Sniffers

Снифферы - это инструменты, позволяющие перехватывать, анализировать и модернизировать все запросы, которые через них проходят. Они полезны, когда из потока нужно извлечь какие-либо сведения или создать нужный ответ сервера. Так можно проводить модульное тестирование продукта, в котором есть и бэк, и фронт, и разные команды со своей версионностью.

Принцип работы, или каким образом сниффер трафика справляется со своей задачей видеть всё

Благодаря тому что снифферы являются инструментами, которые позволяют перехватывать, анализировать и модернизировать все проходящие через них запросы, их удобно использовать в ситуациях, когда из потока нужно извлечь сведения или создать нужный ответ сервера.

Это работает так: браузер отправляет запрос, сниффер его «проксирует» и отправляет от своего лица пользователю. Далее приходит ответ от сервера – он тоже поступает сначала в сниффер, а затем – к нам.

Когда нам надо просмотреть запросы, например из localhost – проблем не возникает, так как там используется протокол HTTP в чистом виде. При использовании протокола HTTPS и SSL-сертификата данные может читать только отправитель и получатель. Поэтому, для того чтобы просмотреть данные по определённому URL, нужно включить в Charles опцию SSL-proxying – и тогда данные станут видны и читаемы

Как использовать сниффер в QA

Charles vs Postman

Представим, что нам нужно быстро и без лишних телодвижений сделать несколько действий:

- достать токен авторизации с мобильного устройства,
- получить тело ответа при оплате на мобильном устройстве,
- выбрать запрос, поменять 1 символ в jwt токене и проверить, будет ли ошибка 401 (или вообще без него отправить запрос),
- выбрать запрос, поменять ему метод POST на PUT и проверить, будет ли ошибка 405.

Если делать это в Postman, то нужно ввести URL, выполнить запрос на авторизацию (конечно, с вводом своих данных) и получить токен. Потом в целевом запросе нужно было бы правильно ввести параметры и токен, и лишь после всей этой цепочки действий поменять Post на Put.

Как сделать проще? Сделать запрос с frontend мобильного устройства в Charles: контекстное меню -> клик на Compose, в теле запросе изменить Post на Put и нажать Execute.

Break points

Представим, что нам нужно протестировать на клиенте вёрстку и проверить, как будет отображаться большое количество бонусов у пользователя. В такой ситуации многие предложили бы изменить в БД количество бонусов. Но у сервера может быть кэш и придётся ждать, либо время может занять подключение к БД.

Есть вариант проще! Изменить ответ сервера. Для этого нужно включить функцию Break points, и у нас появится возможность на любом запросе редактировать и тело самого запроса, и тело ответа.

Перед выполнением запроса Charles его остановит, и нам нужно будет только поменять текст. Почти что аналогия работы с точками остановки в программировании :)

Также Break points можно использовать в другой фиче. Представим абстрактный список пользователей, у которых есть рейтинг. Если этот рейтинг ниже 3.0, то фронт должен показывать смайлик «Палец вниз». Рейтинг может варьироваться от 0.0 до 5.0.

Чтобы не создавать много пользователей с разными рейтингами, можно использовать Break points и подменять параметр «Рейтинг» в ответе. Это упростит работу.

REWRITE – автоматическая подмена запросов и ответов

Rewrite – это инструмент, позволяющий создавать правила, которые изменяют запросы и ответы, когда те проходят через Charles Proxy. Например, можно добавлять и изменять заголовок, искать и заменять текст в теле ответа или запроса и т.д. Настройка выглядит не такой уж и сложной: указывается Location = путь запроса. Далее создаётся правило, которое состоит из 5 разделов:

Type: Body (потому что параметр находится в теле);

Where: Response (потому что параметр находится в ответе от сервера);

Раздел Match: в «Value» указываем значение и параметр, который возвращает сервер; Раздел Replace: в «Value» указываем значение и параметр, который мы хотим увидеть на клиенте.

Throttling – ограничение пропускной способности сети

Полезно, когда нужно протестировать приложение на медленной скорости интернета (Например, 2G). Throttling позволяет также настроить стабильность интернета и даже Пинг! Ограничение можно включить для всего трафика, либо для конкретного хоста.

No-caching

Инструмент No Caching предотвращает кэширование, манипулируя заголовками HTTP. Заголовки If-Modified-Since и If-None-Match удаляются из запросов, добавляются Pragma: no-cache и Cache-control: no-cache. Заголовки Expires, Last-Modified и ETag удаляются из ответов и добавляются Expires: 0 и Cache-Control: no-cache.

Это полезно, если нужно изменить ответ сервера, а браузер берёт данные из кэша. Таким образом при каждом запросе будут использоваться данные от сервера.

Block cookies

По аналогии с No caching эта функция удаляет заголовок Cookie из запросов. Также из ответа сервера она удаляет заголовок Set cookie. После включения этой опции сервер не сможет собирать Cookie с запросов. Также можно настраивать это для всех хостов или для какого-то конкретного.

Итог

Мы рассмотрели основные функции снифферов трафика (в аналогичных программах fiddler, wireshark можно сделать все те же действия). Разобрали примеры использовании в тестировании и то, какие проблемы они могут помочь решить.

Ресурс: https://vc.ru/life/467157-sniffer-trafika-i-dlya-chego-on-nuzhen-testirovshchiku