1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**КУРСОВАЯ РАБОТА**

**Анализ методологий тестирования безопасности веб-приложений**

по дисциплине «Модели безопасности компьютерных систем»

1. Выполнила
2. студентка гр. 4851003/90801 Кулеева А.Г.



1. Руководитель Овасапян Т.Д.
2. ст. преподаватель

«\_\_» \_\_\_\_\_\_\_\_\_\_ 2021 г.

1. Санкт-Петербург
2. 2021

Содержание

|  |  |  |
| --- | --- | --- |
| Введение........................................................................................................... | | 3 |
| 1 | Основные методологии тестирования…………………………….. | 5 |
| 1.1 | OSSTMM............................................................................................ | 5 |
| 1.2 | ISSAF.................................................................................................. | 6 |
| 1.3 | OWASP............................................................................................... | 11 |
| 1.4 | PTF...................................................................................................... | 13 |
| 1.5 | PTES................................................................................................... | 14 |
| 1.6 | Positive Technologies......................................................................... | 16 |
| 2 | Анализ рассмотренных методологий............................................. | 20 |
| 3 | OWASP............................................................................................... | 24 |
| 3.1 | Описание проекта OWASP.............................................................. | 24 |
| 3.1.1 | Что такое тестирование? ................................................................. | 24 |
| 3.1.2 | Зачем нужно тестировать? .............................................................. | 24 |
| 3.1.3 | Когда проводить тестирование? ...................................................... | 24 |
| 3.1.4 | Что тестировать? ............................................................................... | 25 |
| 3.2 | Принципы тестирования................................................................... | 25 |
| 3.3 | Техники, используемые для тестирования..................................... | 26 |
| 3.4 | Получение требований к тестированию......................................... | 30 |
| 3.4.1 | Цели тестирования............................................................................ | 30 |
| 3.4.2 | Документация требований тестирования........................................ | 30 |
| 3.4.3 | Проверка требований безопасности................................................. | 31 |
| 3.4.4 | Таксономия угроз и контрмер.......................................................... | 31 |
| 3.4.5 | Тестирование безопасности и анализ рисков................................. | 31 |
| 3.4.6 | Выведение функциональных и нефункциональных требований к тестированию..................................................................................... | 32 |
| 3.4.6.1 | Функциональные требования безопасности................................... | 32 |
| 3.4.6.2 | Требования безопасности с учетом рисков.................................... | 32 |
| 3.4.7 | Выведение требований к тестированию безопасности с помощью примеров корректного и некорректного использования................................................................................... | 33 |
| 3.5 | Тесты безопасности, интегрированные в рабочие процессы разработки и тестирования.............................................................. | 35 |
| 3.5.1 | Тестирование безопасности в рабочем процессе разработки…… | 35 |
| 3.5.2 | Тестирование безопасности в рабочем процессе тестирования… | 35 |
| 3.5.3 | Тесты безопасности для разработчиков......................................... | 36 |
| 3.5.3.1 | Тестирование безопасности на этапе написания кода: Модульные тесты.............................................................................. | 36 |
| 3.5.4 | Тесты безопасности функциональных тестировщиков................. | 37 |
| 3.5.4.1 | Тестирование безопасности на этапе интеграции и валидации: Интегрированные системные тесты и эксплуатационные тесты | 37 |
| 3.6 | Анализ и отчетность по данным тестирования безопасности…… | 38 |
| 3.6.1 | Цели для метрик и измерений при тестировании безопасности… | 38 |
| 3.6.2 | Требования к отчётности.................................................................. | 41 |
| 3.6.3 | Бизнес-кейсы...................................................................................... | 42 |
| Заключение...................................................................................................... | | 44 |
| Список использованных источников............................................................. | | 45 |

Введение

**Актуальность**

На сегодняшний день проблема безопасности нахождения в сети Интернет является достаточно актуальной, поскольку сама сеть популярна по всему миру. Рассмотрим несколько наиболее популярных видов уязвимостей: XSS (Cross-Site Scripting ― Межсайтовый скриптинг), SQL injection, [подделка межсайтовых запросов (CSRF)](https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Website_security#%D0%BF%D0%BE%D0%B4%D0%B4%D0%B5%D0%BB%D0%BA%D0%B0_%D0%BC%D0%B5%D0%B6%D1%81%D0%B0%D0%B9%D1%82%D0%BE%D0%B2%D1%8B%D1%85_%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%BE%D0%B2_csrf), Clickjacking, Denial of Service (DoS), Directory Traversal (Файл и раскрытие), File Inclusion. Они до сих пор встречаются в некоторых веб приложениях, несмотря на то, что есть элементарные способы предотвратить их. Приведённый список является далеко не полным и есть ещё множество уязвимостей веб-приложений, например, на аппаратном уровне или на уровне социальной инженерии. Очевидно, что разработчик физически не может уследить за всеми параметрами безопасности, поэтому существуют особые методологии тестирования веб-приложений.

**Цель работы**: провести сравнительный анализ различных методологий тестирования безопасности веб-приложений

**Задачи**:

1. Познакомиться с основными принципами тестирования,
2. Рассмотреть наиболее популярные методологии тестирования на проникновение (penetration test),
3. Рассмотреть несколько из представленных методологий более подробно, выявить их ключевые особенности, преимущества и недостатки,
4. Провести сравнительный анализ на предмет полноты и актуальности выбранных методологий.

Также стоит отметить, что помимо тестирования безопасности есть ещё некоторые аспекты тестирования, которые обязательно должны быть рассмотрены, ведь в целом любая ошибка может быть угрозой безопасности. Итак, существуют две категории:

1. Функциональное тестирование веб-приложений
   * Проверка внешних и внутренних ссылок;
   * Тестирование интерфейса;
   * Тестирование бизнес-логики;
   * Тестирование навигации;
   * Кроссбраузерное тестирование, включая тестирование на мобильных устройствах
2. Нефункциональное тестирование веб-приложений
   * Тестирование защищенности;
     + тестирование сервера;
     + тестирование клиента;
     + тестирование баз данных;
   * Нагрузочное тестирование
   * Тестирование удобства использования

Но прежде чем перейти непосредственно к методологиям рассмотрим некоторые принципы, относительно которых может осуществляться тестирование.

* DAST ― динамический (т.е. требующий выполнения) анализ приложения без доступа к исходному коду и серверной части. Взаимодействие осуществляется через веб-интерфейс. Такое тестирование ещё называется Чёрным ящиком (BlackBox).
* SAST ― статический (т.е. не требующий выполнения) анализ приложения с доступом к исходному коду веб-приложения и к веб-серверу. По сути это визуальный анализ исходного кода по формальным признакам наличия уязвимостей и аудит безопасности сервера.
* IAST ― динамический анализ безопасности веб-приложения, с полным доступом к исходному коду и веб-серверу. Такое тестирование носит название Серого ящика (WhiteBox).
* Анализ исходного кода ― статический или динамический анализ с доступом к исходному коду без доступа к серверному окружению. [3]

# основные методологии тестирования веб безопасности

## OSSTMM

OSSTMM ― The Open Source Security Testing Methodology Manual. Методика разработана институтом ISECOM (Institute for Security and open Methodologies), который является открытым сообществом ученых и практиков в области ИБ, основанным совместно в Нью-Йорке и Барселоне. Финансирование ISECOM обеспечивается за счет партнерских отношений, подписок, сертификации, лицензирования, семинаров и исследовательских пожертвований. Первая публикация института была ещё в январе 2001 года. Многие исследователи из различных научных областей поделились своим опытом и знаниями, потому что они видели необходимость в таком открытом методе, который был бы привязан к фактам, а не к коммерческой выгоде или политическим программам. Это был шаг к значительному улучшению тестирования, анализа и реализации безопасности. [6]

Данная методика ориентирована преимущественно на тестирование компьютерных сетей. Важно отметить, что методология не предоставляет классификации уязвимостей, однако вводится понятие уязвимости ― это некоторое ограничение безопасности, дефект или ошибка, запрещающая доступ авторизованным пользователям или процессам к информационным ресурсам или позволяющая несанкционированный доступ неавторизованных пользователей или процессов к ресурсам.

Методика OSSTMM определяет, так называемую «карту безопасности» ― то есть визуальное отображение основных категорий ИБ, которые оцениваются в процессе тестирования:

* информационная безопасность;
* безопасность социальных процессов;
* безопасность информационных процессов;
* безопасность Интернет-технологий;
* безопасность каналов связи;
* безопасность беспроводных технологий;
* безопасность физической инфраструктуры. [1]

Методика OSSTMM содержит следующие разделы:

1. Что необходимо знать;
2. Что необходимо делать;
3. Анализ безопасности;
4. Метрики безопасности объекта;
5. Анализ социальных процессов в персонале объекта тестирования;
6. Процесс тестирования;
7. Тестирование устойчивости персонала к социальной инженерии;
8. Тестирование безопасности физической инфраструктуры;
9. Тестирование безопасности беспроводных технологий;
10. Тестирование безопасности телекоммуникационных технологий;
11. Тестирование безопасности данных;
12. Рекомендации по следованию национальным стандартам и соглашениям;
13. Подготовка отчета о тестировании. [7]

Данное исследование охватывает широкий спектр вопросов, поскольку оно объединяет несколько дисциплин с целью получить более глубокое понимание того, что такое безопасность, а также как на неё можно повлиять физически, культурно, социально и психологически. В документе охватываются такие аспекты безопасности как конфиденциальность, доверие, уязвимости, и атаки. [8]

## ISSAF

Методика ISSAF ― Information System Security Assessment Framework разработана в 2006 году консорциумом OISSG (Open Information Systems Security Group) в качестве стандарта внутреннего аудита организаций этого консорциума. При данном аудите выполняется оценка следующих аспектов ИБ:

* оценка политик и процедур ИБ организации, а также степень их соответствия ИТ-стандартам и требованиям нормативных документов в области ИБ;
* выявление и оценка «зависимости» бизнес-процессов организаций от ИТ-инфраструктуры;
* проведение оценки уязвимостей и тестов на проникновение для выделения уязвимостей в системе, которые могут привести к потенциальным рискам информационных ресурсов;
* указание моделей оценки по доменам безопасности;
* нахождение и устранение неправильных конфигураций аппаратно-программных средств;
* идентификация и снижение рисков, связанных с ИТ;
* идентификация и снижение рисков, связанных с персоналом или бизнес-процессами;
* усиление безопасности существующих процессов и технологий;
* внедрение лучшего опыта обеспечения ИБ в практику и процедуры бизнес-процессов.

Методика ISSAF включает в себя большое количество вопросов, связанных с тестированием ИБ, а материал методики организован в виде двух частей: рекомендации для менеджмента и рекомендации по тестированию. Стоить отметить, что в ISSAF для уязвимостей определяется два типа рисков: технический риск и бизнес-риск. В свою очередь каждый из них делится на 3 уровня: низкий, средний, высокий.

Материал методики декомпозирован на 14 подразделов.

1. Управление проектом по тестированию.
2. Основные принципы и лучшие практики в проведении тестирования.
3. Схема процесса тестирования.
4. Обзор политики безопасности и способов повышения ИБ.
5. Методология оценки рисков.
6. Тестирование технических аспектов ИБ:
   1. тестирование криптоустойчивости паролей;
   2. тестирование безопасности операционных систем Unix/Linux, Windows, Novell Netware;
   3. тестирование безопасности баз данных;
   4. тестирование безопасности беспроводных сетей и коммуникаций;
   5. тестирование безопасности коммутаторов и маршрутизаторов
   6. тестирование безопасности брандмауэров
   7. тестирование безопасности систем обнаружения вторжений
   8. тестирование безопасности частных виртуальных сетей VPN
   9. тестирование безопасности антивирусных систем
   10. тестирование безопасности распределенных систем хранения данных
   11. тестирование безопасности Интернет-коммуникаций
   12. тестирование безопасности пользователей
   13. тестирование безопасности исходного кода
   14. тестирование безопасности бинарного кода
7. Тестирование социально-психологических аспектов ИБ.
8. Тестирование физической инфраструктуры.
9. Анализ инцидентов.
10. Отчетность по результатам аудита и тестирования.
11. Обеспечение непрерывности бизнес-процессов и восстановление после инцидентов.
12. Повышение полноты мониторинга и обучение в области ИБ.
13. Аутсорсинг проведения тестирования и обеспечения ИБ.
14. База знаний:
    1. правовые аспекты тестирования и аудита ИБ;
    2. рекомендации по составлению договора о неразглашении информации;
    3. рекомендации по составлению договора на тестирование и аудит;
    4. шаблоны типовых документов;
    5. контрольный список проверки ОС Windows, Linux, Solaris
    6. порты по умолчанию у брандмауэров
    7. порты по умолчанию у систем обнаружения вторжений;
    8. ссылки на другие документы и ресурсы;
    9. рекомендации по программному обеспечению, которое можно использовать для проведения тестирования.

Также в методологии рассмотрен подробный алгоритм тестирования на проникновение в систему.

1. Планирование и подготовка. Получение начальной исходной информации об объекте тестирования, планирование и подготовка к тестам. Перед тестированием сторонам необходимо будет подписать формальное соглашение, которое обеспечит основу для проведения тестирования и взаимную правовую защиту. В нем также будет указан порядок взаимодействия, точные даты, длительность тестирования, способы проведения тестирования и т.д.
2. Оценка. На этом этапе производится выполнение тестирования. Предусмотрены следующие подэтапы проведения тестирования:
   1. сбор информации. Для сбора информации в методике ISSAF рекомендуется использовать Интернет. При этом получаемая информация делится на две группы: техническая (DNS/WHOIS) и нетехническая (поисковые системы, группы новостей и т.д.). Данный этап позволяет выделить «точки уязвимости», которые будут использоваться в дальнейшем;
   2. сетевое картографирование. Применение специальных технических средств для определения структуры сети и ее ресурсов;
   3. идентификация уязвимостей. Перед этой стадией, аудитор определяет уязвимые объекты и способы их тестирования. Отметим, что в методике ISSAF для уязвимостей определяется два типа рисков: технический риск и бизнес-риск, каждый из которых в свою очередь делится на 3 уровня: низкий, средний, высокий. В процессе тестирования предполагается выполнение следующих мероприятий:
      1. идентификация уязвимостей почтовых сервисов;
      2. выполнение углубленного сканирования сетевых информационных ресурсов и сетевых сервисов на предмет поиска известных уязвимостей. Информация об известных уязвимостях берется из открытых баз данных уязвимостей;
      3. верификация полученной информации об уязвимостях путем сравнения и проверки информации об уязвимостях, полученных из различных источников или различными способами;
      4. документирование обнаруженных уязвимостей;
      5. классификация найденных уязвимостей;
      6. определение сценариев атак и сценариев использования эксплойтов.
3. Непосредственно тестирование на проникновение.
4. Получение доступа или расширение привилегий. Конечной целью аудитора на данном этапе является получение доступа к аккаунту администратора сети. В случае, когда в сети разрешены только аккаунты с минимальным количеством привилегий, выполняется составление карты локальных уязвимостей, производится разработка или получение корректного эксплойта, затем он тестируется в изолированной среде и применяется к компрометированной системе. Получение минимальных привилегий доступа возможно через доступ к непривилегированным аккаунтам с помощью следующих способов:
   1. подбор комбинаций логин/пароль путем атаки со словарем;
   2. поиск пустых или стандартных паролей в системных аккаунтах;
   3. выявление эксплойтов в стандартных настройках сетевого оборудования;
   4. поиск публичных сервисов, допускающих определенные операции в системе (запись/создание/чтение файлов).
5. Дополнительные тесты, например, получение зашифрованных паролей для их последующего взлома в режиме off-line, перехват трафика и его анализ и т.д.
6. Компрометация удаленных пользователей, информационных ресурсов, объектов сети.
7. Поддержка несанкционированного доступа к сети.
8. Сокрытие следов работы.

Данная методика является наиболее подробной из рассматриваемых как в теоретическом, так и в практическом смысле. Эту методику можно использовать как на этапе предварительной оценки защищенности объектов сети в интересах проверки возможности их использования в составе какой-либо информационной системы, так и на этапе разработки объектов для проверки отдельных возможностей и функций ИБ. [1]

## OWASP

Компания OWASP была основана в 2001 году. В 2003 фонд компании был зарегистрирован в США. OWASP служит не личным целям её руководителей, а распространению знаний. В компании официально работает 8 человек, вследствие чего у проекта крайне низкие расходы, покрываемые конференциями, корпоративными спонсорами и рекламой. OWASP ежегодно награждает грантами корпоративных и индивидуальных членов за разработку многообещающих приложений, обеспечивающих безопасность. С 2011 года OWASP зарегистрирована как некоммерческая организация в Бельгии.

Методология OWASP ― Open Web Application Security Project была создана в 2004 г. и развивается по настоящее время международной группой независимых экспертов-энтузиастов. Методика подробно описывает тестирование веб-приложений и фактически является единственной подобной методологией, узко ориентированной именно на веб-приложения. [1] В 2020 г. вышла редакция руководства OWASP Web Security Testing Guide v4.1., она содержит следующие разделы:

1. Введение
   1. Проект тестирования OWASP
   2. Принципы тестирования
   3. Объяснение методов тестирования
   4. Инспекция мануала
   5. Моделирование угроз
   6. Обзор исходного кода
   7. Тестирование на проникновение
   8. Необходимость сбалансированного подхода
   9. Получение требований к тестированию безопасности
   10. Интеграция тестов безопасности в рабочие процессы разработки и тестирования
   11. Анализ данных и создание отчетов по тестированию безопасности
2. Платформа тестирования OWASP.
   1. Платформа тестирования веб-безопасности
   2. Этап 1: До начала разработки
   3. Этап 2: Во время определения и проектирования
   4. Этап 3: В процессе разработки
   5. Этап 4: Во время развертывания
   6. Этап 5: Во время технического обслуживания и эксплуатации
   7. Типичный рабочий процесс тестирования SDLC
   8. Методологии тестирования на проникновение
3. Тестирование безопасности веб-приложений. Этот раздел является наиболее объемным, поскольку именно в нём подробнейшим образом описаны все аспекты тестирования до самых мельчайших деталей. Однако в данной работе обзор будет более поверхностным.
   1. Введение и цели
   2. Сбор информации
   3. Тестирование управления конфигурацией и развертыванием
   4. Тестирование управления идентификацией
   5. Тестирование управления аутентификацией
   6. Тестирование управления авторизацией
   7. Тестирование управления сеансом
   8. Проверочное тестирование входных данных
      1. Тестирование отраженного межсайтового скриптинга
      2. Тестирование уязвимостей HTTP
      3. Тестирование приложения к различному виду инъекций (SQL, SSI, LDAP, XPATH, IMAP SMTP, а также кода)
      4. Тестирование на переполнение буфера
      5. Тестирование инкубированной уязвимости
   9. Тестирование на обработку ошибок
   10. Тестирование криптографии
   11. Тестирование бизнес-логики
   12. Тестирование на стороне клиента
4. Отчетность [4]

## PTF

Методология PTF ― Penetration Testing Framework является детальным техническим руководством по проведению тестирования на проникновение в технической части. Данное руководство не содержит общетеоретической информации, подобно OSSTMM или ISSAF, однако предоставляет практически исчерпывающий перечень уязвимостей подлежащих проверке, в некоторых случаях, с указанием рекомендуемого порядка проведения тестирования и инструментария для него.

Материал PTF организован в виде следующих разделов:

1. разведка ― сбор информации об объекте за счет проведения активного и пассивного мониторинга, поиска информация в сети Интернет, с использованием информационно-психологических воздействий и способов социальной инженерии;
   1. анализ ОС и ПО объекта;
   2. анализ портов сети;
   3. проверка паролей;
   4. проверка уязвимостей:
   5. уязвимости удаленного доступа;
   6. уязвимости внутреннего доступа;
   7. уязвимости bluetooth;
   8. уязвимости телекоммуникационного оборудования Cisco;
   9. уязвимости системы Citrix;
   10. уязвимости транспортных сетей;
2. Непосредственно тестирование на проникновение ― использование выявленных уязвимостей для реализации целенаправленных атак в отношении объекта тестирования. При этом отдельно рассмотрены:
   1. проникновение на сервера;
   2. оценка устойчивости VoIP-инфраструктуры;
   3. проникновение через беспроводные сети;
   4. безопасность физической инфраструктуры.
3. Формирование итогового отчета. [1]

## PTES

Стандарт проведения тестирования на проникновение PTES ― Penetration Testing Execution Standard был разработан в 2009 г. международной группой независимых экспертов-энтузиастов в области ИБ. PTES в качестве стандарта официально зарегистрирован только в США. Версия 1.0 была опубликована 16 августа 2014 года. Как уверяют её авторы, до этого она проходила тестирование на реальных объектах не менее года, а именно с мая 2012 года. С момента появления стандарт получил развитие в виде версии 1.1, вышедшей 16 июня 2021 года, которая просто расширяет некоторые аспекты первой публикации.

Стандарт предусматривает 7 основных этапов проведения тестирования на проникновение, описанных в соответствующих разделах:

1. этап первоначального общения ― инструментарий тестирования:
   1. ОС и ПО;
   2. аппаратные средства;
   3. радиотехнические средства;
2. сбор информации:
   1. разведка по открытым источникам (OSINT);
   2. использование информационно-психологических воздействий и социальной инженерии;
   3. анализ социальных сетей и контактов;
   4. анализ email и телефонных контактов;
   5. сканирование сети;
   6. анализ используемого ПО и ОС;
   7. анализ периметра безопасности;
   8. анализ физической инфраструктуры;
3. моделирование угроз;
4. анализ уязвимостей:
   1. анализ уязвимостей ОС и ПО;
   2. анализ уязвимостей БД;
   3. анализ уязвимостей VPN;
   4. анализ уязвимостей транспортной сети и сетевых протоколов;
   5. анализ уязвимостей беспроводной сети;
   6. анализ уязвимостей Интернет-подключений;
   7. анализ уязвимостей аутентификации и криптоустойчивости паролей;
   8. формирование целевых атак компьютерной разведки;
5. эксплуатация ― формирование средств, ориентированных на проникновеннее в защищаемый периметр организации за счет эксплуатации выявленных уязвимостей:
   1. атаки на ОС и ПО;
   2. атаки на аутентификацию по стандартным паролям;
   3. атаки на сетевые протоколы;
   4. атаки на VPN;
   5. DOS-атаки;
   6. атаки на протоколы семейства WEP и WPA беспроводных сетей;
   7. атаки на шлюзы с сетью Интернет;
   8. информационно-психологические воздействия и социальная инженерия;
   9. атаки на инфраструктуру контроля периметра (видеонаблюдение, пропускная система, учет передвижений персонала и т.д.);
6. постэксплуатация ― формирование атакующих средств, ориентированных на расширение привилегий и несанкционированные действия после проникновения за защищаемый периметр организации;
   1. эксплуатация уязвимостей ОС, ПО и БД:
   2. формирование закладок и уязвимостей для последующей эксплуатации;
   3. получение доступа и работа с системными файлами;
   4. получение доступа к важным файлам;
   5. получение доступа к информации авторизации пользователей;
   6. обход внутренних средств защиты
7. отчетность [9]

Данные этапы охватывают практически все действия, связанные с пентестом от первоначального общения и обоснования задания на тестирование до этапа формирования отчёта, в котором весь процесс фиксируется наиболее эргономичным для заказчика образом, а также формируются рекомендации по повышению защищённости тестируемой системы.

## Positive Technologies

Данная методология тестирования разработана в рамках услуги Pentest365 российской компанией Positive Technologies, являющейся одной из ведущих в области ИБ. Компания специализируется на комплексном аудите ИБ, оценке защищенности прикладных систем и веб-приложений, тестировании на проникновение и внедрении процессов мониторинга ИБ. Цели пентестинга:

* обоснование необходимости проведения работ по повышению уровня защищенности информационной системы;
* получение независимой оценки уровня безопасности информационной системы.

На этапе планирования определяются области тестирования и режимы проведения тестов. При этом, проведение тестов может проводиться как с уведомлением персонала объекта, так и без него. Данная методика предусматривает три варианта тестов на проникновение, различающихся между собой акцентом на различных типах уязвимостей.

1. Технический тест:
   1. получение предварительной информации о сети заказчика. Используются те же источники информации, которые доступны злоумышленникам (Интернет, новости, конференции);
   2. анализ сети, определение типов устройств, ОС, ПО посредствам реакции на внешние информационно-технические воздействия;
   3. выявление уязвимостей сетевых служб и ПО;
   4. анализ веб-приложений заказчика, проверка следующих уязвимостей:
      1. внедрение операторов SQL (SQL Injection);
      2. межсайтовое исполнение сценариев (Cross-Site Scripting);
      3. подмена содержимого (Content Spoofing);
      4. выполнение команд ОС (OS Commanding);
      5. уязвимости, связанные с некорректной настройкой механизмов аутентификации и авторизации и пр.;
   5. эксплуатации выявленных уязвимостей. Методы и инструментарий выбираются индивидуально для каждого типа уязвимости. Используются как общедоступные утилиты, так и инструментарий собственной разработки;
   6. анализ защищенности беспроводных сетей;
   7. анализ устойчивости внешнего периметра объекта и открытых ресурсов на атаки типа отказа в обслуживании (DDOS-атаки);
   8. анализ устойчивости сети к атакам на канальном уровне (нарушение функционирования протоколов канального уровня: STP, VTP, CDP, ARP);
   9. анализ сетевого трафика на предмет выявления утечек и фактов хранения и передачи важной информации (пароли пользователей, конфиденциальные документы и пр.);
   10. анализ устойчивости сетевой маршрутизации путем реализации атак направленных на фальсификацию маршрутов и проведения DDOS атак против используемых протоколов маршрутизации;
   11. анализ возможности получения злоумышленником несанкционированного доступа к конфиденциальной информации или информации ограниченного доступа, путем проверки прав доступа к различным информационным ресурсам с привилегиями, полученными на различных этапах тестирования;
   12. документирование полученной в ходе тестирования информации, ее анализ с целью выработки рекомендаций по улучшению защищенности сети.
2. Социотехнический тест:
   1. рассылка сообщений от имени анонимных пользователей и сотрудников организации, содержащих ссылки на веб-ресурсы с исполняемым кодом, содержащие исполняемый код в теле письма, содержащие просьбу сменить пароли, переслать пароли или свою персональную информацию и пр.;
   2. выборочная проверка исполнения политики «чистого стола» (стикеры с паролями, незаблокированные в отсутствие пользователя консоли, наличие конфиденциальных документов в офисе, доступных посетителям, оставленные без присмотра сотовые телефоны и пр.);
   3. звонки пользователям от имени персонала ИТ и ИБ отделов с просьбами получения/смены пароля, пересылки конфиденциальных документов и пр.;
   4. выбор целевых групп пользователей и определение методов социальной инженерии для тестирования каждой из групп;
   5. использование полученных в результате предыдущих этапов привилегий для получения несанкционированного доступа к ресурсам тестируемого объекта;
   6. документирование полученной в ходе тестирования информации, ее анализ с целью выработки рекомендаций по улучшению защищенности сети.
3. Комплексный тест. Комплексный тест на проникновение наиболее близок к реальным действиям злоумышленников. Используя различные вышеуказанные мероприятия технического и социоинженерного тестов, аудиторы пытаются обойти существующие защитные механизмы с целью выполнения поставленных заказчиком задач. Данную методику рекомендуется использовать для тестирования конечного продукта, уже введенного в эксплуатацию. [5]

# анализ рассмотренных методологий

В целом все рассмотренные выше методологии имеют много сходств между собой: в них описан один и тот же алгоритм проникновения в систему, рассматриваются примерно одинаковые аспекты на одних и тех же уровнях. Поэтому прежде чем перейти к более глубокому анализу каждой из представленных методик, хотелось бы затронуть вопрос актуальности. Данный аспект является крайне важным, поскольку мы живём в мире быстро меняющихся и развивающихся технологий во всех сферах науки, в частности в IT-сфере. Необходимо ориентироваться на эти изменения, чтобы обеспечить наиболее качественное тестирование, отвечающее запросам текущего времени.

Таблица 1 ― Обновляемость методологий

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Методология | OSSTMM | ISSAF | OWASP | PTF | PTES | PT |
| Дата последнего обновления | 2010 | 2006 | 2020 | ? | 2021 | ? |
| Всего обновлений | 3.0 | Draft 0.2.1 | 4.1 | ? | 1.1 | ? |

Методология ISSAF хоть и является самой объемной из всех представленных в работе, тем не менее, она была выпущена 15 лет назад и никаких обновлений найти не удалось. Это говорит о том, что её полнота и подробность актуальна лишь для устаревших компьютерных сетей и не отвечает требованиям современного мира информационных технологий.

OSSTMM также достаточна старая методология, она не обновлялась с 2010 года, хотя у неё было как минимум три полноценные версии, не считая промежуточных. Таким образом, данную методологию тоже можно использовать только для тестирования компьютерных сетей, которые не имеют в себе каких-либо новейших технологий.

Наиболее актуальными являются PTES и OWASP. Первую из них можно считать самой свежей, её содержание можно назвать исчерпывающим. Однако она обновляется не так часто, как хотелось бы.

OWASP также является достаточно актуальной методологией, к тому же она обновляется намного чаще. То есть тестировщикам не нужно самостоятельно думать, как проверить корректность того или иного элемента в сети, скорей всего это уже описано в методике. При этом данная методология больше ориентирована именно на тестирование веб-приложений, в OWASP отсутствует тестирование социальной инженерии, физической инфраструктуры и беспроводных сетей. Можно даже сказать, что эта методология больше про тестирование непосредственно кода.

Таблица 2 ― Сравнение содержания методологий

| Характеристика | OSSTMM | ISSAF | OWASP | PTES | PTF | Positive Technologies |
| --- | --- | --- | --- | --- | --- | --- |
| Сбор информации | + | + | + | + | + | + |
| Анализ и оценка уязвимостей | + | + | + | + | + | + |
| Описание этапов тестирования | + | + | + | + | + | + |
| Тестирование телекоммуникационных сетей | + | + | + | + | + | + |
| Тестирование беспроводных сетей | + | + | – | + | + | + |
| Тестирование веб-приложений | + | + | + | + | + | + |
| Проверка безопасности физической инфраструктуры | + | + | – | + | + | – |
| Проверка безопасности паролей | – | + | + | + | + | + |
| Проверка безопасности баз данных | – | + | + | – | – | – |
| Проверка безопасности исходного кода | – | + | + | – | – | – |
| Конкретные атаки на объект | – | + | + | + | + | + |
| Социальная инженерия | + | + | – | + | + | + |
| Рекомендации по конкретному ПО для тестирования | – | + | + | + | + | – |
| Анализ и рекомендации по устранению найденных уязвимостей | – | + | – | – | – | – |
| Всего | 8 | 14 | 10 | 11 | 11 | 9 |

Помимо актуальности хочется также отдельно выделить тестирование социальной инженерии, поскольку человек является самым слабым звеном в системе. Любой объект будь то hard- или software можно спокойно протестировать, выявить уязвимости, и защитить, а с людьми ситуация обстоит сложнее. Поэтому автор считает, что в каждой методологии тестирования должно быть как минимум напоминание о тестировании средствами социальной инженерии, а как максимум конкретные механизмы тестирования. В этом вопросе больше всех преуспела методология, разработанная компанией Positive Technologies, ведь у них данный вариант тестирования выделен в отдельный блок и расписан максимально подробно. Такое тестирование поможет выявить пробелы в знаниях сотрудников и подготовить образовательную лекцию на тему информационной безопасности.

Теперь после подсчёта всех «+» в таблице оказалось, что ISSAF самая обширная методология, несмотря на то, что она уже морально устарела. Второе место делят PTES и PTF, а на третьем месте оказалась OWASP. Несмотря на это предлагается более подробно рассмотреть именно OWASP. Она представляет особый интерес для автора по причине её новизны.

# owasp

## Описание проекта OWASP

Во введении к документу предлагается ответить на ряд вопросов о тестировании. Таким образом, ставятся ключевые цели и задачи тестирования.

### Что такое тестирование?

Тестирование ― это сравнение некоторых состояний системы или приложения относительно некоторых критериев. Документ обещает доступное объяснение процессов тестирования даже для тех людей, которые далеки от информационной безопасности и не обладают глубокими знаниями в этой сфере.

### Зачем нужно проводить тестирование?

Документ призван помочь организациям понять, что включает в себя программа тестирования, и помочь им определить шаги, необходимые для создания и эксплуатации программы тестирования веб-приложений. Это руководство можно использовать как справочное руководство и как методологию, помогающую определить разрыв между существующими практиками и лучшими отраслевыми практиками. Изучив весь этот материал, организация сможет сравнить свои характеристики с коллегами по отрасли, понять объем ресурсов, необходимых для тестирования и обслуживания программного обеспечения, или подготовиться к аудиту.

### Когда проводить тестирование?

Разработчики методологии уверены, что тестирование должно проводиться на каждом этапе жизненного цикла веб-приложения. Именно такой подход будет наиболее эффективным и наименее дорогостоящим. На Рисунке 1 представлена зависимость стоимости исправления ошибок от этапа жизненного цикла приложения.

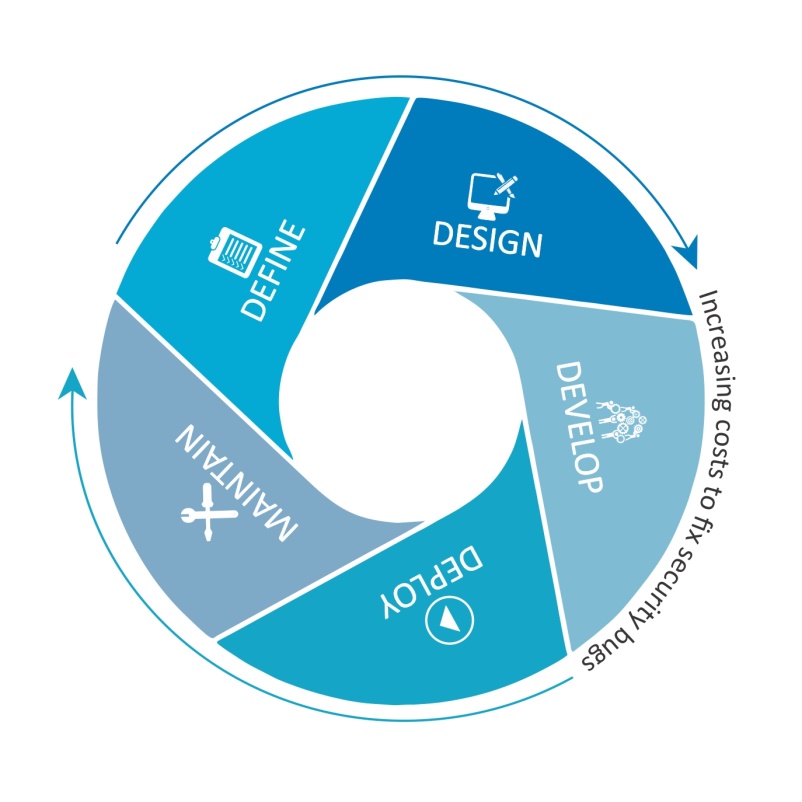


Рисунок 1 ― Зависимость стоимости исправления ошибок от этапа жизненного цикла приложения

Действительно, если приложение уже полностью готово, затем оно будет тестироваться в первый раз, то будет найдено очень много ошибок на каждом уровне ЖЦ. При исправлении ошибок на самом первом уровне, придется переделывать все остальные уровни практически заново. Таким образом, работа будет выполняться намного дольше, что экономически невыгодно.

### Что тестировать?

Авторы методологии выделают три основных вектора тестирования:

* Персонал: обладают ли работники достаточными компетенциями?
* Процесс: адекватность политик и стандартов, а также могут ли сотрудники следовать этим политикам?
* Технологии: насколько эффективен процесс?

Разработчики убеждены, что только такое комплексное тестирование поможет отследить все проблемы, которые в будущем могут повлиять на состояние конечного продукта.

## Принципы тестирования

Далее авторы рассматривают несколько важных правил, которые стоит помнить при проведении тестирования веб-приложения.

1. Не существует единого механизма, который бы обеспечил 100% защиту. Специализированное ПО помогает увидеть лишь верхушку айсберга, для полноценной безопасности необходим более скрупулёзный подход. Помните, что безопасность это процесс, а не продукт;
2. Мыслить стратегически, а не тактически;
3. Придерживаться жизненного цикла приложения;
4. Тестируйте как можно раньше и чаще;
5. Важно иметь понимание особенностей и количества защиты для конкретного приложения;
6. Развитие креативного образа мышления. Стандартные тесты направлены на действия обычного юзера, использующего приложение по назначению. Необходимо выходить за рамки и мыслить как злоумышленник, который пытается сломать ПО;
7. Документация должна быть максимально подробной и отражать в себе не только ожидаемое использование приложения, но и конкретные запрещённые варианты использования;
8. Грамотное использование вспомогательного инструментария;
9. «Дьявол в деталях»;
10. Использование исходного кода по возможности;
11. Отработка показателей безопасности для оценки эффективности вводимых изменений и планирования дальнейших действий;
12. Документация результатов тестирования.

## Техники, используемые для тестирования

Авторы методики представляют 4 основные техники, которые можно использовать для построения программы тестирования (Таблица 3). Для каждой техники представлено краткое описание её особенностей, а также достоинства и недостатки.

Таблица 3 ― Техники тестирования и их особенности

|  |  |  |  |
| --- | --- | --- | --- |
| Техника | Описание | Плюсы | Минусы |
| Анализ руководства и отзывов персонала | Мощный и эффективный инструмент для понимания, насколько сотрудники понимают специфику своего продукта и его жизненного цикла. Могут ли они ответить на вопросы «Как это работает?» и «Почему это именно так реализовано?»? Общение с персоналом поможет выявить потенциальные «дыры» в безопасности продукта | Не требует специального оборудования | Может занимать слишком много времени |
| Может быть применён в ряде ситуаций | Вспомогательный материал не всегда доступен |
| Гибкость | Требует особых софт-скиллов для максимальной эффективности |
| Улучшает командную работу |
| Ранний этап ЖЦ |
| Моделирование угроз | 1. Декомпозиция приложения с целью понимания его работы; 2. Классификация ресурсов и ранжирование их по важности; 3. Нахождение потенциальных уязвимостей; 4. Выявление потенциальных атак; 5. Создание стратегий смягчения последствий. | Практическое видение системы глазами нарушителя | Относительно новая техника |
| Гибкость | Хорошее моделирование угроз не означает хорошее приложение на выходе |
| Ранний этап ЖЦ |
| Анализ исходного кода | В некоторых случаях именно такой анализ является наиболее эффективным. Могут быть обнаружены проблемы распараллеливания, ошибки в бизнес-логике, проблемы контроля доступа и слабая криптографические, а также бэкдоры, трояны, пасхальные яйца, бомбы замедленного действия, логические бомбы и другие формы вредоносного кода. | Полнота и эффективность | Требуется высоко квалифицированный специалист |
| Точность | Можно пропустить проблемы в библиотеках |
| Скорость | Сложно определить ошибки во время выполнения |
| Фактически развернутый исходный код может отличаться от анализируемого |
| Тестирование на проникновение | Так называемый этичный хаккинг. Тестирование проводится по принципу черного ящика: у злоумышленника нет доступа к системе, поэтому он сначала собирает информацию, а затем атакует, то есть пытается проникнуть в систему с целью прослушивания канала или даже нанесения ущерба. | Быстро и дёшево | Поздний этап в ЖЦ |
| Требует меньшей квалификации, чем анализ кода | Тестирование только «в лоб» |
| Тестирует уже готовый код |

Важно построить такую модель тестирования, которая бы отвечала требованиям конкретного продукта. Каждая из вышеперечисленных техник должна применяться грамотно как с точки зрения безопасности веб-приложения, так и с точки зрения имеющихся у компании ресурсов, в частности экономических. Рекомендуется использовать все техники в совокупности, однако бывают случаи, когда такой возможности нет. Например, приложение уже готово к выпуску. Тогда стоит провести хотя бы пентест, чем не тестировать ничего вообще. По опыту компании уделяют особое внимание тестированию именно на ранних этапах жизненного цикла приложения, поэтому баланс распределения ресурсов на тестирование должен выглядеть примерно, как на Рисунке 2.

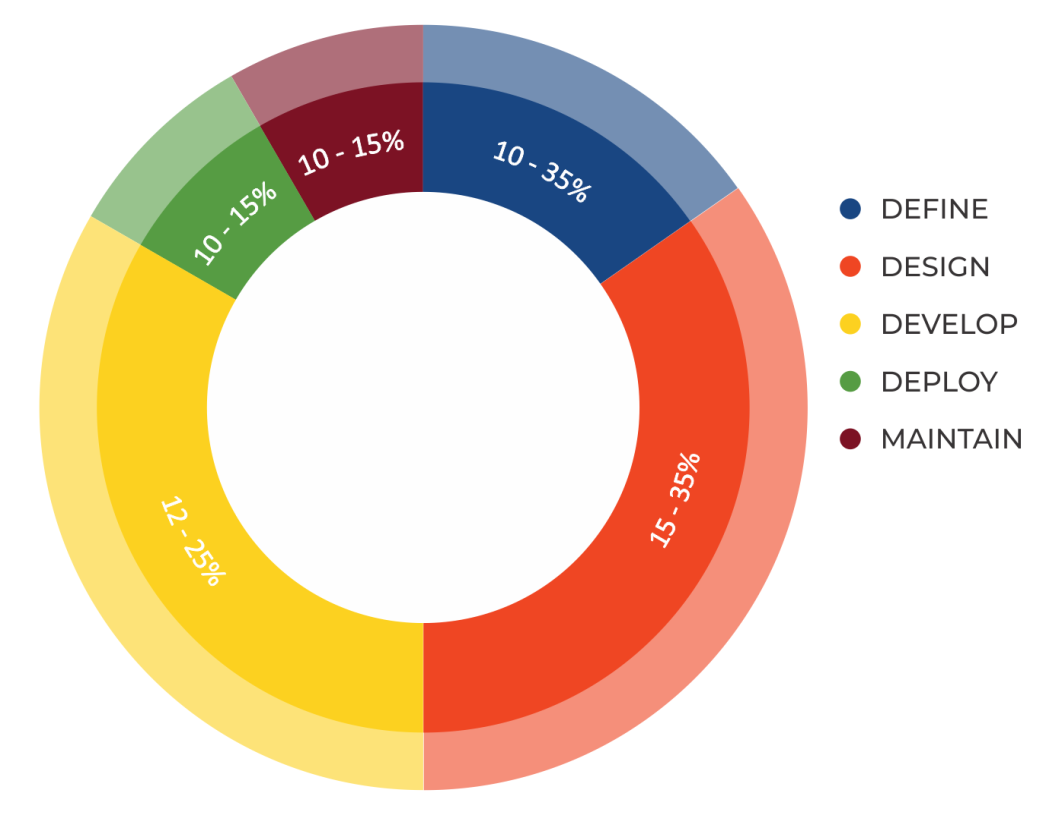


Рисунок 2 ― Распределение ресурсов при тестировании

Также разработчики методологии предложили диаграмму, показывающую приблизительную долю использования каждой техники тестирования (Рисунок 3).

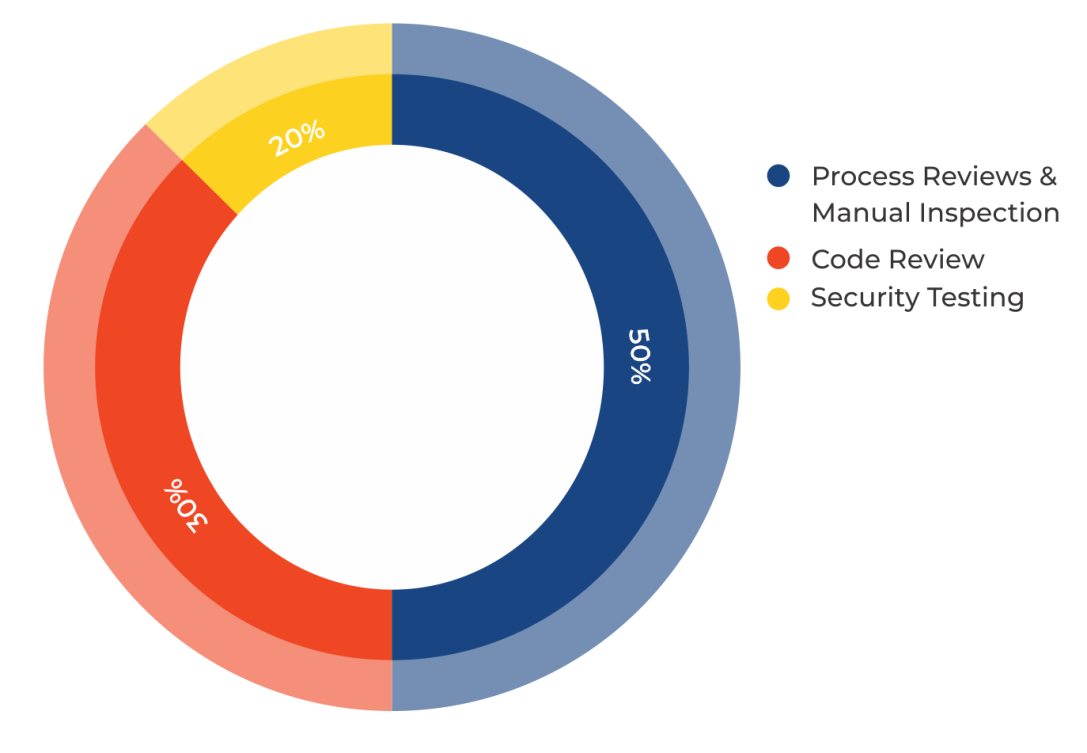


Рисунок 3 ― Использование техник тестирования

## Получение требований к тестированию

### Цели тестирования

Для успешного проведения тестирования необходимо чёткое понимание целевых объектов. Одной из основных целей является проверка того, что все введенные механизмы защиты работают должным образом. Данный аспект документируется требованиями безопасности. Иными словами, на высоком уровне это означает конфиденциальность, целостность и доступность данных. Другая цель ― подтвердить, что меры безопасности реализованы с небольшим количеством уязвимостей или вообще без них. Общий список уязвимостей представлен в OWASP Top Ten. Также рассматриваются уязвимости, которые были ранее выявлены с помощью оценок безопасности, таких как моделирование угроз, анализ исходного кода и тестирование на проникновение.

### Документация требований безопасности

Для грамотной документации необходимо прежде всего понимать бизнес-требования. Именно в них отражён основной функционал приложения, а также можно сделать выводы о том, как обращаться с пользовательскими данными. Важным аспектом является понимание, в каком бизнес-секторе и в какой стране будет реализовываться веб-приложение. Например, для финансовых приложений накладываются дополнительные требования по снижению рисков слабой аутентификации и рекомендуется делать её многоуровневой. Также накладывается запрет на хранение некоторых данных банковской карты пользователя. Кроме того, каждая организация имеет свою политику безопасности и свои стандарты, которые также стоит учитывать.

### Проверка требований безопасности

С точки зрения функциональности и управления рисками проверка требований безопасности является основной целью тестирования. На высоком уровне имеется в виду выявление пробелов в средствах управления безопасностью, обеспечивающих конфиденциальность, целостность и доступность данных. Допустим, в приложении хранятся некоторая личная информация пользователя. Тогда необходимо проверить, что она хранится в зашифрованном виде, при чем, алгоритмы шифрования и длины ключей должны соответствовать стандартам шифрования организации. Требования безопасности могут быть проверены на разных этапах SDLC с использованием различных артефактов и методологий тестирования.

### Таксономия угроз и контрмер

Классификация угроз и мер противодействия должна учитывать первопричины уязвимостей. Угроза может быть классифицирована с помощью STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege): спуфинг, фальсификация, отказ, раскрытие информации, отказ в обслуживании и превышение привилегий. Классификация угроз и мер противодействия уязвимостям также может использоваться для документирования стандартов безопасного написания кода. Пример распространенной ошибки состоит в применении хеш-функции для шифрования пароля без применения начального значения.

### Тестирование безопасности и анализ рисков

Требования безопасности должны учитывать серьезность уязвимостей для поддержки стратегии снижения рисков. Проблемы безопасности в базе данных об уязвимостях могут быть представлены по типу, проблеме, смягчению последствий, первопричине и сопоставлены с приложениями, в которых они были найдены. Такая база также может быть использована для создания метрик для анализа эффективности тестов безопасности на протяжении всего SDLC. При рассмотрении сценариев угроз использования распространенных уязвимостей, можно определить потенциальные риски безопасности. Например, OWASP Top Ten включают фишинг, нарушение конфиденциальности, кражу личных данных, компрометацию системы, изменение или уничтожение данных, финансовые потери и потерю репутации. Если мыслить категориями угроз и уязвимостей, база знаний организации об уязвимостях может быть использована для составления тестовых примеров с учетом рисков безопасности для проверки наиболее вероятных сценариев атак.

### Выведение функциональных и нефункциональных требований к тестированию

#### Функциональные требования безопасности

Применимые стандарты, политики и правила определяют как необходимость в том или ином типе контроля безопасности, так и функциональность контроля. Подобные требования называют позитивными, поскольку они определяют ожидаемую функциональность, которая может быть подтверждена с помощью тестов безопасности. Положительными требованиями к приложению является выполнение заданного заказчиком функционала. Тогда проверка положительных требований может быть проведена путем выполнения теста в соответствии с заранее определенными входными данными. В таком случае требования безопасности должны быть функционально-ориентированными: они должны подчеркивать ожидаемую функциональность (что) и подразумевать реализацию (как).

#### Требования безопасности с учетом рисков

Тесты безопасности также должны быть ориентированы на риск, проверяя приложение на наличие неожиданного поведения или негативных требований. Примеры таких негативных требований:

* Приложение не должно позволять изменять или уничтожать данные;
* Приложение не должно быть взломано или использовано злоумышленником для несанкционированных финансовых операций.

Негативные требования сложнее протестировать, поскольку в них нет ожидаемого поведения. Ключевым моментом является документирование сценариев потенциальных угроз и функциональности контрмер. Для получения негативных сценариев тестирования могут быть полезны такие инструменты моделирования угроз, как деревья угроз и библиотеки атак.

### Выведение требований к тестированию безопасности с помощью примеров корректного и некорректного использования

Основная цель состоит в описании наиболее важных сценариев использования и злоупотребления приложением. Сценарии злоупотреблений позволяют проанализировать приложение с точки зрения злоумышленника и способствуют выявлению потенциальных уязвимостей и контрмер. Для этого важно определить функциональные и негативные сценарии и представить их в графической форме. В качестве примера приведем пошаговую методику для случая выведения требований безопасности для аутентификации.

Шаг 1: Опишите функциональный сценарий;

Шаг 2: Опишите негативный сценарий;

Шаг 3: Опишите функциональные и негативные сценарии с корректным и некорректным использованием случая;

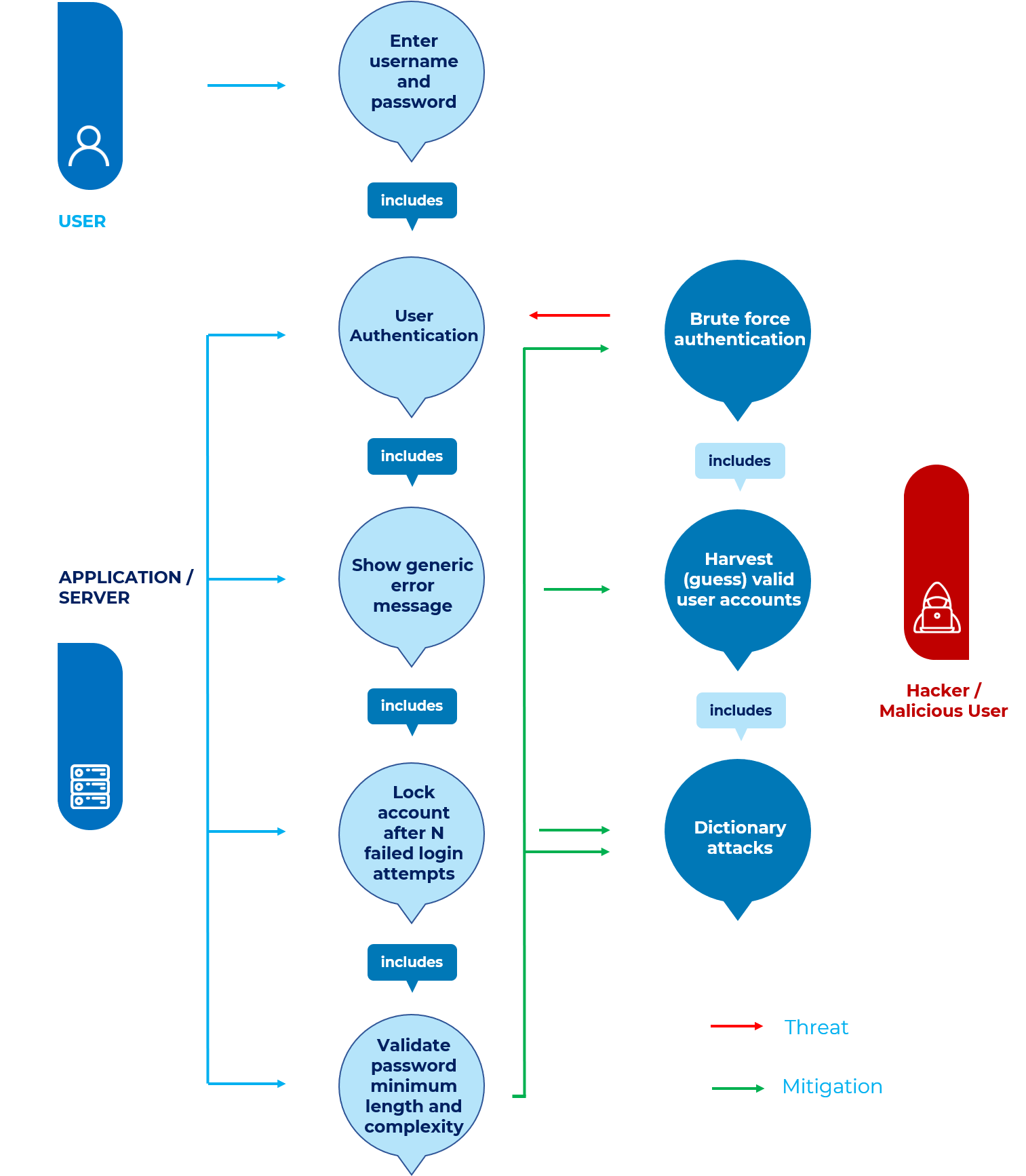


Рисунок 4 ― Пример

Шаг 4: Выявление требований безопасности.

В данном примере можно выявить следующие требования безопасности для аутентификации:

1. Требования к паролям должны соответствовать действующим стандартам по достаточной сложности.
2. Учетные записи должны быть заблокированы после пяти неудачных попыток входа в систему.
3. Сообщения об ошибках при входе в систему должны быть общими.

Эти требования должны быть документированы и протестированы.

## Тесты безопасности, интегрированные в рабочие процессы разработки и тестирования

### Тестирование безопасности в рабочем процессе разработки

Тестирование безопасности на этапе разработки SDLC представляет собой первую возможность убедиться, что разработанные отдельные компоненты программного обеспечения прошли проверку на безопасность до того, как они будут интегрированы с другими компонентами и встроены в приложение. Программные компоненты могут состоять из программных артефактов, таких как функции, методы и классы, а также интерфейсы прикладного программирования, библиотеки и исполняемые файлы. При тестировании безопасности разработчики могут полагаться на результаты анализа исходного кода для статической проверки того, что разработанный исходный код не содержит потенциальных уязвимостей и соответствует стандартам безопасного кодирования. Модульные тесты безопасности могут дополнительно проверить динамически, что компоненты функционируют так, как ожидается. Прежде чем интегрировать новые и существующие изменения кода в сборку приложения, необходимо просмотреть и подтвердить результаты статического и динамического анализа.

Этот рабочий процесс безопасного обзора кода может быть обеспечен посредством формальной приемки, а также проверки в инструменте управления рабочим процессом.

### Тестирование безопасности в рабочем процессе тестирования

После того как компоненты и изменения кода протестированы разработчиками и внесены в сборку приложения, следующим шагом в рабочем процессе разработки программного обеспечения являются интегрированное тестирование и тестированием на уровне системы. Когда тесты безопасности являются частью этих видов тестирования, они могут быть использованы для проверки как функциональности безопасности приложения в целом, так и уязвимостей на уровне приложения.

Важно, чтобы проблемы были устранены в соответствии с рекомендациями команды тестирования, включающими изменение кода, дизайна или конфигурации. На этом уровне аудиторы безопасности и сотрудники службы информационной безопасности обсуждают сообщенные проблемы безопасности и анализируют потенциальные риски в соответствии с процедурами управления информационными рисками.

### Тесты безопасности для разработчиков

#### Тестирование безопасности на этапе написания кода: Модульные тесты

С точки зрения разработчика, основная цель тестов безопасности ― подтвердить, что код разрабатывается в соответствии с требованиями стандартов безопасности, которые должны быть задокументированы в стандартах и подтверждены с помощью статического и динамического анализа.

Хорошей практикой для разработчиков является создание тестовых случаев в виде общего набора тестов безопасности, который является частью существующей структуры модульного тестирования. Он может быть получен из ранее определенных случаев использования и неправильного использования для тестирования функций, методов и классов безопасности и может включать тестовые примеры для проверки как положительных, так и отрицательных требований к средствам контроля безопасности, таких как:

* Идентификация, аутентификация и контроль доступа;
* Валидация и кодирование ввода;
* Шифрование;
* Управление пользователями и сеансами;
* Обработка ошибок и исключений;
* Аудит и протоколирование.

Разработчики, обладающие инструментом анализа исходного кода, интегрированным в IDE, стандартами безопасного написания кода и системой модульного тестирования безопасности, могут оценивать и проверять безопасность разрабатываемых программных компонентов. В случае функциональных тестов безопасности тесты на уровне модулей могут проверять функциональность элементов управления безопасностью на уровне компонентов программного обеспечения, таких как функции, методы или классы.

Сценарии угроз, выявленные с помощью примеров с корректного и некорректного использования, можно использовать для документирования процедур тестирования компонентов программного обеспечения.

На уровне компонентов модульные тесты безопасности могут проверять положительные и отрицательные утверждения, такие как ошибки и обработка исключений. Безопасная обработка ошибок может подтвердить потенциальное раскрытие информации с помощью информативных сообщений об ошибках и трассировки стека.

Анализ исходного кода и модульные тесты могут подтвердить, что изменение кода устраняет уязвимость, найденную ранее.

### Тесты безопасности функциональных тестировщиков

#### Тестирование безопасности на этапе интеграции и валидации: Интегрированные системные тесты и эксплуатационные тесты

Основной целью интегрированных системных тестов является подтверждение того, что реализация средств контроля безопасности обеспечивает безопасность на разных уровнях. Среда тестирования интеграционной системы также является первой средой, где тестировщики могут имитировать реальные сценарии атак, которые потенциально могут быть выполнены злонамеренным внешним или внутренним пользователем приложения. Реальные сценарии атак могут быть проверены как с помощью методов ручного тестирования, так и с помощью инструментов тестирования на проникновение. Целью данного тестирования является сборка приложения, которая представляет собой версию приложения, развертываемого в производстве.

Включение тестирования безопасности в фазу интеграции и проверки имеет решающее значение для выявления уязвимостей, возникающих в результате интеграции компонентов, а также проверки подверженности таким уязвимостям. Тестирование безопасности приложений требует специализированного набора навыков, включая знания в области программного обеспечения и безопасности, которые не характерны для инженеров по безопасности. В результате организациям часто приходится обучать разработчиков программного обеспечения методам этического взлома, процедурам и инструментам оценки безопасности. Реалистичным сценарием является разработка таких ресурсов собственными силами и их документирование в руководствах и процедурах тестирования безопасности, которые учитывают знания разработчика в данной области.

## Анализ и отчетность по данным тестирования безопасности

### Цели для метрик и измерений при тестировании безопасности

Определение целей для метрик и измерений тестирования безопасности является необходимым условием для использования данных тестирования безопасности для анализа рисков и процессов управления

В традиционном тестировании ПО количество дефектов может служить показателем качества данного программного обеспечения. Аналогичным образом тестирование безопасности может служить показателем безопасности программного обеспечения. С точки зрения управления дефектами и отчетности, тестирование качества программного обеспечения и тестирование безопасности могут использовать аналогичную категоризацию для основных причин и усилий по устранению дефектов. С точки зрения первопричины, дефект безопасности может быть вызван ошибкой в проектировании или написании кода. С точки зрения усилий, необходимых для устранения дефекта, как дефекты безопасности, так и дефекты качества могут быть измерены с точки зрения часов, затраченных разработчиками на реализацию исправления, необходимых инструментов и ресурсов, а также стоимости реализации исправления.

Особенностью данного тестирования безопасности является категоризация с точки зрения угрозы, подверженности уязвимости и потенциального воздействия, оказываемого уязвимостью, для определения риска. Тестирование приложений на безопасность заключается в управлении техническими рисками, чтобы убедиться, что контрмеры приложения соответствуют приемлемым уровням. По этой причине данные тестирования безопасности должны поддерживать стратегию риска безопасности на критических контрольных точках в ходе SDLC. Мера риска для уязвимости может быть рассчитана путем определения факторов воздействия и вероятности, а также путем проверки уязвимости с помощью тестов на проникновение. Метрики риска, связанные с уязвимостями, обнаруженными с помощью тестов безопасности, позволяют руководству предприятия принимать решения по управлению рисками, например, решать, можно ли принять, уменьшить или перенести риски на разных уровнях в организации.

При оценке уровня безопасности приложения важно учитывать некоторые факторы, такие как размер разрабатываемого приложения. Статистически доказано, что размер приложения связан с количеством проблем, обнаруженных в приложении во время тестирования. Поскольку тестирование уменьшает количество проблем, логично, что приложения большего размера должны тестироваться чаще, чем приложения меньшего размера.

Когда тестирование безопасности проводится на нескольких этапах жизненного цикла, данные тестирования могут доказать способность тестов безопасности обнаруживать уязвимости сразу после их появления. Данные тестирования также могут доказать эффективность устранения уязвимостей путем реализации контрмер на различных контрольных точках ЖЦ. Измерения такого типа также определяются как "метрики сдерживания" и позволяют оценить способность оценки безопасности, выполненной на каждом этапе процесса разработки, поддерживать безопасность в рамках каждого этапа. Эти метрики сдерживания также являются критическим фактором снижения стоимости устранения уязвимостей. Устранение уязвимостей на том же этапе ЖЦ, на котором они были обнаружены, обходится дешевле, чем их устранение на другом этапе.

Метрики тестирования безопасности могут поддерживать анализ рисков безопасности, затрат и управления дефектами, если они связаны с осязаемыми и определенными по времени целями, такими как:

* Сокращение общего числа уязвимостей на 30%;
* Устранение проблем безопасности к определенному сроку.

Данные тестирования безопасности могут быть абсолютными, а также сравнительными. Чтобы ответить на вопросы о качестве процесса обеспечения безопасности, важно определить базовый уровень того, что можно считать приемлемым и хорошим.

Данные тестирования безопасности могут также поддерживать конкретные цели анализа безопасности. Этими целями могут быть соответствие нормативным требованиям по безопасности и стандартам информационной безопасности, управление процессами безопасности, выявление первопричин безопасности и улучшение процессов, а также анализ выгоды от затрат на безопасность.

Когда данные тестирования безопасности сообщаются, они должны обеспечивать метрики для поддержки анализа. Сфера анализа ― это интерпретация данных тестирования для поиска сведений о безопасности производимого программного обеспечения, а также об эффективности процесса. Чтобы вынести обоснованное суждение, используя данные тестирования, важно хорошо понимать процесс тестирования, а также инструменты тестирования. Важно отметить, что неизвестные проблемы безопасности не проверяются. Тот факт, что тест безопасности не выявил проблем, не означает, что программное обеспечение или приложение отвечает стандартам качества.

Даже самые сложные средства автоматизации не могут сравниться с опытным тестировщиком безопасности. Просто полагаясь на успешные результаты тестирования с помощью автоматизированных инструментов, специалисты по безопасности будут испытывать ложное чувство безопасности. Как правило, чем опытнее тестировщики безопасности в методологии тестирования безопасности и инструментах тестирования, тем лучше будут результаты тестирования и анализа безопасности. Важно, чтобы руководители, делающие инвестиции в инструменты тестирования безопасности, также рассматривали инвестиции в наем квалифицированных кадров, а также в обучение тестированию безопасности.

### Требования к отчетности

Уровень безопасности приложения может быть охарактеризован с точки зрения последствий, таких как количество уязвимостей и рейтинг риска уязвимостей, а также с точки зрения причины или происхождения, таких как ошибки в написании кода, архитектурные недостатки и проблемы конфигурации. Уязвимости можно классифицировать по различным критериям.

При представлении данных о тестировании безопасности лучше всего включать следующую информацию:

* категоризация каждой уязвимости по типам;
* угроза безопасности, которой подвергается каждый вопрос;
* первопричину каждой проблемы безопасности;
* каждый метод тестирования, используемый для поиска проблем;
* меры по устранению, или контрмеры, для каждой уязвимости;
* рейтинг серьезности каждой уязвимости.

Описав, в чем заключается угроза безопасности, можно понять, является ли контроль по снижению воздействия неэффективным и почему он неэффективен для снижения угрозы. Сообщение о первопричине проблемы может помочь точно определить, что необходимо исправить. После получения сообщений о проблемах важно также предоставить разработчику программного обеспечения руководство по повторному тестированию и поиску уязвимости.

Информация о том, как устранить уязвимость, должна быть достаточно подробной, чтобы разработчик смог реализовать исправление. В ней должны быть приведены примеры безопасного написания кода, изменения конфигурации и даны соответствующие ссылки. Наконец, оценка серьезности вносит вклад в расчет рейтинга риска и помогает определить приоритетность усилий по исправлению.

### Бизнес-кейсы

Чтобы метрики тестирования безопасности были полезными, они должны обеспечивать ценность для заинтересованных сторон. В число заинтересованных сторон могут входить руководители проектов, разработчики, отделы информационной безопасности, аудиторы и руководители информационных служб.

Разработчики программного обеспечения смотрят на данные тестов безопасности, чтобы показать, что программное обеспечение написано безопасно и эффективно, что позволяет им обосновать необходимость использования инструментов анализа исходного кода, следования стандартам и посещения тренингов по безопасности программного обеспечения.

Руководители проектов ищут данные, которые позволяют им успешно управлять и использовать мероприятия и ресурсы в соответствии с планом проекта. Для них данные тестирования безопасности могут показать, что проекты идут по графику, соблюдают сроки сдачи и становятся лучше в ходе тестирования.

Данные тестирования безопасности также помогают обосновать необходимость тестирования безопасности, если инициатива исходит от сотрудников по информационной безопасности.

Для аудиторов соответствия метрики тестирования безопасности обеспечивают уровень гарантии безопасности программного обеспечения и уверенность в том, что соответствие стандартам безопасности рассматривается в рамках процессов проверки безопасности в организации.

Наконец, руководители информационных служб и руководители служб информационной безопасности, отвечающие за бюджет, который необходимо выделить на ресурсы безопасности, стремятся получить анализ затрат и выгод на основе данных тестирования безопасности. Это позволяет им принимать обоснованные решения о том, в какие мероприятия и инструменты безопасности следует вкладывать средства. Одной из метрик, поддерживающих такой анализ, является возврат инвестиций в безопасность. Для получения таких показателей на основе данных тестирования безопасности важно определить разницу между риском, связанным с обнаружением уязвимостей, и эффективностью тестов безопасности в снижении риска безопасности, а затем соотнести этот разрыв со стоимостью мероприятий по тестированию безопасности или используемых инструментов тестирования.

Заключение

В ходе выполнения данной курсовой работы был рассмотрен ряд методологий тестирования безопасности веб-приложений. Поставленная цель была достигнута и выделенные задачи были решены. Были рассмотрены такие методологии как OSSTMM, ISSAF, OWASP, PTF, PTES, Positive Technologies. В ходе анализа OWASP и PTES оказались самыми свежими по дате обновления. Однако по количеству версий PTES значительно отстает от OWASP. Анализ по этим параметрам может показать, насколько современными являются принципы тестирования, а также актуальность возможных уязвимостей, изложенных в данных методологиях. Далее был проведён анализ непосредственно по содержанию методологий. Самой полной в этом плане оказалась ISSAF. На втором месте оказались PTES и PTF, а на третьем месте расположилась OWASP.

Данная методология оказалась единственной, которая не имеет подробного описания тестирования на социальную инженерию, однако во введении к документу упоминается о важности этого аспекта. Тем не менее, она является очень хорошим примером анализа именно технической части, в частности кода. В документе рассматриваются как общие техники тестирования, так и конкретные примеры уязвимостей, и способы борьбы с ними. Особое внимание уделяется подготовке отчётов по тестированию безопасности. Отчёты должны быть представлены как перед тестированием, так и по его результатам. Тестирование физической инфраструктуры также не представлено.

Как уже говорилось выше, ISSAF рассматривает все возможные уязвимости от анализа кода и пентестинга до социальной инженерии, однако у неё есть только одна версия, выпущенная в 2006 году, что ставит под вопрос актуальность и своевременность методологии.

Методология OSSTMM охватывает больше тестирование непосредственно сетей, в том числе оборудования и беспроводных сетей. Также рассматривается тестирование социальных процессов в организации. При этом совсем отсутствует тестирование исходного кода, безопасности паролей и баз данных. Не представлены рекомендации по конкретному ПО для тестирования и по устранению найденных уязвимостей.

Говоря о PTF, эта методология полнее OSSTMM, но в ней также совершенно не рассматривается безопасность исходного кода и данных. Данная методология основана на технике тестирования на проникновение. Представлен подробный сценарий проникновения в систему от сбора информации (в методологии уточнено, какая именно информация будет собрана), до подготовки отчета об имеющихся уязвимостях и способах их нивелирования.

Методология PTES по своему содержанию похожа на PTF. Она также основана на пентестинге и моделировании угроз. Сбор информации происходит из открытых источников (принцип OSINT).

Компания Positive Technologies представляет тестирование по трём методикам. Первая методика направлена больше на технический аспект. Проводится подробнейший анализ сети, проводятся попытки эксплуатации уязвимостей, связанных с SQL и XSS инъекциями, атаки на ПО, беспроводные сети и DDOS атаки. Физическая инфраструктура, исходный код и хранение данных не затрагиваются. Вторая методика тестирования приводит множество примеров уязвимостей социальной инженерии. Она тестирует только компетентность сотрудников по части информационной безопасности и больше ничего. Третья методика объединяет первую и вторую, то есть с помощью неё можно провести более комплексное тестирование.

Список использованных источников

1. Макаренко С. И., Смирнов Г. Е. Анализ стандартов и методик тестирования на проникновение // Системы управления, связи и безопасности. – 2020. – С. 50-56, 60-61.
2. Веб безопасность [Электронный ресурс]. URL: <https://developer.mozilla.org/ru/docs/Learn/Server-side/First_steps/Website_security>. – (дата обращения: 16.10.2021).
3. Современные методы исследования безопасности веб-приложений [Электронный ресурс]. URL: <https://habr.com/ru/company/owasp/blog/335820/>. – (дата обращения: 16.10.2021)
4. OWASP WSTG-v4.1 [Электронный ресурс]. URL: <https://owasp.org/www-project-web-security-testing-guide/v41/>. – (дата обращения: 19.11.2021).
5. Тестирование на проникновение [Электронный ресурс]. URL: <https://www.ptsecurity.com/ru-ru/services/pentest/>. – (дата обращения: 19.11.2021).
6. ISECOM. About us [Электронный ресурс]. URL: <https://www.isecom.org/about.html>. – (дата обращения: 22.11.2021).
7. Institute for Security and open Methodologies. OSSTMM 3 The Open Source Security Testing Methodology Manual – 2010. – С. 8-9.
8. ISECOM. Research [Электронный ресурс]. URL: <https://www.isecom.org/research.html#content5-9d>. – (дата обращения: 22.11.2021).
9. PTES Technical Guidelines [Электронный ресурс]. URL: <http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines> – (дата обращения: 23.11.2021).