти.
- Бизнес-логика включает ограничения для обнаружения и предотвращения автоматизированных атак, таких как многократно повторяющиеся денежные переводы незначительных сумм или добавление миллиона друзей по одному за раз и т.д.
- Ключевые потоки бизнес-логики учитывают возможность злоупотреблений и злонамеренных действий и имеют защиту от несанкционированного доступа, разглашения и искажения информации и атак с повышением привилегий.

V11.1 Безопасность бизнес-логики

Безопасность бизнес-логики настолько индивидуальна для каждого приложения, что не поможет ни один чек-лист. Она должна проектироваться для защиты от вероятных внешних угроз, но её нельзя «включить» как набор правил на WAF или, организовав защищенный канал связи. Мы рекомендуем использовать моделирование угроз во время спринтов по дизайну, например, с помощью OWASP Cornucopia или аналогичных инструментов.

№	Описание	L1	L2	L3	CWE
11.1.1	Убедитесь, что приложение обрабатывает потоки бизнес-логики для того же пользователя, последовательно, и без пропуска шагов.	✓	✓	✓	841
11.1.2	Убедитесь, что приложение обрабатывает потоки бизнес-логики только в том случае, если все шаги обрабатываются в реалистичном для человека масштабе времени, т.е. транзакции не делаются слишком быстро.	✓	✓	✓	799
11.1.3	Убедитесь, что приложение ограничивает определенные операции бизнес-процессов или транзакции по частоте и потребляемым ресурсам, и эти ограничения корректно применяются для каждого пользователя.	✓	✓	✓	770
11.1.4	Убедитесь, что в приложении предусмотрены меры защиты от автоматизации аномальных по размеру запросов на массовую выгрузку данных, запросов бизнес-логики, загрузки файлов или атак типа «отказ в обслуживании».	✓	✓	✓	770
11.1.5	Убедитесь, что приложение имеет ограничения бизнес-логики или меры защиты от наиболее вероятных бизнес-рисков или угроз, выявленных с помощью моделирования угроз или аналогичных методологий.	✓	✓	✓	841
11.1.6	Убедитесь, что приложение не подвержено атакам типа «момент проверки до момента использования» (TOCTOU) или других условий гонки 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 совершил. (C9)		✓	✓	754
11.1.8	Убедитесь, что приложение имеет настраиваемое оповещение при обнаружении автоматизированных атак или необычных действий.		✓	✓	390

Ссылки

Для дополнительной информации см. также:

- OWASP Web Security Testing Guide 4.1: Business Logic Testing
- "Анти-автоматизация" атак может быть достигнута многими способами, включая OWASP AppSensor и OWASP Automated Threats to Web Applications
- OWASP AppSensor также может помочь с обнаружением и реагированием на атаки.
- OWASP Cornucopia

V12 Файлы и ресурсы

Задачи контроля

Убедитесь, что исследуемое приложение удовлетворяет следующим концептуальным требованиям:

- Недоверенные данные файлов должны обрабатываться соответствующим и безопасным образом.
- Недоверенные данные файлов, полученные из недоверенных источников, хранятся вне корневого каталога webroot и с ограниченными разрешениями.

V12.1 Загрузка файлов на сервер

Хотя zip-бомбы хорошо поддаются тестированию методами пентеста, они относятся к уровню L2 и выше, чтобы поощрять проектирование и разработку с тщательным ручным тестированием и избегать автоматизированного или неквалифицированного ручного тестирования на выявление ситуации отказа в обслуживании.

№	Описание	L1	L2	L3	CWE
12.1.1	Убедитесь, что приложение не будет принимать файлы, размер которых превышает доступное место или вызывает отказ в обслуживании.	✓	✓	✓	400
12.1.2	Убедитесь, что приложение проверяет сжатые файлы (например, zip, gz, docx, odt) на соответствие максимально допустимому несжатому размеру и максимальному количеству файлов перед распаковкой файла.		✓	✓	409
12.1.3	Убедитесь, что применяются квоты на размер файла и на максимальное количество файлов на одного пользователя, чтобы гарантировать, что пользователь не может заполнить все хранилище слишком большим количеством файлов или файлами чрезмерно большого размера.		✓	✓	770

V12.2 Целостность файлов

№	Описание	L1	L2	L3	CWE
12.2.1	Убедитесь, что файлы, полученные из недоверенных источников, соответствуют ожидаемому типу на основе содержимого файла (а не только его наименования).		✓	✓	434

V12.3 Исполнение файла

№	Описание	L1	L2	L3	CWE
12.3.1	Убедитесь, что метаданные имени файла, отправляемые пользователем, не используются напрямую файловыми системами ОС или платформы. Вместо этого для защиты от атак path traversal (перемещения по каталогам) используется API URL.	✓	✓	✓	22
12.3.2	Убедитесь, что передаваемые пользователем метаданные файла проверяются на предмет раскрытия, создания, изменения или удаления локальных файлов (LFI) или игнорируются.	✓	✓	✓	73
12.3.3	Убедитесь, что передаваемые пользователем метаданные файла проверяются на предмет раскрытия или исполнения файлов посредством удаленного включения (RFI) или подделки запросов на стороне сервера (SSRF), или игнорируются.	✓	✓	✓	98

№	Описание	L1	L2	L3	CWE
12.3.4	Убедитесь, что приложение защищено от отраженной загрузки файлов (Reflected File Download, RFD), проверяя или игнорируя передаваемые пользователем в JSON, JSONP или в параметрах URL имена файлов. При этом заголовок ответа Content-Type должен быть установлен в text/plain, а заголовок Content-Disposition должно иметь фиксированное имя файла.	✓	✓	✓	641
12.3.5	Убедитесь, что недоверенные метаданные файла не используются напрямую с системными API или библиотеками для защиты от инъекции команд операционной системы.	✓	✓	✓	78
12.3.6	Убедитесь, что приложение не включает и не исполняет функции из недоверенных источников, таких как непроверенные сети доставки контента (CDN), библиотеки JavaScript, пакеты из npm в Node.js, или серверные библиотеки DLL.		✓	✓	829

V12.4 Хранение файлов

№	Описание	L1	L2	L3	CWE
12.4.1	Убедитесь, что файлы, полученные из недоверенных источников, хранятся вне корневого каталога webroot и имеют ограниченные разрешения.	✓	✓	✓	552
12.4.2	Убедитесь, что файлы, полученные из недоверенных источников, проверяются антивирусными сканерами, чтобы предотвратить загрузку и распространение известного вредоносного кода.	✓	✓	✓	509

V12.5 Загрузка файлов клиентом

№	Описание	L1	L2	L3	CWE
12.5.1	Убедитесь, что уровень представления (web) обслуживает только файлы с заранее определенными расширениями, чтобы предотвратить непреднамеренную утечку информации и исходного кода. Например, файлы резервных копий (.bak), временные рабочие файлы (.swp), сжатые файлы (.zip, .tar.gz и т. д.) и другие расширения, обычно используемые редакторами, должны быть заблокированы, если в них нет необходимости.	✓	✓	✓	552
12.5.2	Убедитесь, что прямые запросы к загруженным файлам никогда не будут интерпретироваться как содержимое HTML/JavaScript.	✓	✓	✓	434

V12.6 Защита от SSRF

№	Описание	L1	L2	L3	CWE
12.6.1	Убедитесь, что на web-сервере или сервере приложений настроен список разрешенных ресурсов или систем, к которым сервер может отправлять запросы или с которых он может загружать данные/файлы.	✓	✓	✓	918

Ссылки

Для дополнительной информации см. также:

- Unrestricted File Upload
- Reflective file download by Oren Hafif
- OWASP Third Party JavaScript Management Cheat Sheet

V13 API и web-сервисы

Задачи контроля

Убедитесь, что исследуемое приложение, использующее API на уровне доверенного сервиса (обычно это JSON, XML или GraphQL), имеет:

- Адекватную аутентификацию, управление сессиями и авторизацию для всех web-сервисов.
- Форматно-логический контроль всех входных параметров, которые передаются с более низкого уровня доверия на более высокий.
- Эффективные меры безопасности для всех типов API, включая облачные и бессерверные API.

Пожалуйста, воспринимайте эту главу в контексте с другими главами; мы больше не дублируем требования аутентификации или управления сессиями для API.

V13.1 Базовая безопасность web-сервисов

№	Описание	L1	L2	L3	CWE
13.1.1	Убедитесь, что все компоненты приложения используют одну и ту же кодировку символов и ту же библиотеку синтаксического разбора, чтобы избежать атак, которые используют различные URI или поведение парсера, которое может привести к атакам SSRF и RFI.	✓	✓	✓	116
13.1.2	[УДАЛЕНО, ДУБЛИРУЕТ 4.3.1]				
13.1.3	Убедитесь, что URL-адреса API не содержат конфиденциальной информации (ключ API, сессионный токен и т.д.)	✓	✓	✓	598
13.1.4	Убедитесь, что решения об авторизации принимаются как на уровне URI, за счет процедурной или декларативной безопасности на контроллере или маршрутизаторе, так и на уровне ресурсной системы, с учетом разрешений ролевой модели.		✓	✓	285
13.1.5	Убедитесь, что HTTP-запросы, содержащие непредусмотренные или несуществующие типы контента, отклоняются с соответствующими заголовками (статусы ответа HTTP 406 Unacceptable или HTTP 415 Unsupported Media Type).		✓	✓	434

V13.2 RESTful web-сервисы

JSON-схемы находятся на стадии проекта стандарта (см. Ссылки). При рассмотрении вопроса о возможности проверки JSON-схем — лучшей практики для RESTful web-сервисов, — рассмотрите возможность использования следующих дополнительных проверок данных в сочетании с проверкой JSON по схеме:

- Контроль синтаксического разбора объекта в JSON, например, на наличие или отсутствие элементов.
- Проверка объекта JSON с использованием стандартных методов форматно-логического контроля, таких как тип и формат данных, ограничение длины строки и т.д.
- и формальная проверка JSON-схемы.

Как только стандарт проверки JSON-схем будет формализован, ASVS обновит свои рекомендации в этой области. Внимательно следите за используемыми библиотеками валидации схем, поскольку их необходимо будет регулярно обновлять до тех пор, пока стандарт не будет окончательно утверждён и устранены ошибки в эталонных реализациях.

№	Описание	L1	L2	L3	CWE
13.2.1	Убедитесь, что разрешенные RESTful HTTP-методы допустимы для пользователя или действия, например, обычным пользователям нельзя использовать DELETE или PUT для защищенных API или ресурсов.	✓	✓	✓	650
13.2.2	Прежде чем принимать входные данные в формате JSON, убедитесь в наличии схемы данных, и что по ней проводится форматно-логический контроль.	✓	✓	✓	20
13.2.3	Убедитесь, что RESTful web-сервисы, использующие cookie, защищены от подделки межсайтовых запросов с помощью, по крайней мере, одного или нескольких из следующих способов: двойной проверки случайного значения (в POST при отправке формы и в cookie), анти-CSRF-токена (nonce) или проверки заголовка источника запроса (Origin).	✓	✓	✓	352
13.2.4	[УДАЛЕНО, ДУБЛИРУЕТ 11.1.4]				
13.2.5	Убедитесь, что REST API явно проверяют, предусмотрена ли обработка указанного во входных данных Content-Type, например application/xml или application/json.		✓	✓	436
13.2.6	Убедитесь, что заголовки и тело сообщения заслуживают доверия и не изменяются при передаче. Требования стойкого шифрования при передаче (только TLS) может быть достаточно во многих случаях, поскольку оно обеспечивает как конфиденциальность, так и защиту целостности. Электронные подписи для каждого сообщения могут дать дополнительные гарантии к средствам защиты канала передачи для приложений с высокими требованиями к безопасности, но они сопряжены с дополнительными сложностями и рисками, которые перевешивают преимущества.		✓	✓	345

V13.3 Web сервисы SOAP

№	Описание	L1	L2	L3	CWE
13.3.1	Убедитесь, что для обеспечения правильного формирования XML-документа до его обработки выполняется проверка каждого поля на соответствие ограничений XSD-схемы.	✓	✓	✓	20

№	Описание	L1	L2	L3	CWE
13.3.2	Убедитесь, что тело сообщения подписывается с помощью WS-Security, чтобы обеспечить безопасную передачу между клиентом и сервисом.		✓	✓	345

Примечание. Из-за XXE-атак на DTD не следует использовать проверку DTD, и отключить её в соответствии с требованиями, изложенными в разделе V14.

V13.4 GraphQL

№	Описание	L1	L2	L3	CWE
13.4.1	Убедитесь, что для предотвращения отказа в обслуживании (DoS) GraphQL или выражения уровня данных в результате ресурсоёмких вложенных запросов применяется список разрешенных запросов или комбинация ограничения по глубине и количеству. Для более сложных сценариев следует использовать анализ стоимости запроса.		✓	✓	770
13.4.2	Убедитесь, что логика авторизации в GraphQL или на другом уровне данных реализована на уровне бизнес-логики, а не на уровне GraphQL.		✓	✓	285

Ссылки

Для дополнительной информации см. также:

- OWASP Serverless Top 10
- Проект OWASP Serverless
- OWASP Testing Guide 4.0: Configuration and Deployment Management Testing
- OWASP Cross-Site Request Forgery cheat sheet
- OWASP XML External Entity Prevention Cheat Sheet - General Guidance
- JSON Web Tokens (and Signing)
- REST Security Cheat Sheet
- JSON Schema
- XML DTD Entity Attacks
- Orange Tsai - A new era of SSRF Exploiting URL Parser In Trending Programming Languages

V14 Конфигурация

Задачи контроля

Убедитесь, что исследуемое приложение имеет:

- Безопасную, воспроизводимую, автоматизированную среду сборки.
- Усиленный контроль над библиотеками, зависимостями и конфигурациями сторонних разработчиков таким образом, чтобы устаревшие или небезопасные компоненты не включались в приложение.

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

V14.1 Сборка и развёртывание

Конвейеры сборки являются основой для повторяемой безопасности. Каждый раз, при обнаружении чего-то небезопасного, его можно устранить в исходном коде, сценариях сборки или развертывания и автоматически протестировать. Мы настоятельно рекомендуем использовать конвейеры сборки с автоматическим контролем безопасности и зависимостей, которые предупреждают или прерывают сборку для предотвращения попадания известных уязвимостей в промышленную среду. Ручные, нерегулярно выполняемые проверки, напрямую приводят к ошибкам, которых можно избежать.

По мере перехода к модели DevSecOps важно обеспечить постоянную доступность и целостность развёртывания и конфигурации для достижения эталонного состояния. В прошлом, если система была взломана, требовалось от нескольких дней до месяцев, чтобы доказать, что вторжения не повторялись. Сегодня, с появлением программно-определяемой инфраструктуры, быстрого A/B развёртывания с нулевым временем простоя и автоматизации контейнерных сборок, стало возможным автоматически и непрерывно создавать, укреплять и развёртывать эталонную замену для скомпрометированной системы.

Если до сих пор используются традиционные модели, то необходимо предпринять ручные шаги по укрепления защиты и резервному копированию этой конфигурации, чтобы обеспечить быструю и своевременную замену скомпрометированных систем на безопасные системы с высокой степенью целостности.

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

№	Описание	L1	L2	L3	CWE
14.1.1	Убедитесь, что процессы сборки и развертывания приложений выполняются безопасным и воспроизводимым образом, например посредством автоматизации CI/CD, автоматизированного управления конфигурациями и автоматизированными сценариями развертывания.		✓	✓	
14.1.2	Убедитесь, что включены флаги компилятора для всех имеющихся мер защиты и предупреждений: о переполнении буфера, включая рандомизацию стека (ASLR), предотвращение выполнения данных (DEP) и прерывание сборки при обнаружении небезопасных операций с указателями, памятью, строками форматирования, целыми числами или строками.		✓	✓	120
14.1.3	Убедитесь, что в соответствии с рекомендациями используемого сервера приложений и фреймворка укреплена безопасность конфигурации сервера.		✓	✓	16

№	Описание	L1	L2	L3	CWE
14.1.4	Убедитесь, что приложение, конфигурация и все зависимости могут быть повторно развернуты с помощью сценариев автоматического развёртывания, созданы из документированного и протестированного runbook в разумные сроки или своевременно восстановлены из резервных копий.		✓	✓	
14.1.5	Убедитесь, что для обнаружения несанкционированного доступа уполномоченные администраторы могут проверять целостность всех конфигураций, связанных с безопасностью.			✓	

V14.2 Зависимости

Управление зависимостями имеет решающее значение для безопасной работы любого приложения любого типа. Отсутствие своевременного обновления устаревших или небезопасных зависимостей является основной причиной самых масштабных и дорого обошедшихся атак на сегодняшний день.

Примечание: Соответствие требованию 14.2.1 на уровне L1 относится к определению состава клиентских и прочих библиотек и компонентов, а не к более точному статическому анализу кода во время сборки или анализу зависимостей. Эти детали при необходимости могут быть выяснены устно.

№	Описание	L1	L2	L3	CWE
14.2.1	Убедитесь, что все компоненты обновляются, желательно с помощью инструмента проверки зависимостей во время сборки или компиляции. (C2)	✓	✓	✓	1026
14.2.2	Убедитесь, что удалены все ненужные функции, документация, примеры приложений и конфигураций.	✓	✓	✓	1002
14.2.3	Убедитесь, что если ресурсы приложения, такие как библиотеки JavaScript, CSS или web-шрифты, размещаются извне (в сети доставки контента (CDN) или у внешнего разработчика), то для проверки целостности этого ресурса используется Subresource Integrity (SRI).	✓	✓	✓	829
14.2.4	Убедитесь, что сторонние компоненты берутся из утвержденных, доверенных и постоянно поддерживаемых репозиториев. (C2)		✓	✓	829
14.2.5	Убедитесь, что для всех используемых сторонних библиотек ведется спецификация программного обеспечения (SBOM). (C2)		✓	✓	
14.2.6	Убедитесь, что поверхность атаки сокращается за счет изоляции сторонних библиотек или инкапсуляции их функций таким образом, чтобы они демонстрировали приложению только требуемое поведение. (C2)		✓	✓	265

V14.3 Непреднамеренное раскрытие информации о безопасности

Конфигурации для среды промышленной эксплуатации должны быть укреплены для защиты от распространенных атак, таких как консоли отладки, межсайтовые сценарии (XSS), удаленное включение файлов (RFI) а также для устранения тривиальных «уязвимостей» раскрытия информации, которые, к сожалению, являются приметой многих отчетов о тестировании на проникновение. Сами подобные уязвимости редко оцениваются как значительные, но они «приглашают друзей». Если их нет по умолчанию, это повышает уровень защиты, т.к. многие атаки не смогут увенчаться успехом.

№	Описание	L1	L2	L3	CWE
14.3.1	[УДАЛЕНО, ДУБЛИРУЕТ 7.4.1]				

№	Описание	L1	L2	L3	CWE
14.3.2	Убедитесь, что в промышленной среде отключены режимы отладки web-сервера, сервера или платформы приложений, чтобы отключить функции отладки, консоль разработчика и исключить непреднамеренное раскрытие информации о безопасности.	✓	✓	✓	497
14.3.3	Убедитесь, что в HTTP-заголовках или HTTP-ответах не содержится информации о версии системных компонентов.	✓	✓	✓	200

V14.4 HTTP-заголовки безопасности

№	Описание	L1	L2	L3	CWE
14.4.1	Убедитесь, что каждый HTTP-ответ содержит заголовок Content-Type. Если тип контента — text/*, /+xml или application/xml, то должна быть указана безопасная кодовая страница (например, charset=UTF-8 или ISO-8859-5). Контент должен соответствовать предоставленному заголовку Content-Type.	✓	✓	✓	173
14.4.2	Убедитесь, что все ответы API содержат заголовок Content-Disposition: attachment; filename="api.json" (или другое наименование файла, подходящее для этого типа контента).	✓	✓	✓	116
14.4.3	Убедитесь, что присутствует заголовок ответа Content Security Policy (CSP), который помогает ослабить воздействие XSS-атак через HTML, DOM, JSON или через инъекции JavaScript.	✓	✓	✓	1021
14.4.4	Убедитесь, что ответы содержат заголовок X-Content-Type-Options: nosniff.	✓	✓	✓	116
14.4.5	Убедитесь, что заголовок Strict-Transport-Security включен во все ответы и для всех поддоменов, например, Strict-Transport-Security: max-age=15778463; includeSubDomains.	✓	✓	✓	523
14.4.6	Убедитесь, что присутствует заголовок Referrer-Policy с уместным для приложения значением, чтобы избежать раскрытия недоверенным сторонам конфиденциальной информации в URL-адресе через заголовок Referer.	✓	✓	✓	116
14.4.7	Убедитесь, что содержимое web-приложения по умолчанию не может быть встроено на сторонний сайт, и что встраивание оригинальных ресурсов приложения возможно только с разрешения автора, используя соответствующие заголовки ответов Content-Security-Policy: frame-ancestors и X-Frame-Options.	✓	✓	✓	1021

V14.5 Проверка заголовка HTTP-запроса

№	Описание	L1	L2	L3	CWE
14.5.1	Убедитесь, что сервер приложений принимает только те HTTP-методы, которые использует приложение/API, и регистрирует/предупреждает о любых запросах, недопустимых для контекста приложения.	✓	✓	✓	749
14.5.2	Убедитесь, что полученный заголовок Origin не используется для аутентификации или при принятии решений о контроле доступа, поскольку он может быть легко изменен злоумышленником.	✓	✓	✓	346

№	Описание	L1	L2	L3	CWE
14.5.3	Убедитесь, что заголовок Access-Control-Allow-Origin при совместном использовании ресурсов (CORS) применяет для сопоставления строгий список разрешенных доверенных доменов и поддоменов, а не "null".	✓	✓	✓	346
14.5.4	Убедитесь, что приложением аутентифицируются HTTP-заголовки, добавляемые доверенным прокси-сервером или устройствами единого входа (SSO), например, через токен на предъявителя (bearer).		✓	✓	306

Ссылки

Для дополнительной информации см. также:

- OWASP Web Security Testing Guide 4.1: Testing for HTTP Verb Tampering
- Добавление Content-Disposition к ответам API помогает предотвратить многие атаки, основанные на несогласованности в MIME type между клиентом и сервером, а опция filename помогает предотвратить атаки Reflected File Download (RFD)
- Content Security Policy Cheat Sheet
- Exploiting CORS misconfiguration for BitCoins and Bounties
- OWASP Web Security Testing Guide: Configuration and Deployment Management Testing
- Sandboxing third party components

Приложение A: Термины и сокращения

- **Address Space Layout Randomization** (ASLR) –*рандомизация распределения адресного пространства*– метод, позволяющий усложнить использование ошибок, связанных с повреждением памяти.
- **Allow list** – список разрешённых данных или операций, например список символов, разрешённых форматно-логическим контролем.
- **Application Security** –*безопасность приложений*– анализ компонентов, составляющих прикладной уровень эталонной модели взаимодействия открытых систем (модели OSI), и в меньшей степени – инфраструктуры, например, операционных систем или сетей.
- **Application Security Verification** –*верификация требований к безопасности приложений*– технический анализ приложения по стандарту OWASP ASVS.
- **Application Security Verification Report** – отчёт по тестированию конкретного приложения, в котором аудитор документирует общие результаты и даёт сопроводительные пояснения.
- **Authentication** –*аутентификация*– проверка учетных данных, предъявленных пользователем приложения.
- **Automated Verification** –*автоматизированная проверка*– поиск сигнатур уязвимостей с помощью автоматизированных инструментов динамического и/или статического анализа безопасности кода.
- **Black box testing** –*метод «чёрного ящика»*– метод тестирования программного обеспечения, который проверяет функциональность приложения, не заглядывая в его внутреннюю структуру и операции.
- **Component** –*компонент*– автономный блок кода со связанными дисковыми и сетевыми интерфейсами, который взаимодействует с другими компонентами.
- **Cross-Site Scripting** (XSS) –*межсайтовый скриптинг*– уязвимость в системе безопасности, обычно обнаруживаемая в web-приложениях, позволяющая внедрять скрипты в контент на стороне клиента.
- **Cryptographic module** –*криптографический модуль*– аппаратное, программное и/или микропрограммное обеспечение, реализующее криптографические алгоритмы и/или генерирующее криптографические ключи.
- **Common Weakness Enumeration** (CWE) –*список общих недостатков*– разработанный сообществом список распространенных недостатков безопасности программного обеспечения. Он служит общим языком, мерилом для средств обеспечения безопасности программного обеспечения, а также основой для выявления недостатков, их устранения и предотвращения.
- **Design Verification** –*верификация дизайна*– технический анализ архитектуры безопасности приложения.
- **Dynamic Application Security Testing** (DAST) –*динамический анализ кода*– технологии, предназначенные для обнаружения условий, указывающих на уязвимость системы безопасности в приложении, производящийся во время его выполнения на реальном или виртуальном процессоре.
- **Dynamic Verification** –*динамическая верификация*– использование автоматизированных инструментов, использующих сигнатуры уязвимостей для поиска проблем во время выполнения приложения.
- **Fast IDentity Online** (FIDO) - набор стандартов аутентификации, которые позволяют использовать различные методы аутентификации, включая биометрические данные, модули доверенной платформы (TPM), USB-токены и т.д.

- **Globally Unique Identifier** (GUID) – уникальный индекс, используемый в качестве идентификатора в программном обеспечении.
- **Hyper Text Transfer Protocol** (HTTPS) – протокол приложений для распределенных, гипермедиа (гипертекст+мультимедиа)-систем и совместной работы. Основа для передачи данных в World Wide Web.
- **Hardcoded keys** –*жёстко запрограммированные ключи*– криптографические ключи, которые хранятся в файловой системе, будь то код, комментарии или файлы.
- **Hardware Security Module** (HSM) —*аппаратный модуль безопасности*— аппаратный компонент, способный хранить криптографические ключи и другие секреты в защищённом виде.
- **Hibernate Query Language** (HQL) – язык запросов, внешне похожий на SQL, используемый библиотекой Hibernate ORM.
- **Input Validation** —*форматно-логический контроль*— нормализация и проверка корректности недоверенных входных данных.
- **Malicious Code** —*вредоносный код*— код, попавший в приложение во время его разработки без ведома его автора, который обходит имеющуюся политику безопасности приложения. Не путать с вредоносным ПО, таким как вирусы или черви.
- **Malware** —*вредоносное ПО*— исполняемый код, который попадает в приложение во время выполнения без ведома пользователя или администратора приложения.
- **Open Web Application Security Project** (OWASP) – бесплатно распространяемый проект и открытое международное сообщество, занимающееся повышением безопасности прикладного программного обеспечения. Наша миссия — сделать безопасность приложений «видимой», чтобы люди и организации могли принимать обоснованные решения о рисках безопасности приложений. См.: https://www.owasp.org/
- **One-time Password** (OTP) —*одноразовый пароль*— уникальный пароль, сгенерированный для одноразового использования.
- **Object-relational Mapping** (ORM) - система, позволяющая приложению делать запросы к реляционной/табличной базе данных, используя объектную модель, совместимую с приложением.
- **Password-Based Key Derivation Function 2** (PBKDF2) - специальный односторонний алгоритм для формирования стойкого криптографического ключа на основе введённого текста (например, пароля) и дополнительного случайного значения — соли, которая затрудняет взлом пароля в режиме offline, если вместо исходного пароля хранится результирующее значение.
- **Personally Identifiable Information** (PII) —*информация, позволяющая установить личность*— информация, которая может быть использована сама по себе или в совокупности с другой информацией для идентификации, установления контакта или определения местонахождения конкретного человека.
- **Position-independent executable** (PIE) —*позиционно-независимый исполняемый файл*— текст машинного кода, который, будучи помещен где-то в основной памяти, выполняется должным образом независимо от его абсолютного адреса.
- **Public Key Infrastructure** (PKI) —*инфраструктура открытых ключей*— механизм, который связывает открытые ключи с соответствующими идентификаторами сущностей. Привязка устанавливается посредством процесса регистрации и выдачи сертификатов в центре сертификации (CA) и с его помощью.
- **Public Switched Telephone Network** (PSTN) —*коммутируемая телефонная сеть общего пользования (ТСОП, ТфОП)*— традиционная телефонная сеть, соединяющая как стационарные, так и мобильные телефоны.
- **Relying Party** (RP) —*проверяющая сторона*— как правило, приложение, которое удостоверяет пользователя данного поставщика учетных данных. Приложение проверяет токен или набор

подписанных утверждений, предоставляемых этим поставщиком, чтобы убедиться, что пользователь является тем, за кого себя выдает.

- **Static application security testing** (SAST) –*статический анализ кода*– ряд технологий, предназначенных для анализа исходного кода приложения, байт-кода и бинарных файлов на наличие условий разработки и проектирования, которые указывают на уязвимости в системе безопасности. Решения SAST анализируют приложение «изнутри» в незапущенном состоянии.
- **Software development lifecycle** (SDLC) –*жизненный цикл разработки программного обеспечения*– пошаговый процесс разработки ПО, начиная с первоначальных требований, и заканчивая развертыванием и сопровождением.
- **Security Architecture** –*архитектура безопасности*– абстракция дизайна приложения, которая определяет и описывает, где и как применяются меры безопасности, а также местоположение и категории информации.
- **Security Configuration** –*конфигурация безопасности*– конфигурация среды выполнения приложения, которая влияет на то, как применяются меры безопасности.
- **Security Control** –*мера обеспечения безопасности*– функция или компонент, который контролирует (например, меры контроля доступа) или приводит к действию по обеспечению безопасности (например, регистрирует запись в журнале аудита).
- **Server-side Request Forgery** (SSRF) –*подделка запросов на стороне сервера*– атака, которая злоупотребляет функциональными возможностями сервера для чтения или обновления внутренних ресурсов путем предоставления или изменения URL-адреса, по которому код, запущенный на сервере, будет читать или отправлять данные.
- **Single Sign-on Authentication** (SSO) –*аутентификация единого входа*– пользователь входит в одно приложение, одновременно получая возможность войти в несколько других, без необходимости повторной аутентификации. Например, когда вы входите в Google, вы автоматически входите в другие сервисы Google, такие как YouTube, Google Docs, Gmail и т.д.
- **SQL Injection** (SQLi) –*SQL-инъекция*– метод внедрения кода, используемый для атаки приложений, управляемых данными, в которых вредоносные SQL-выражения вставляются в точку входа.
- **SVG** - Scalable Vector Graphics –*масштабируемая векторная графика*
- **Time-based OTP** (TOTP) - метод создания OTP, при котором текущее время является входным аргументом для алгоритма формирования пароля.
- **Threat Modeling** –*моделирование угроз*– метод, состоящий в совершенствовании архитектур безопасности для выявления нарушителей, границ доверия, мер безопасности, а также ключевых технических и бизнес-активов.
- **Transport Layer Security** (TLS) –*безопасность транспортного уровня*– криптографические протоколы, обеспечивающие безопасность канала связи по сетевому соединению.
- **Trusted Platform Module** (TPM) –*модуль доверенной платформы*– тип HSM, который обычно является компонентом другого оборудования, например, материнской платы, и выступает в роли «корня доверия» для этой системы.
- **Two-factor authentication** (2FA) –*двухфакторная аутентификация (2ФА)*– добавляет еще один уровень аутентификации для входа в учетную запись.
- **Universal 2nd Factor** (U2F) –*универсальный второй фактор*– один из стандартов, разработанных FIDO специально для использования в качестве второго фактора аутентификации NFC- или USB-токена.
- **URI/URL/фрагменты URL** –*унифицированный идентификатор ресурса*– это строка символов, используемая для идентификации имени или web-ресурса. –*Унифицированный указатель ресурсов*– часто используется в качестве ссылки на ресурс.

- **Verifier** –*верификатор/аудитор*– лицо или группа людей, которые анализируют безопасность приложение на соответствие требованиям OWASP ASVS.
- **What You See Is What You Get** (WYSIWYG) – тип редактора гипермедиа, который показывает, как на самом деле будет выглядеть содержимое при отображении, а не код разметки, управляющий отображением.
- **Сертификат X.509** – электронный сертификат, соответствующий общепринятому международному стандарту инфраструктуры открытых ключей X.509 (PKI) для проверки принадлежности открытого ключа идентификатору пользователя, компьютера или сервиса, указанному в сертификате.
- **XML eXternal Entity** (XXE) – тип элемента XML, который может получать доступ к локальному или удаленному содержимому через объявленный идентификатор. Может привести к атакам инъекции.

Приложение B: Полезные ссылки

Следующие проекты OWASP, скорее всего, будут полезны для читателей/последователей этого стандарта:

Основные проекты OWASP

1. Проект OWASP Top 10: 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/

Проект OWASP Cheat Sheet Series

В этом проекте есть памятки, которые будут актуальны для многих требований ASVS.

Сопоставление памяток с требованиями ASVS можно найти здесь:
https://cheatsheetseries.owasp.org/cheatsheets/IndexASVS.html

Проекты Mobile Security

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/

Проект OWASP Internet of Things

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

Проект OWASP Serverless

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

Другие

Следующие web-сайты, скорее всего, также будут полезны для читателей/последователей этого стандарта:

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 верификации Internet of Things

Эта глава изначально была в основной ветке, но с учётом работы, проделанной командой OWASP IoT, нет смысла поддерживать две разные ветки по этому вопросу. Для версии 4.0 мы переносим это в Приложение и призываем всех, кому это требуется, переходить в основной проект OWASP IoT

Задачи контроля

Встраиваемые / IoT-устройства должны:

- Обеспечивать тот же уровень безопасности на устройстве, что и на сервере, применяя меры безопасности в доверенной среде.
- Конфиденциальные данные, хранящиеся на устройстве, должны храниться безопасным образом с использованием аппаратного хранилища, например, Secure Element.
- Все конфиденциальные данные, передаваемые с устройства, должны использовать безопасность транспортного уровня (TLS).

Верификация требований к безопасности

№	Описание	L1	L2	L3	Версия
C.1	Убедитесь, что последовательные интерфейсы отладки прикладного уровня, такие как USB, UART и другие, 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 во встроенной операционной системе/IoT включены средства защиты памяти, такие как ASLR и DEP, если применимо.	✓	✓	✓	4.0
C.4	Убедитесь, что отключены встроенные интерфейсы отладки, такие как JTAG или SWD, или что имеющийся механизм защиты включен и настроен соответствующим образом.	✓	✓	✓	4.0
C.5	Убедитесь, что если на SoC или процессоре устройства реализовано доверенное выполнение, то оно включено.	✓	✓	✓	4.0
C.6	Убедитесь, что конфиденциальные данные, закрытые ключи и сертификаты безопасно хранятся в Secure Element, TPM, TEE (Trusted Execution Environment) или защищены с помощью стойкой криптографии.	✓	✓	✓	4.0
C.7	Убедитесь, что приложения встроенного ПО защищают передаваемые данные с помощью безопасности транспортного уровня (TLS).	✓	✓	✓	4.0
C.8	Убедитесь, что приложения встроенного ПО проверяют электронную подпись подключений к серверу.	✓	✓	✓	4.0
C.9	Убедитесь, что беспроводные подключения проходят взаимную аутентификацию.	✓	✓	✓	4.0
C.10	Убедитесь, что беспроводные подключения осуществляются по зашифрованному каналу.	✓	✓	✓	4.0
C.11	Убедитесь, что любое использование запрещенных функций на языке C заменено их безопасными эквивалентами.	✓	✓	✓	4.0

№	Описание	L1	L2	L3	Версия
C.12	Убедитесь, что каждая "прошивка" содержит спецификацию программного обеспечения с каталогом сторонних компонентов, версиями и опубликованными уязвимостями.	✓	✓	✓	4.0
C.13	Убедитесь, что весь код, включая сторонние бинарные файлы, библиотеки и фреймворки, проверяется на наличие в коде учётных данных (бэкдоров).	✓	✓	✓	4.0
C.14	Убедитесь, что компоненты приложения и встроенного ПО не подвержены инъекции команд операционной системы, вызывая команды оболочки ОС или скрипты, и что меры безопасности их предотвращают.	✓	✓	✓	4.0
C.15	Убедитесь, что приложения встроенного ПО закрепляют (pinning) электронную подпись на доверенных серверах.		✓	✓	4.0
C.16	Убедитесь в наличии средств защиты от несанкционированного доступа и/или его обнаружения.		✓	✓	4.0
C.17	Убедитесь, что включены все имеющиеся технологии защиты интеллектуальной собственности, предоставляемые производителем чипа.		✓	✓	4.0
C.18	Убедитесь, что меры безопасности препятствуют обратному инжинирингу встроенного ПО (например, удалению символов детализации отладки).		✓	✓	4.0
C.19	Убедитесь, что перед загрузкой устройство проверяет подпись загрузочного образа.		✓	✓	4.0
C.20	Убедитесь, что процесс обновления прошивки не подвержен атакам типа «момент проверки до момента использования».		✓	✓	4.0
C.21	Перед установкой убедитесь, что устройство использует подпись кода и проверяет файлы обновления прошивки.		✓	✓	4.0
C.22	Убедитесь, что версия прошивки устройство не может быть понижена до старых версий (анти-откат).		✓	✓	4.0
C.23	Проверьте использование криптографического генератора псевдослучайных чисел на встроенном устройстве (например, с использованием генераторов случайных чисел, предоставляемых чипом).		✓	✓	4.0
C.24	Убедитесь, что прошивка может выполнять автоматическое обновление по заданному расписанию.		✓	✓	4.0
C.25	Убедитесь, что устройство стирает прошивку и конфиденциальные данные при обнаружении несанкционированного доступа 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 те микроконтроллеры, которые обеспечивают существенную защиту от декапинга (анг.: de-capping) и атак по побочным каналам.			✓	4.0

№	Описание	L1	L2	L3	Версия
C.28	Убедитесь, что конфиденциальные дорожки не соприкасаются с внешними слоями печатной платы.			✓	4.0
C.29	Убедитесь, что канал связи между микросхемами зашифрован (например, между основной 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 встроенного ПО используют контейнеры ядра для изоляции между приложениями.			✓	4.0
C.33	Убедитесь, что для сборок микропрограмм настроены безопасные флаги компилятора, такие как -fPIE, -fstack-protector-all, -Wl,-z,noexecstack, -Wl,-z,noexecheap.			✓	4.0
C.34	Убедитесь, что в микроконтроллере настроена защита кода (если применимо).			✓	4.0

Ссылки

Для дополнительной информации см. также:

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