Сценарии тестирования API

Сценарий тестирования микросервисов/сервисов только с API - REST и gRPC

Список инструментов:

- Инструменты динамического анализа приложений (тестирования API)
  - Burp Suite Professional
  - grpcui
  - PostMan
  - Zed Attack

Номер требования: 004 - Сервис корректно работает с JWT токенами

Тестовый сценарий:

1. Запустить Burp suite и пройтись по методам размещенным на API Gateway
2. Выбрать метод проверяющий права доступа и экспортировать в Repeater.
3. JWT токен закинуть в расширение JOSEPH для формирования JWT-токенов для тестирования сервисов:
   a. *JWT Verification Bypass* - поделка заголовка и подписи для обхода проверки
   b. *Key Confusion Attack* - подмена ключа на другой публичный ключ
   c. *None Algorithm Exploitation* - замена алгортма на none/None и исключение подписи
4. Закинуть JWT в утилиту `jwt_tool` для проверки токена и формирования токена для тестирования следующих атак:
   a. *HMAC Key Cracking* - попытаться подобрать ключ используемого для создания подписи ключа симметричным алгоритмом

      ```
      python3 jwt_tool.py -C -d ~/Downloads/10k-most-common.txt [JWT]
      ```

   b. *Key Confusion Attack* - подмена ключа на другой публичный ключ

      ```
      python3 jwt_tool.py  -X k -pk pubkey.cer [JWT]
      ```

   c. Spoofing JWKS - замена ключей на публично доступный JWKS.

      ```
      python3 jwt_tool.py  -X s -ju http://7.212.31.1/test.jwks [JWT]
      ```

5. Полученные ключи отправить в сервис получить ответы
6. Проанализировать ответы на корректность

Вводные данные: подготовленные JWT-токены

Выходные данные: Сервер отвечает ошибками или кодами из-за отсутствия доступов.

Номер требования: 005 - Сервис корректно работает с аутентификационными данными, не дает доступы к методам с ограниченным доступом.

Тестовый сценарий:

1. Запустить Burp Suite, собрать все нужные методы
2. Попробовать получить аутентификационные данные с помощью перебора
3. Протестировать создание случайных или сгенерированных ключей доступа для получения доступа
4. Протестировать все методы на проверку аутентификационных данных

Вводные данные: список методов, модель разграничения доступов для конкретного сервиса

Выходные данные: Сервер отвечает ошибками или кодами из-за отсутствия доступов.

Номер требования: 006 - Сервис корректно работает с авторизацией, корректно ограничивает доступы

Тестовый сценарий:

1. Создать несколько учетных записей с разными доступами
2. Открыть разные методы с разными токенами для проверки корректной работы авторизации
3. Провалидировать крайние варианты - от минимальных доступов до максимальных.
4. Протестировать сценарий когда пользователь А создал объект, и к нему имеет доступ только пользователь А, а пользователь В не имеет доступа.

Вводные данные: список методов, модель разграничения доступов для конкретного сервиса, разные учетные записи с выпущенными аутентификационными данными

Выходные данные: Сервер отвечает ошибками или кодами из-за отсутствия доступов, на ограниченных аутентификационных данных.

Номер требования: 007 - Сервис имеет лимиты на методах.

Тестовый сценарий:

1. Открыть values/config.yaml и просмотреть есть ли ограничения на методах API
2. С помощью Intruder/TurboIntruder протестировать работают ли лимиты
3. Проанализировать лимиты на корректность с точки зрения бизнес-процесса и функциональности метода

Вводные данные: список методов, настройки мтеодов в config.yaml

Выходные данные: Сервер отвечает ошибками 429 или 200-Error при достижении рейтлимита.

Номер требования: 008 - Сервис корректно реализует ограничение доступа на уровне функций (Broken Function Level Authorization - BFLA)

Тестовый сценарий:

1. Создать ресурс/объект: выполнить действие от имени пользователя А.
2. Попытка несанкционированного действия: выполнить то же действие от имени пользователя Б, у которого нет необходимых привилегий.
3. Подтвердить несанкционированный доступ: если пользователь Б может выполнить действие, это указывает на уязвимость BFLA.

Вводные данные: список методов, настройки мтеодов в config.yaml

Выходные данные: Сервер отвечает ошибками 403 или 200-Error при попытке выполнить функцию.

Номер требования: 009 - Сервис исключает утечку чувствительных данных

Тестовый сценарий:

1. Проверить ответы API на предмет чрезмерного раскрытия информации, например конфиденциальных данных, представленных в виде обычного текста.
2. Протестировать пагинацию и схемы доступа к спискам данных
3. Протестировать вызов ошибки сервиса для получения технической информации (трейс информацию)

Вводные данные: список методов, пейлоды для вызова ошибки

Выходные данные: Сервер отвечает в рамках спецификации, не отдает лишнюю информацию

Номер требования: 010 - Сервис исключает инъекции

Тестовый сценарий:

1. Проверить методы на инъекции с помощью Intruder и вшитых вариантов инъекций (XSS, SQL, Command и др.) в следующих входных точках
    a. Строки запроса: управление параметрами в строке запроса URL.
    b. Токены: использование уязвимостей в механизмах аутентификации на основе токенов.
    c. Параметры в запросах: внедрение полезных данных в параметры запросов POST или PUT.
    d. Заголовки: манипулирование заголовками для выполнения вредоносных действий. (Referrer, Location, Authorization)
2. Методы обнаружения:
    a. Сообщения об ошибках: проанализировать сообщения об ошибках на предмет признаков успешных попыток инъекции.
    b. Поведенческий анализ:протестировать изменения в поведении системы или реакции на введенную полезную нагрузку.
    c. Ручное тестирование: провести ручное тестирование с использованием специально созданных полезных данных для выявления уязвимых конечных точек.
3. Пейлоды для инъекций можно взять в репозитории - https://github.com/danielmiessler/SecLists/

Вводные данные: список методов, пейлоды для вызова ошибки

Выходные данные: Сервер отвечает в рамках спецификации, не отдает лишнюю информацию

Номер требования: 011 - Сервис исключает злоупотребление API

Тестовый сценарий:

1. Проверить методы на возможность массово выгружать данные
    a. Проверить параметры `pageSize, count, limit` на возможность применения больших чисел
    b. Проверить методы экспорта на возможность применения без фильтров и получения всех данных разом
2. Протестировать наличие доступных неопубликованных методов, которые должны быть доступны только из инфрасети
3. Проверить объем данных выдаваемый API-методами

Вводные данные: список методов, пейлоды для вызова ошибки

Выходные данные: Сервер отвечает в рамках спецификации, не отдает лишнюю информацию

Номер требования: 012 - Сервис корректно реализует бизнес логику

Тестовый сценарий:

1. Пройти по бизнес-процессу, протестировать корректную логику
2. Вызвать методы и функции в обход логики, провалидировать корректность результата

Вводные данные: список методов, описание бизнес-процесса

Выходные данные: Сервер отвечает в рамках спецификации, не разрешает вызвать метод вне правильного процесса

Номер требования: 013 - Сервис корректно работает с объектами и сущностями в рамках процесса

Тестовый сценарий:

1. Если API реализует работу с объектами или сущностями, необходимо проверить работу с параметрами-идентификаторами
2. Если идентификатор числовой (int), проверить доступность соседних объектов (перебор параметра с помощью Intruder /fuzzer)
3. Если идентификатор строковый, необходимо проверить потенциальную возможность перебора других объектов путем замены символов или случайного перебора.
   Пример: тестирование возможности перебора промокода
4. Если идентификаторы представляют собой UUID/GUID, смотрим возможность переиспользования идентификаторов после удаления и перемещения объекта.

Вводные данные: список методов, описание бизнес-процесса

Выходные данные: Сервер отвечает в рамках спецификации, не разрешает обратиться к объектам в обход прав доступа