ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ ВЕБ-ПРИЛОЖЕНИЙЙ

ЛЕКЦИЯ 1

БЕЗОПАСНОСТЬ В ИНТЕРНЕТЕ

Интернет теперь доступен едва ли не каждому. Многие хранят свою личную информацию в Сети, пользуются системами Интернет-банкинга, оплачивают услуги сотовой связи, зарабатывают деньги в Интернете.

Однако вместе с колоссальным ростом популярности Интернета возникает беспрецедентная угроза разглашения персональных данных, критически важных корпоративных ресурсов, государственных тайн и т.д. Каждый день хакеры подвергают угрозе эти ресурсы, пытаясь получить к ним доступ с помощью специальных атак. Эти атаки становятся все более изощренными и простыми в исполнении. Этому способствуют два основных фактора. Во-первых, это повсеместное проникновение интернета. Во-вторых, это всеобщее распространение простых в использовании операционных систем и сред разработки.

Приведем несколько определений:

Безопасность информации **(данных)** – состояние защищенности информации (данных), при котором обеспечены ее (их) конфиденциальность, доступность и целостность .

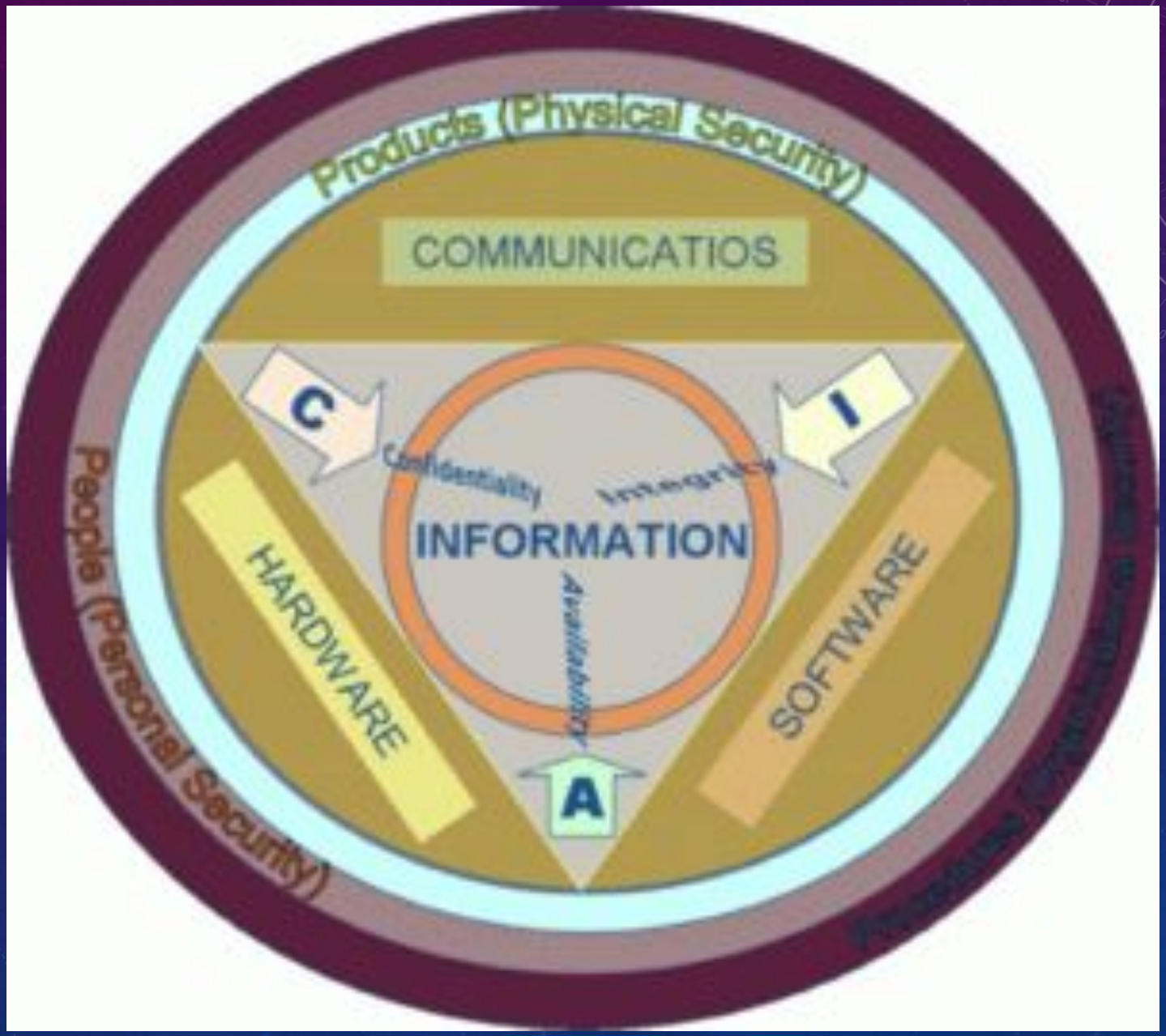
Информационная безопасность – защита конфиденциальности, целостности и доступности информации. В качестве стандартной модели безопасности часто приводят модель из трех категорий:

- конфиденциальность (*confidentiality*) – состояние информации, при котором доступ к ней осуществляют только субъекты, имеющие на него право ;
- целостность (integrity) – избежание несанкционированной модификации информации;
- доступность (availability) – избежание временного или постоянного *сокрытия информации* от пользователей, получивших права доступа .

Выделяют и другие не всегда обязательные категории модели безопасности:

- неотказуемость или апеллируемость (non-*repudiation*) – невозможность отказа от авторства;
- подотчетность (accountability) – обеспечение идентификации *субъекта доступа* и регистрации его действий;
- достоверность (reliability) – свойство соответствия предусмотренному поведению или результату;
- аутентичность или подлинность (authenticity) – свойство, гарантирующее, что субъект или ресурс идентичны заявленным.

Точками приложения процесса защиты информации к информационной системе являются аппаратное обеспечение, программное обеспечение и обеспечение связи (коммуникации). Сами процедуры (механизмы) защиты разделяются на защиту физического уровня, защиту персонала и организационный уровень


Products (Physical Security)
COMMUNICATIOS
C
I
Confidentiality
Integrity
INFORMATION
Availability
HARDWARE
SOFTWARE
A
People (Personal Security)

Системный подход к описанию информационной безопасности предлагает выделить следующие составляющие информационной безопасности:

- законодательная, нормативно-правовая и научная база;
- структура и задачи органов (подразделений), обеспечивающих безопасность ИТ;
- организационно-технические и режимные меры и методы (Политика информационной безопасности);
- программно-технические способы и средства обеспечения информационной безопасности.

Рассмотрим подробнее каждую из составляющих информационной безопасности:

ЗАКОНОДАТЕЛЬНАЯ, НОРМАТИВНО-ПРАВОВАЯ И НАУЧНАЯ БАЗА

В Российской Федерации к нормативно-правовым актам в области информационной безопасности относятся:

- Акты федерального законодательства:
  - Международные договоры РФ;
  - Конституция РФ;
  - Законы федерального уровня (включая федеральные конституционные законы, кодексы);
  - Указы Президента РФ;
  - Постановления правительства РФ;
  - Нормативные правовые акты федеральных министерств и ведомств;
  - Нормативные правовые акты субъектов РФ, органов местного самоуправления и т. д

К нормативно-методическим документам можно отнести

- Методические документы государственных органов России:
  - Доктрина информационной безопасности РФ;
  - Руководящие документы ФСТЭК (Гостехкомиссии России);
  - Приказы ФСБ;
- Стандарты информационной безопасности, из которых выделяют:
  - Международные стандарты;
  - Государственные (национальные) стандарты РФ;
  - Рекомендации по стандартизации;
  - Методические указания.

- https://webnvpks.github.io/files/proectirovanie_razrabotka_i_optimizacia_web_prilosheniy/lectures/security2.html

КЛАССИФИКАЦИЯ УГРОЗ БЕЗОПАСНОСТИ WEB-ПРИЛОЖЕНИЙ

Данная классификация представляет собой совместную попытку членов международного консорциума собрать воедино и упорядочить угрозы безопасности Web-сайтов. Члены Web Application Security Consortium создали данный проект для унификации стандартной терминологии описания угроз безопасности Web-приложений. Это даёт возможность разработчикам приложений, специалистам в области безопасности, производителям программных продуктов и аудиторам использовать единый язык при своем взаимодействии.

Данный документ содержит компиляцию и квинтэссенцию известных классов атак, которые представляют угрозы для Web-приложений. Каждому классу атак присвоено стандартное название и описаны его ключевые особенности. Классы организованы в иерархическую структуру.

1. АУТЕНТИФИКАЦИЯ (AUTHENTICATION)

Раздел, посвященный аутентификации, описывает атаки направленные на используемые Web-приложением методы проверки идентификатора пользователя, службы или приложения. Аутентификация использует как минимум один из трех механизмов (факторов): "что-то, что мы имеем", "что-то, что мы знаем" или "что-то, что мы есть". В этом разделе описываются атаки, направленные на обход или эксплуатацию уязвимостей в механизмах реализации аутентификации Web-серверов.

Краткое описание

- Подбор (Brute Force)автоматизированный процесс проб и ошибок, использующийся для того, чтобы угадать имя пользователя, пароль, номер кредитной карточки, ключ шифрования и т.д.
- Недостаточная аутентификация (Insufficient Authentication)эта уязвимость возникает, когда Web-сервер позволяет атакующему получать доступ к важной информации или функциям сервера без должной аутентификации
- Небезопасное восстановление паролей (Weak Password Recovery Validation)эта уязвимость возникает, когда Web-сервер позволяет атакующему несанкционированно получать, модифицировать или восстанавливать пароли других пользователей

2. АВТОРИЗАЦИЯ (AUTHORIZATION)

Данный раздел посвящен атакам, направленным на методы, которые используются Web-сервером для определения того, имеет ли пользователь, служба или приложение необходимые для совершения действия разрешения. Многие Web-сайты разрешают только определенным пользователям получать доступ к некоторому содержимому или функциям приложения. Доступ другим пользователям должен быть ограничен. Используя различные техники, злоумышленник может повысить свои привилегии и получить доступ к защищенным ресурсам.

Краткое описание

- Предсказуемое значение идентификатора сессии (Credential/Session Prediction)предсказуемое значение идентификатора сессии позволяет перехватывать сессии других пользователей
- Недостаточная авторизация (Insufficient Authorization)недостаточная авторизация возникает, когда Web-сервер позволяет атакующему получать доступ к важной информации или функциям, доступ к которым должен быть ограничен
- Отсутствие таймаута сессии (Insufficient Session Expiration)в случае если для идентификатора сессии или учетных данных не предусмотрен таймаут или его значение слишком велико, злоумышленник может воспользоваться старыми данными для авторизации
- Фиксация сессии (Session Fixation)используя данный класс атак, злоумышленник присваивает идентификатору сессии пользователя заданное значение

3. АТАКИ НА КЛИЕНТОВ (CLIENT-SIDE ATTACKS)

Этот раздел описывает атаки на пользователей Web-сервера. Во время посещения сайта, между пользователем и севером устанавливаются доверительные отношения, как в технологическом, так и в психологическом аспектах. Пользователь ожидает, что сайт предоставит ему легитимное содержимое. Кроме того, пользователь не ожидает атак со стороны сайта. Эксплуатируя это доверие, злоумышленник может использовать различные методы для проведения атак на клиентов сервера.

Краткое описание

- Подмена содержимого (Content Spoofing)используя эту технику, злоумышленник заставляет пользователя поверить, что страницы сгенерированны Web-сервером, а не переданы из внешнего источника
- Межсайтовое выполнение сценариев (Cross-site Scripting, XSS)наличие уязвимости Cross-site Scripting позволяет атакующему передать серверу исполняемый код, который будет перенаправлен браузеру пользователя
- Расщепление HTTP-запроса (HTTP Response Splitting)при использовании данной уязвимости злоумышленник посылает серверу специальным образом сформированный запрос, ответ на который интерпретируется целью атаки как два разных ответа

4. ВЫПОЛНЕНИЕ КОДА (COMMAND EXECUTION)

Эта секция описывает атаки, направленные на выполнение кода на Web-сервере. Все серверы используют данные, преданные пользователем при обработке запросов. Часто эти данные используются при составлении команд, применяемых для генерации динамического содержимого. Если при разработке не учитываются требования безопасности, злоумышленник получает возможность модифицировать исполняемые команды.

Краткое описание

- Переполнение буфера (Buffer Overflow)эксплуатация переполнения буфера позволяет злоумышленнику изменить путь исполнения программы путем перезаписи данных в памяти системы
- Атака на функции форматирования строк (Format String Attack)при использовании этих атак путь исполнения программы модифицируется методои перезаписи областей памяти с помощью функций форматирования символьных переменных
- Внедрение операторов LDAP (LDAP Injection)атаки этого типа направлены на Web-серверы, создающие запросы к службе LDAP на основе данных, вводимых пользователем
- Выполнение команд ОС (OS Commanding)атаки этого класса направлены на выполнение команд операционной системы на Web-сервере путем манипуляции входными данными
- Внедрение операторов SQL (SQL Injection)эти атаки направлены на Web-серверы, создающие SQL запросы к серверам СУБД на основе данных, вводимых пользователем
- Внедрение серверных сценариев (SSI Injection)атаки данного класса позволяют злоумышленнику передать исполняемый код, который в дальнейшем будет выполнен на Web-сервере
- Внедрение операторов XPath (XPath Injection)эти атаки направлены на Web-серверы, создающие запросы на языке XPath на основе данных, вводимых пользователем

5. РАЗГЛАШЕНИЕ ИНФОРМАЦИИ (INFORMATION DISCLOSURE)

Атаки данного класса направлены на получение дополнительной информации о Web-приложении. Используя эти уязвимости, злоумышленник может определить используемые дистрибутивы ПО, номера версий клиента и сервера и установленные обновления. В других случаях, в утекающей информации может содержаться расположение временных файлов или резервных копий. Во многих случаях эти данные не требуются для работы пользователя. Большинство серверов предоставляют доступ к чрезмерному объему данных, однако необходимо минимизировать объем служебной информации. Чем большими знаниями о приложении будет располагать злоумышленник, тем легче ему будет скомпрометировать систему.

Краткое описание

- Индексирование директорий (Directory Indexing)атаки данного класса позволяют атакующему получить информацию о наличии файлов в Web каталоге, которые недоступны при обычной навигации по Web сайту
- Идентификация приложений (Web Server/Application Fingerprinting) определение версий приложений используется злоумышленником для получения информации об используемых сервером и клиентом операционных системах, Web-северах и браузерах
- Утечка информации (Information Leakage)эти уязвимости возникают в ситуациях, когда сервер публикует важную информацию, например комментарии разработчиков или сообщения об ошибках, которая может быть использована для компрометации системы
- Обратный путь в директориях (Path Traversal)данная техника атак направлена на получение доступа к файлам, директориям и командам, находящимся вне основной директории Web-сервера.
- Предсказуемое расположение ресурсов (Predictable Resource Location) позволяет злоумышленнику получить доступ к скрытым данным или функциональным возможностя

6. ЛОГИЧЕСКИЕ АТАКИ (LOGICAL ATTACKS)

Атаки данного класса направлены на эксплуатацию функций приложения или логики его функционирования. Логика приложения представляет собой ожидаемый процесс функционирования программы при выполнении определенных действий. В качестве примеров можно привести восстановление пролей, регистрацию учетных записей, , аукционные торги, транзакции в системах электронной коммерции. Приложение может требовать от пользователя корректного выполнения нескольких последовательных действий для выполнения определенной задачи. Злоумышленник может обойти или использовать эти механизмы в своих целях.

Краткое описание

- Злоупотребление функциональными возможностями (Abuse of Functionality)данные атаки направлены на использование функций Web-приложения с целью обхода механизмов разграничение доступа
- Отказ в обслуживании (Denial of Service)данный класс атак направлен на нарушение доступности Web-сервера
- Недостаточное противодействие автоматизации (Insufficient Anti-automation)эти уязвимости возникаеют, в случае, если сервер позволяет автоматически выполнять операции, которые должны проводиться вручную
- Недостаточная проверка процесса (Insufficient Process Validation) уязвимости этого класса возникают, когда сервер недостаточно проверяет последовательность выполнения операций приложения

МЕТОД ЧЕРНОГО И БЕЛОГО ЯЩИКА

ЛЕКЦИЯ 2

ЧТО ТАКОЕ «ЧЕРНЫЙ ЯЩИК» СОГЛАСНО ТЕРМИНОЛОГИИ

Black-box тестирование – это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы. *Метод тестирования «черного ящика»* – процедура получения и выбора тестовых случаев на основе анализа спецификации (функциональной или нефункциональной), компонентов или системы без ссылки на их внутреннее устройство.

ГДЕ ИСПОЛЬЗУЕТСЯ МЕТОД «ЧЕРНОГО ЯЩИКА»?

- **1. Интеграционное тестирование.**
  Тестирование, в котором программные и аппаратные компоненты объединяются и тестируются для оценки взаимодействия между ними. При использовании метода «черного ящика» тестировщик проверяет, корректно ли работают все компоненты в целом тогда, когда они интегрированы в большую систему. И действительно, нормальная работа каждой составляющей по отдельности – это еще не гарантия того, что они будут работать вместе в рамках всего проекта. Например, данные могут не отправиться через интерфейс, или интерфейс не отработает согласно документации. При планировании таких тестов тестировщики опираются на спецификацию.
- **2. Функциональное тестирование.**
  Используя этот метод, тестировщик проверяет, выполняет ли программное обеспечение все заявленные функции и требования клиента в полном объеме согласно документации.

- **3. Стресс-тестирование.**
  Предположим, что у нас есть букмекерская онлайн-контора, в документации к которой заявлена возможность одновременной регистрации 1000 пользователей. В этом случае стрессовым тестированием будет непрерывный поток автоматизированных регистраций (как минимум, 1000 регистраций в минуту) на протяжении 12 часов.

- **4. Usability-тестирование.**
  Пусть в упомянутой букмекерской конторе есть функционал «Купон»: мы проверяем, сколько времени уходит у пользователя для добавления ставки в купон, ввода суммы и завершения ставки.

- **5. Тестирование производительности.**
  Таким видом тестирования мы можем проверить: есть ли утечки памяти, насколько быстро система работает и выдает обратную связь, не потребляет ли наше ПО слишком много трафика и не создает ли избыточное количество подключений.

- **6. Приемочное тестирование.**
  После проверки ПО тестировщиками его отдают заказчику, который запускает приемочные тесты «черного ящика» на основе ожиданий от функциональности. Как правило, набор тестов в этом случае определяет сам заказчик, за ним же остается право отказаться от приемки (если его не устроили результаты тестирования).

- **7. Регрессионное тестирование.**
  Проводится на протяжении всего цикла разработки. Цель такого тестирования – проверить работоспособность нового кода и выяснить, не привел ли он к ошибкам или поломкам в старом функционале.

При выборе набора регрессионных тестов следует использовать следующие рекомендации:

- делаем репрезентативную выборку тестов, в которой используются все функции ПО;
- выбираем тесты, сосредоточенные на программных компонентах/функциях, которые подверглись изменениям;
- используем дополнительные тестовые примеры, уделяя основное внимание функциям, на которые с наибольшей вероятностью повлияли изменения.

Хочу обратить ваше внимание на то, что регрессионное тестирование не всегда проводится только методом «черного ящика»; для регресса также используется метод «белого ящика», особенно при поиске функций, на которые с большой вероятностью повлияли изменения.

- **8. Beta-тестирование.**

Это тестирование также проводится методом «черного ящика». Практически готовое ПО отдают для «обкатки» желающим для выявления максимального количества ошибок еще до того, как оно попадет к конечному пользователю.

- Что это дает:
- идентификацию непредвиденных ошибок (так как бета-тестеры используют ПО нестандартно);
- широкий набор окружений для проверки, который трудно обеспечить иными методами (разные операционные системы, разные настройки, разные версии браузеров);
- снижение расходов (так как работа бета-тестеров, как правило, не оплачивается).

ТЕХНИКИ ТЕСТИРОВАНИЯ «ЧЕРНЫМ ЯЩИКОМ»

- **1. Эквивалентное разбиение.**
  Эта техника включает в себя разделение входных значений на допустимые и недопустимые разделы и выбор репрезентативных значений из каждого раздела в качестве тестовых данных. Она может быть использована для уменьшения количества тестовых случаев. Допустим, у нас есть целая переменная N в диапазоне от -99 до 99: позитивными классами эквивалентности будут [-99, -10], [-9, -1], 0, [1, 9], [10, 99], а недействительными (негативными) – <-99, >99, пустое значение, нечисловые строки.

- **2. Анализ граничных значений.**
  Техника, которая включает в себя определение границ входных значений и выбор в качестве тестовых данных значений, находящихся на границах, внутри и вне границ. Многие системы имеют тенденцию вести себя некорректно при граничных значениях, поэтому оценка значений границ приложения очень важна. При проверке мы берем следующие величины: минимум, (минимум-1), максимум, (максимум+1), стандартные значения. Например, в том же случае -99 <= N <= 99 будет использоваться набор: -100, -99, -98, -10, -9 -1, 0, 1, 9, 10, 98, 99, 100.

- **Тестирование таблицы переходов.**
  При данной технике сценарии тестирования выбираются на основе выполнения корректных и некорректных переходов состояний. Допустим, мы хотим записаться на прием к врачу и зарезервировать время своего приема: заходим в форму, выбираем удобное для нас время и нажимаем кнопку «Записаться». Сразу после этого выбранное нами время становится недоступно для другой записи, так как первая запись привела к изменению в базе.
- **4. Тестирование по сценариям использования.**
  Эта техника используется при написании тестов для индивидуального сценария пользователя с целью проверки его работы.

ДОСТОИНСТВА МЕТОДА

- Тестирование методом «черного ящика» позволяет найти ошибки, которые невозможно обнаружить методом «белого ящика». Простейший пример: разработчик забыл добавить какую-то функциональность. С точки зрения кода все работает идеально, но с точки зрения спецификации это – сверхкритичный баг.
- «Черный ящик» позволяет быстро выявить ошибки в функциональных спецификациях (в них описаны не только входные значения, но и то, что мы должны в итоге получить). Если полученный при тестировании результат отличается от заявленного в спецификации, то у нас появляется повод для общения с аналитиком для уточнения конечного результата.

- Тестировщику не нужна дополнительная квалификация. Часто мы пользуемся различными сервисами и приложениями, не очень в них разбираясь. Для того, чтобы открыть инстаграм и обработать свою фотографию, нам совсем не нужно знать способ реализации фильтров. Мы хотим открыть фотографию, выбрать фильтр и получить красивую картинку на выходе. Задача тестировщика, который тестирует эту функцию в инстаграм, – убедиться, что пользователь получит эту самую красивую картинку в соответствии с выбранным фильтром. При этом нам совсем не обязательно иметь какую-либо специализацию – нужны лишь телефон и инстаграм.
- Тестирование проходит «с позиции пользователя». Пользователь всегда прав, он конечный потребитель практически любого ПО, а значит, ему должно быть удобно, комфортно и понятно.
- Составлять тест-кейсы можно сразу после подготовки спецификации. Это значительно сокращает время на тестирование: к тому моменту, как продукт готов к тестированию, тест-кейсы уже разработаны, и тестировщик может сразу приступать к проверке.

НЕДОСТАТКИ МЕТОДА

- Основным недостатком метода «черного ящика» является возможность пропуска границ и переходов, которые не очевидны из спецификации, но есть в реализации кода (собственно, это и заставляет тестировщиков использовать метод «белого ящика»). Вспоминается случай, когда система получала котировки валют с биржи Forex и округляла до 3 знаков после запятой. Система успешно прошла тестирование методом «черного ящика» (так как ни одна валюта не выходила за соответствующие границы) и хорошо работала до тех пор, пока курс доллара к биткоин не вышел за границы 1000 долларов. Тестирование «белым ящиком» выявило бы ошибку: специалист увидел бы, что коэффициент конверсии валюты ограничен 3 знаками.
- Можно протестировать только небольшое количество возможных вводных (входящих) значений; многие варианты остаются без проверки.
- Тесты могут быть избыточными, если разработчик уже проверил данную функциональность (например, Unit-тестом).
- При отсутствии четкой и полной спецификации проектировать тесты и тест-сценарии оказывается затруднительно.

ПОДВЕДЕМ ИТОГИ

Из представленной информации мы можем сделать следующий вывод: метод «черного ящика» является эффективным при различных видах тестирования; но следует помнить, что некоторые ошибки невозможно найти, используя только этот метод (например, ошибки во внутренней структуре кода).

Проведение «black-box» тестирования увеличивает уверенность в том, что приложение надежно работает на широком диапазоне входных данных, так как набор тестовых данных зависит только от спецификации, а не от особенностей внутренней реализации продукта (как в случае применения методов «белого» и «серого» ящиков).

Метод «черного ящика» выгодно применять, если вы ищете:

- неправильно реализованные функции приложения или сервиса;
- ошибки в пользовательском интерфейсе;
- ошибки в функциональных спецификациях.

Для реализации наиболее полной проверки я рекомендую использовать методы «черного» и «белого» ящиков одновременно. Это позволит увеличить покрытие возможных сценариев, снизить риск пропуска ошибки, а также качественно улучшить результаты тестирования, так как приложение или сервис будет проверено двумя разными методами – с позиций пользователя и внутреннего устройства системы.

ЧТО ТАКОЕ ТЕСТИРОВАНИЕ БЕЛОГО ЯЩИКА?

WHITE BOX TESTING — это тестирование внутренней структуры, дизайна и кодирования программного решения. В этом типе тестирования код виден тестеру. Основное внимание уделяется проверке потока входных и выходных данных через приложение, улучшению дизайна и удобства использования, усилению безопасности. Тестирование белого ящика также известно как тестирование Clear Box, тестирование Open Box, структурное тестирование, тестирование прозрачного бокса, тестирование на основе кода и тестирование Glass Box. Это обычно выполняется разработчиками

- Это одна из двух частей подхода **Box Testing** к тестированию программного обеспечения. Его аналог, **тестирование Blackbox** , включает тестирование с точки зрения внешнего или конечного пользователя. С другой стороны, тестирование Whitebox основано на внутренней работе приложения и вращается вокруг внутреннего тестирования.
- Термин «WhiteBox» был использован из-за концепции прозрачной коробки. Ясное поле или имя WhiteBox символизирует способность видеть сквозь внешнюю оболочку программного обеспечения (или «коробку») в его внутренней работе. Аналогично, «черный ящик» в « Тестировании черного ящика » символизирует невозможность увидеть внутреннюю работу программного обеспечения, так что может быть протестирован только опыт конечного пользователя

ЧТО ВЫ ПРОВЕРЯЕТЕ В WHITE BOX TESTING?

Тестирование белого ящика включает тестирование программного кода для следующего:

- Внутренние дыры в безопасности
- Сломанные или плохо структурированные пути в процессах кодирования
- Поток конкретных входов через код
- Ожидаемый результат
- Функциональность условных циклов
- Тестирование каждого оператора, объекта и функции на индивидуальной основе

Тестирование может проводиться на системном, интеграционном и модульном уровнях разработки программного обеспечения. Одной из основных целей тестирования whitebox является проверка рабочего процесса для приложения. Он включает в себя тестирование ряда предопределенных входных данных по отношению к ожидаемым или желаемым выходным данным, так что когда конкретный ввод не приводит к ожидаемому выходному сигналу, вы столкнулись с ошибкой.

КАК ВЫ ПРОВОДИТЕ ТЕСТИРОВАНИЕ БЕЛОГО ЯЩИКА?

Чтобы дать вам упрощенное объяснение тестирования белого ящика, мы разделили его на **два основных этапа** . Вот что делают тестеры при тестировании приложения с использованием техники тестирования белого ящика:

ШАГ 1) ПОНИМАТЬ ИСТОЧНИК КОД

Первое, что часто делает тестер, — это изучает и понимает исходный код приложения. Поскольку тестирование белого ящика включает в себя тестирование внутренней работы приложения, тестировщик должен быть очень хорошо осведомлен в языках программирования, используемых в тестируемых приложениях. Кроме того, тестирующий должен быть хорошо осведомлен о методах безопасного кодирования. Безопасность часто является одной из основных задач тестирования программного обеспечения. Тестер должен уметь обнаруживать проблемы с безопасностью и предотвращать атаки хакеров и наивных пользователей, которые могут вводить вредоносный код в приложение как сознательно, так и неосознанно.

Шаг 2) СОЗДАТЬ ИСПЫТАТЕЛЬНЫЕ ДЕЛА И ИСПОЛНИТЬ

Второй базовый шаг к тестированию белого ящика включает в себя тестирование исходного кода приложения на предмет правильной работы и структуры. Одним из способов является написание большего количества кода для проверки исходного кода приложения. Тестировщик разработает небольшие тесты для каждого процесса или серии процессов в приложении. Этот метод требует, чтобы тестировщик имел глубокие знания кода и часто выполнялся разработчиком. Другие методы включают ручное тестирование , тестирование и тестирование на ошибки, а также использование инструментов тестирования, как мы объясним далее в этой статье.

МЕТОДЫ ИСПЫТАНИЙ БЕЛОЙ КОРОБКИ

Основным методом тестирования Белого ящика является анализ покрытия кода. Анализ покрытия кода устраняет пробелы в наборе тестовых примеров. Он определяет области программы, которые не выполняются набором тестовых случаев. После выявления пробелов вы создаете контрольные примеры для проверки непроверенных частей кода, тем самым повышая качество программного продукта.

Доступны автоматизированные инструменты для анализа покрытия кода. Ниже приведены несколько методов анализа покрытия

- **Охват** операторов: — Этот метод требует, чтобы каждое возможное утверждение в коде было проверено хотя бы один раз в процессе тестирования разработки программного обеспечения.
- **Покрытие ветвления —** этот метод проверяет все возможные пути (если-еще и другие условные циклы) программного приложения.

Помимо вышесказанного, существует множество типов покрытия, таких как покрытие условий, покрытие нескольких условий, покрытие пути, покрытие функций и т. Д. Каждый метод имеет свои преимущества и пытается протестировать (охватить) все части программного кода. Используя покрытие Statement и Branch, вы обычно достигаете 80-90% покрытия кода, что является достаточным.

ТИПЫ ТЕСТИРОВАНИЯ БЕЛОГО ЯЩИКА

Тестирование белого ящика включает в себя несколько типов тестирования, используемых для оценки удобства использования приложения, блока кода или конкретного программного пакета.

- **Модульное тестирование:** это часто первый тип тестирования, выполняемый в приложении. Модульное тестирование выполняется на каждом модуле или блоке кода по мере его разработки. Модульное тестирование по сути делается программистом. Как разработчик программного обеспечения, вы разрабатываете несколько строк кода, одну функцию или объект и тестируете их, чтобы убедиться, что они работают, прежде чем продолжить модульное тестирование, которое помогает выявить большинство ошибок на ранних этапах жизненного цикла разработки программного обеспечения. Ошибки, выявленные на этом этапе, дешевле и их легко исправить.

- **Тестирование на утечки памяти** : утечки памяти являются основными причинами медленной работы приложений. Специалист по обеспечению качества, имеющий опыт в обнаружении утечек памяти, необходим в тех случаях, когда у вас медленно работает программное приложение.

Помимо вышесказанного, несколько типов тестирования являются частью тестирования как черного ящика, так и белого ящика. Они перечислены ниже

- **<u>Тестирование проникновения в</u> White Box :** В этом тестировании тестировщик / разработчик имеет полную информацию об исходном коде приложения, подробную информацию о сети, задействованные IP-адреса и всю информацию о сервере, на котором работает приложение. Цель состоит в том, чтобы атаковать код с нескольких сторон, чтобы выявить угрозы безопасности.
- **Тестирование** мутации **White Box** : **Тестирование** мутации часто используется, чтобы обнаружить лучшие методы кодирования, используемые для расширения программного решения.

ПРЕИМУЩЕСТВА ТЕСТИРОВАНИЯ БЕЛОГО ЯЩИКА

- Оптимизация кода путем поиска скрытых ошибок.
- Тестовые случаи «белого ящика» могут быть легко автоматизированы.
- Тестирование является более тщательным, поскольку обычно покрываются все пути кода.
- Тестирование может начаться рано в SDLC, даже если GUI недоступен.

НЕДОСТАТКИ ТЕСТИРОВАНИЯ WHITEBOX

- Тестирование белого ящика может быть довольно сложным и дорогостоящим.
- Разработчики, которые обычно выполняют тесты «белого ящика», ненавидят это. Тестирование белого ящика разработчиками не детально может привести к производственным ошибкам.
- Тестирование белого ящика требует профессиональных ресурсов с подробным пониманием программирования и реализации.
- Тестирование белого ящика отнимает много времени, более крупные приложения для программирования требуют времени для полного тестирования.

КОНЕЧНЫЕ ЗАМЕТКИ:

- Тестирование белого ящика может быть довольно сложным. Сложность связана с тестируемым приложением. Небольшое приложение, которое выполняет одну простую операцию, может быть протестировано в течение нескольких минут, в то время как более крупным программным приложениям требуются дни, недели и даже больше для полного тестирования.
- Тестирование белого ящика должно проводиться на программном приложении в том виде, в каком оно разрабатывается после его написания, и снова после каждой модификации.

АВТОМАТИЧЕСКОЕ СКАНИРОВАНИЕ.

ЛЕКЦИЯ 3.1

Сканеры веб-приложений —это автоматизированные программы, которые проводят общесистемное **сканирование** программного обеспечения и **веб-приложений** для поиска уязвимостей, которые они могут содержать. **Эти сканеры** просматривают весь веб-сайт, помещают файлы, которые они находят в результате глубокого анализа, и визуализируют структуру веб-сайта в целом. Эти сканеры также известны тем, что имитируют атаки на приложения, чтобы найти и оценить серьезность обнаруженной уязвимости.

- Сканер уязвимостей веб-приложений, также известный как сканер безопасности веб-приложений, представляет собой автоматизированный инструмент безопасности. Он сканирует веб-приложения на наличие вредоносных программ, уязвимостей и логических недостатков. Сканер использует тесты «черного ящика», так как эти тесты не требуют доступа к исходному коду, а вместо этого запускают внешние атаки для проверки уязвимостей безопасности. Эти симулированные атаки могут обнаруживать обход пути, межсайтовый скриптинг (XSS) и внедрение команд.

- Сканеры веб-приложений относятся к категории инструментов динамического тестирования безопасности приложений (DAST). Инструменты DAST дают представление о том, как ведут себя ваши веб-приложения во время их работы, позволяя вашему бизнесу устранить потенциальные уязвимости, прежде чем хакер использует их для проведения атаки. По мере развития ваших веб-приложений решения DAST продолжают сканировать их, чтобы ваш бизнес мог быстро выявлять и устранять возникающие проблемы, прежде чем они перерастут в серьезные риски.

- Сканер уязвимостей веб-приложения сначала сканирует весь веб-сайт, анализируя каждый найденный файл и отображая всю структуру веб-сайта. После этого этапа обнаружения он выполняет автоматический аудит распространенных уязвимостей безопасности, запуская серию веб-атак. Сканеры веб-приложений проверяют наличие уязвимостей на веб-сервере, прокси-сервере, сервере веб-приложений и даже на других веб-службах. В отличие от сканеров исходного кода, сканеры веб-приложений не имеют доступа к исходному коду и поэтому обнаруживают уязвимости, фактически выполняя атаки.

- Оценка уязвимости веб-приложения сильно отличается от общей оценки уязвимости, когда безопасность фокусируется на сетях и хостах. Сканер уязвимостей приложения сканирует порты, подключается к службам и использует другие методы для сбора информации, раскрывающей уровни исправлений, конфигурации и потенциальные уязвимости нашей инфраструктуры.
- Инструменты автоматического сканирования веб-приложений помогают пользователю убедиться в том, что весь веб-сайт правильно сканируется, и что никакие входные данные или параметры не остаются не проверенными. Автоматические сканеры веб-уязвимостей также помогают находить высокий процент технических уязвимостей и дают вам очень хороший обзор структуры веб-сайта и состояния безопасности.

- Лучший способ определить угрозы безопасности веб-приложений - это выполнить оценку уязвимости веб-приложений. Важность этих угроз может сделать вашу организацию уязвимой, если они не будут должным образом идентифицированы и смягчены. Поэтому внедрение решения для сканирования безопасности веб-приложений должно иметь первостепенное значение для планов безопасности вашей организации в будущем.

ИСТОЧНИКИ УГРОЗ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ И МЕРЫ ПО ИХ ПРЕДОТВРАЩЕНИЮ.

ЛЕКЦИЯ 3.2

РАЗНОВИДНОСТИ УГРОЗ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Угрозой информации называют потенциально возможное влияние или воздействие на автоматизированную систему с последующим нанесением убытка чьим-то потребностям.

На сегодня существует более 100 позиций и разновидностей угроз информационной системе. Важно проанализировать все риски с помощью разных методик диагностики. На основе проанализированных показателей с их детализацией можно грамотно выстроить систему защиты от угроз в информационном пространстве.

КЛАССИФИКАЦИЯ УЯЗВИМОСТЕЙ СИСТЕМ БЕЗОПАСНОСТИ

Угрозы информационной безопасности проявляются не самостоятельно, а через возможное взаимодействие с наиболее слабыми звеньями системы защиты, то есть через факторы уязвимости. Угроза приводит к нарушению деятельности систем на конкретном объекте-носителе.

Основные уязвимости возникают по причине действия следующих факторов:

- несовершенство программного обеспечения, аппаратной платформы;
- разные характеристики строения автоматизированных систем в информационном потоке;
- часть процессов функционирования систем является неполноценной;
- неточность протоколов обмена информацией и интерфейса;
- сложные условия эксплуатации и расположения информации.

Чаще всего источники угрозы запускаются с целью получения незаконной выгоды вследствие нанесения ущерба информации. Но возможно и случайное действие угроз из-за недостаточной степени защиты и массового действия угрожающего фактора.

- Существует разделение уязвимостей по классам, они могут быть:
- объективными;
- случайными;
- субъективными.

Если устранить или как минимум ослабить влияние уязвимостей, можно избежать полноценной угрозы, направленной на систему хранения информации.

КЛАССЫ УЯЗВИМОСТЕЙ

ОБЪЕКТИВНЫЕ

СЛУЧАЙНЫЕ

СУБЪЕКТИВНЫЕ

ОБЪЕКТИВНЫЕ УЯЗВИМОСТИ

Этот вид напрямую зависит от технического построения оборудования на объекте, требующем защиты, и его характеристик. Полноценное избавление от этих факторов невозможно, но их частичное устранение достигается с помощью инженерно-технических приемов, следующими способами:

- **Связанные с техническими средствами излучения:**
- электромагнитные методики (побочные варианты излучения и сигналов от кабельных линий, элементов техсредств);
- звуковые варианты (акустические или с добавлением вибросигналов);
- электрические (проскальзывание сигналов в цепочки электрической сети, по наводкам на линии и проводники, по неравномерному распределению тока)

- **Активизируемые:**
- вредоносные ПО, нелегальные программы, технологические выходы из программ, что объединяется термином «программные закладки»;
- закладки аппаратуры – факторы, которые внедряются напрямую в телефонные линии, в электрические сети или просто в помещения.

- **Те, что создаются особенностями объекта, находящегося под защитой:**
- расположение объекта (видимость и отсутствие контролируемой зоны вокруг объекта информации, наличие вибро- или звукоотражающих элементов вокруг объекта, наличие удаленных элементов объекта);
- организация каналов обмена информацией (применение радиоканалов, аренда частот или использование всеобщих сетей).

- **Те, что зависят от особенностей элементов-носителей:**
- детали, обладающие электроакустическими модификациями (трансформаторы, телефонные устройства, микрофоны и громкоговорители, катушки индуктивности);
- вещи, подпадающие под влияние электромагнитного поля (носители, микросхемы и другие элементы).

СЛУЧАЙНЫЕ УЯЗВИМОСТИ

- Эти факторы зависят от непредвиденных обстоятельств и особенностей окружения информационной среды. Их практически невозможно предугадать в информационном пространстве, но важно быть готовым к их быстрому устранению. Устранить такие неполадки можно с помощью проведения инженерно-технического разбирательства и ответного удара, нанесенного угрозе информационной безопасности:

Сбои и отказы работы систем:

- вследствие неисправности технических средств на разных уровнях обработки и хранения информации (в том числе и тех, что отвечают за работоспособность системы и за контроль доступа к ней);
- неисправности и устаревания отдельных элементов (размагничивание носителей данных, таких как дискеты, кабели, соединительные линии и микросхемы);
- сбои разного программного обеспечения, которое поддерживает все звенья в цепи хранения и обработки информации (антивирусы, прикладные и сервисные программы);
- перебои в работе вспомогательного оборудования информационных систем (неполадки на уровне электропередачи).

Ослабляющие информационную безопасность факторы:

- повреждение коммуникаций вроде водоснабжения или электроснабжения, а также вентиляции, канализации;
- неисправности в работе ограждающих устройств (заборы, перекрытия в здании, корпуса оборудования, где хранится информация).

СУБЪЕКТИВНЫЕ УЯЗВИМОСТИ

Этот подвид в большинстве случаев представляет собой результат неправильных действий сотрудников на уровне разработки систем хранения и защиты информации. Поэтому устранение таких факторов возможно при помощи методик с использованием аппаратуры и ПО:

Неточности и грубые ошибки, нарушающие информационную безопасность:

- на этапе загрузки готового программного обеспечения или предварительной разработки алгоритмов, а также в момент его использования (возможно во время ежедневной эксплуатации, во время ввода данных);
- на этапе управления программами и информационными системами (сложности в процессе обучения работе с системой, настройки сервисов в индивидуальном порядке, во время манипуляций с потоками информации);
- во время пользования технической аппаратурой (на этапе включения или выключения, эксплуатации устройств для передачи или получения информации).

Нарушения работы систем в информационном пространстве:

- режима защиты личных данных (проблему создают уволенные работники или действующие сотрудники в нерабочее время, они получают несанкционированный доступ к системе);
- режима сохранности и защищенности (во время получения доступа на объект или к техническим устройствам);
- во время работы с техустройствами (возможны нарушения в энергосбережении или обеспечении техники);
- во время работы с данными (преобразование информации, ее сохранение, поиск и уничтожение данных, устранение брака и неточностей).

РАНЖИРОВАНИЕ УЯЗВИМОСТЕЙ

Каждая уязвимость должна быть учтена и оценена специалистами. Поэтому важно определить критерии оценки опасности возникновения угрозы и вероятности поломки или обхода защиты информации. Показатели подсчитываются с помощью применения ранжирования. Среди всех критериев выделяют три основных:

- **Доступность** – это критерий, который учитывает, насколько удобно источнику угроз использовать определенный вид уязвимости, чтобы нарушить информационную безопасность. В показатель входят технические данные носителя информации (вроде габаритов аппаратуры, ее сложности и стоимости, а также возможности использования для взлома информационных систем неспециализированных систем и устройств).
- **Фатальность** – характеристика, которая оценивает глубину влияния уязвимости на возможности программистов справиться с последствиями созданной угрозы для информационных систем. Если оценивать только объективные уязвимости, то определяется их информативность – способность передать в другое место полезный сигнал с конфиденциальными данными без его деформации.
- **Количество** – характеристика подсчета деталей системы хранения и реализации информации, которым присущ любой вид уязвимости в системе.

https://searchinform.ru/informatsionnaya-bezopasnost/osnovy-ib/ugrozy-informatsionnoj-bezopasnosti/

РЕГЛАМЕНТЫ И МЕТОДЫ РАЗРАБОТКИ БЕЗОПАСНЫХ ВЕБ-ПРИЛОЖЕНИЙ. ОЦЕНКА РИСКА

ЛЕКЦИЯ 4