Fiddler = удобный сниффер + прокси сервер

Fiddler это: кроссплатформенное приложение прокси-сервера для отладки HTTP. Он позволяет пользователю просматривать HTTP, HTTPS и активированный трафик TCP-порта, доступ к которому осуществляется с локального компьютера, на него или через него. Сюда входят запросы и ответы, включая HTTP-заголовки и метаданные (например, файлы cookie, кэширование и кодирование информации), с функциями, предназначенными для помощи разработчикам и тестировщикам в анализе соединений и обмене сообщениями.

Зачем это делать?

Пример 1. **Анализ трафика.**
Пользователи вашей сети пользуются вашим прокси-сервером. Вы можете увидеть на какие сайты заходят пользователи, запрещать дальнейшие переходы на эти сайты.

Пример 2. **Сбор данных.**
Ваши пользователи пользуются через вас некоторыми веб-ресурсами. Например, они вводят vin-номер своего автомобиля на сайте дилера авто и получают в ответ данные этого автомобиля. Вы можете сохранять эти данные в свою базу данных.

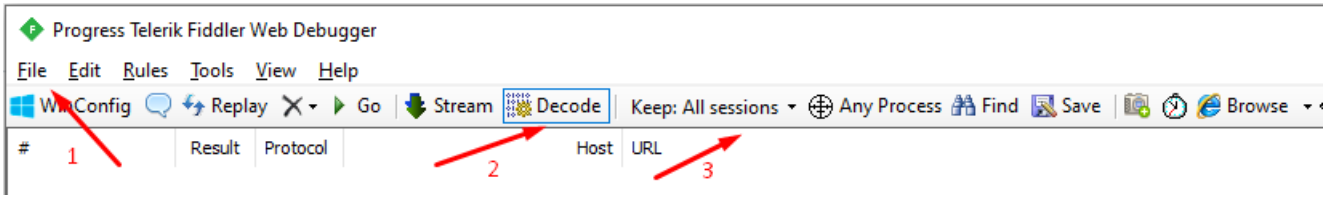
Пример 3. **Подмена HTTP-пакетов.**
Вам нужно изменить для ваших пользователей внешний вид сайта. Вы можете изменить стили сайта, скрывать любые элементы, добавить свои элементы, вырезать определенные слова или заменить их на другие слова, изменить картинку сайта на любую свою.

Пример 4. **Подмена POST-данных.**
Вам нужно подправить данные передаваемые на веб-сервер через POST-запрос. Существует множество информации передаваемой в POST-запросах. Пример: отправка логина/пароля на сервер в процессе авторизации. Или онлайн тест отправляет на сервер результаты вашего теста.

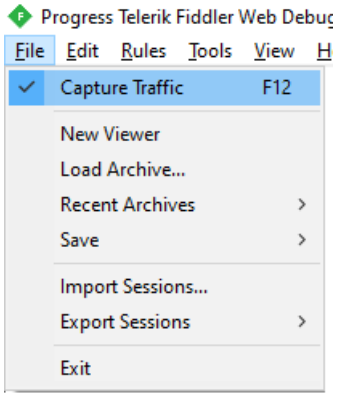
Установка Fiddler

1. Переходим на https://www.telerik.com/download/fiddler, скачиваем Fiddler Classic.
2. Установка простая и быстрая.
3. Запускаем программу.

Настройка Fiddler



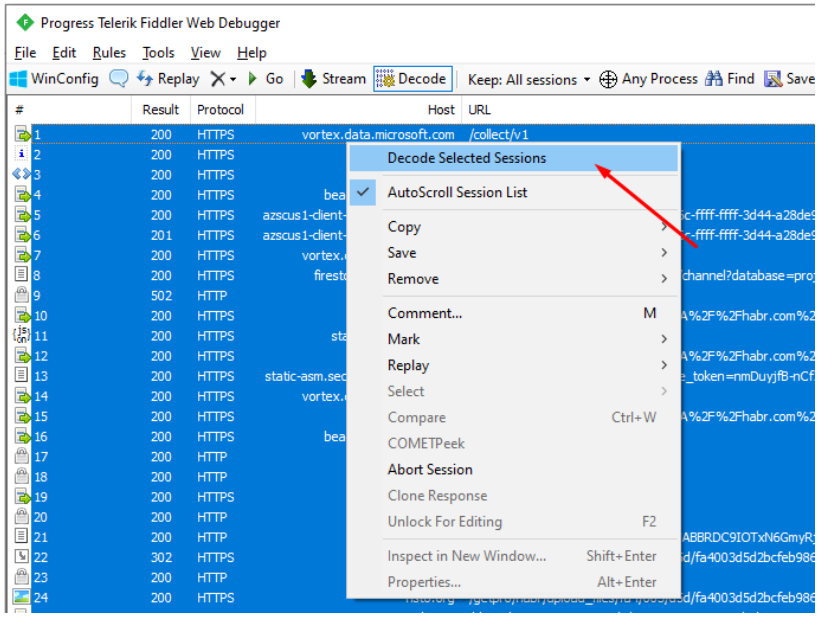
В меню File есть опция "*Capture Traffic*". По умолчанию опция включена. Это означает что Fiddler прописывает в реестре Windows себя в качестве прокси-сервера. Браузеры Internet Explorer, Edge, Chrome используют данную настройку, а значит HTTP-пакеты от этих браузеров пойдут через Fiddler.



Если опция "*File -> Capture Traffic*" выключена, то Fiddler перестает работать как системный прокси-сервер и перехватывает только те пакеты, которые идут непосредственно на адрес Fiddler. Это может быть когда вы настроили ваше приложение или браузер сами для работы через ip/port Fiddler. По умолчанию Fiddler слушает на порту 127.0.0.1:8888

Опция "*Keep: All sessions*".
В данном режиме Fiddler не очищает журнал собранных HTTP-пакетов. Если требуется продолжительная работа Fiddler, то при большой нагрузке этих пакетов будет очень много и Fiddler скушает всю доступную оперативную память компьютера. Чтобы этого не случилось переключите в режим "Keep: 100 sessions".

Опция "*Decode*".
По умолчанию выключена. В процессе анализа собранных пакетов рекомендуется включить чтобы пакеты автоматически декодировались. Либо можно выделить собранные пакеты через Ctrl+A, вызвать меню нажатием правой кнопки мыши по выделенным пакетам и нажать "Decode Selected Sessions".



Основные настройки

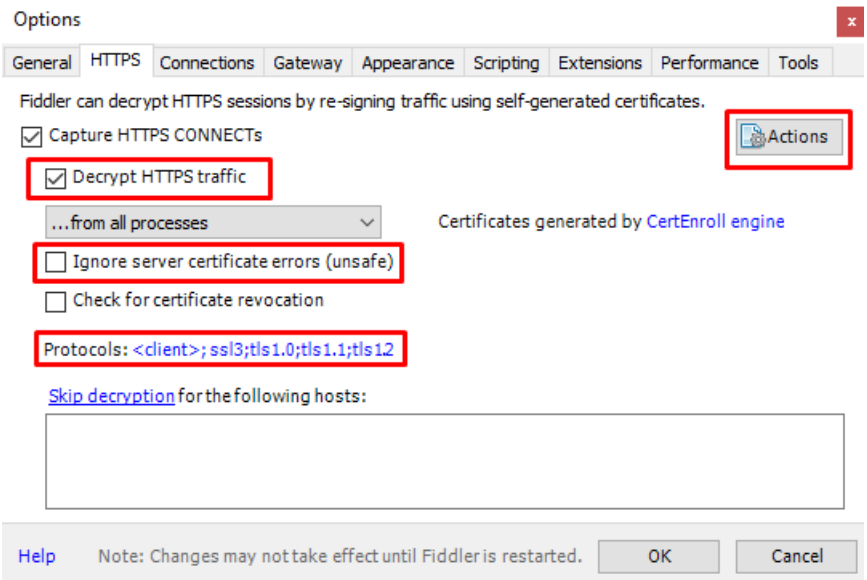
Переходим в "*Tools -> Options...*".

Вкладка "HTTPS".

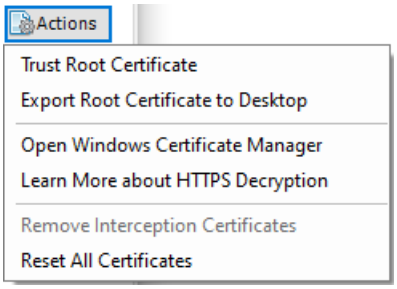
После установки Fiddler не собирает HTTPS-трафик, это необходимо включить. Ставим галочку в опции "*Decrypt HTTPS traffic*". После этого Fiddler сгенерирует самоподписанный сертификат и спросит хотите ли установить данный сертификат. Отвечаем да.

Опция "*Ignore server certificate errors (unsafe)*" - сразу можно не включать. На некоторых порталах бывают ошибки сертификатов, но это редко. Как увидите так включите).

Настройка протоколов. По умолчанию стоит значение "<client>;ssl3;tls1.0". Советую сразу установить значение на "<client>;ssl3;tls1.0;tls1.1;tls1.2". После изменения настроек необходимо перезапустить программу чтобы настройки вступили в силу.



Кнопка "*Actions*":



"*Trust Root Certificate*" - если сгенерированный Fiddler сертификат вы не установили после включения опции "Decrypt HTTPS traffic", то можно это сделать здесь.

"*Export Root Certificate to Desktop*" - если вы планируете использовать Fiddler как прокси-сервер локальной сети, то на каждом устройстве пользователя необходимо установить сгенерированный выше сертификат. С помощью этой опции сохраняете сертификат на ваш рабочий стол.

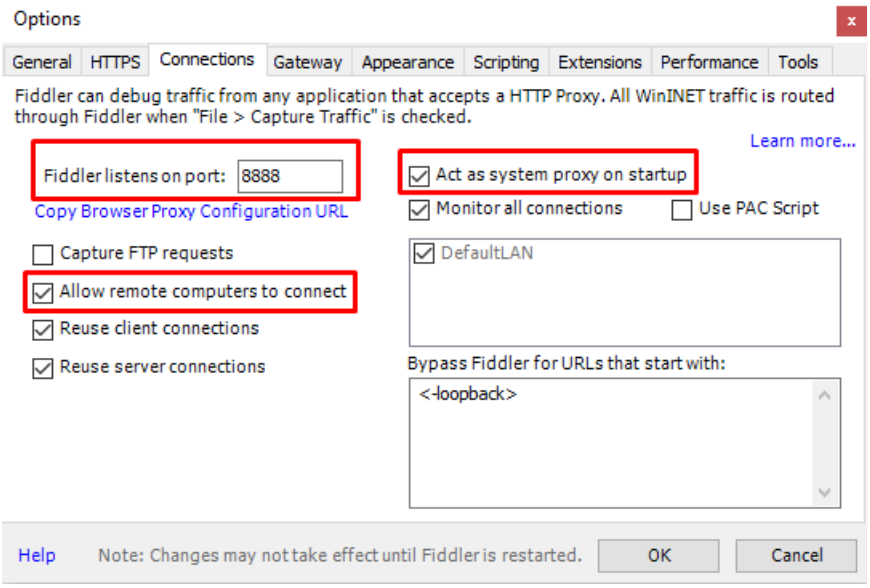
"*Reset All Certificates*" - в некоторых случаях необходимо сгенерировать новый сертификат взамен старого. В этом случае сбрасываем все Fiddler-сертификаты и генерируем новый сертификат.

Вкладка "Connections".

Здесь устанавливаем на каком порту Fiddler работает как прокси-сервер. Порт по умолчанию "8888".

"*Allow remote computers to connect*" - включаем опцию чтобы Fiddler начал принимать подключения от других компьютеров.

"*Act as system proxy on startup*" - по умолчанию опция включена. Если включена, то при запуске опция "File -> Capture Traffic" включена.



После изменения данных настроек необходимо перезапустить программу чтобы настройки вступили в силу.

Вкладка "Gateway".

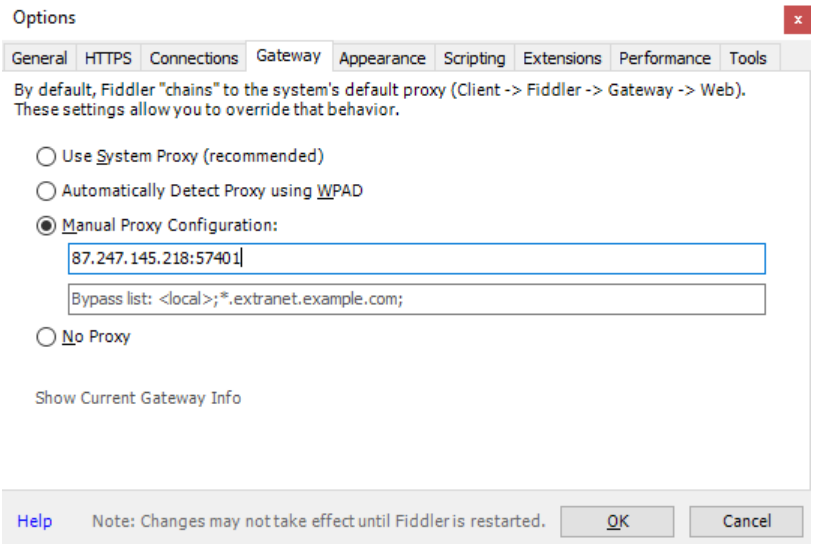
Здесь устанавливаем куда Fiddler отправляет входящие пакеты, какой прокси использует.

"*Use System Proxy (recommended)*" - использование системного прокси из реестра текущего пользователя.

"*Manual Proxy Configuration*" - возможность задать вручную прокси-сервер.

"*No proxy*" - задаем что выход в Интернет напрямую, без использования прокси.

После изменения данных настроек необходимо перезапустить программу чтобы настройки вступили в силу.

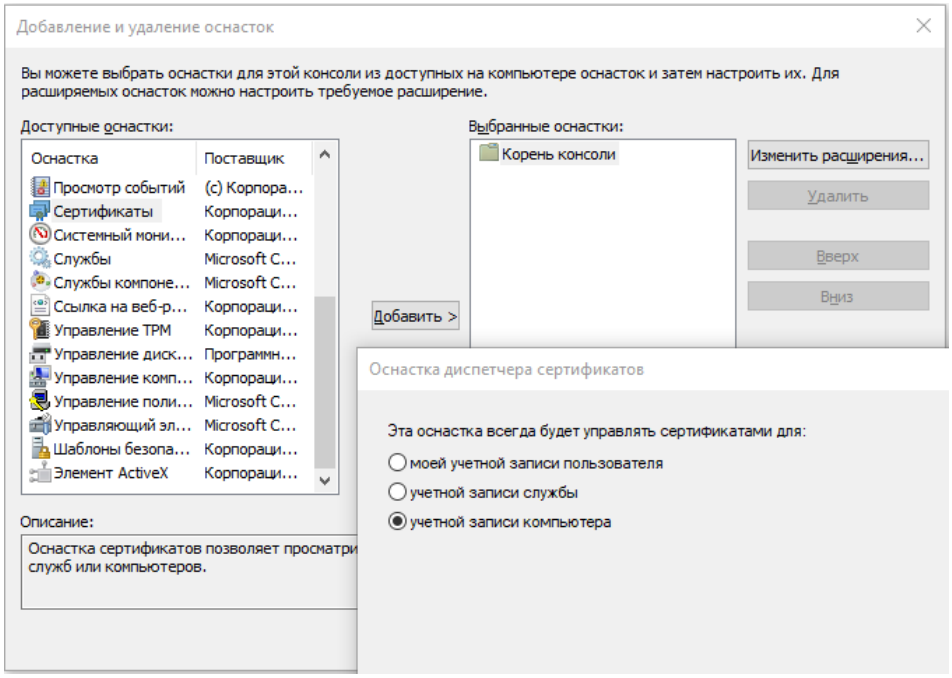


Установка сертификатов на Windows устройствах

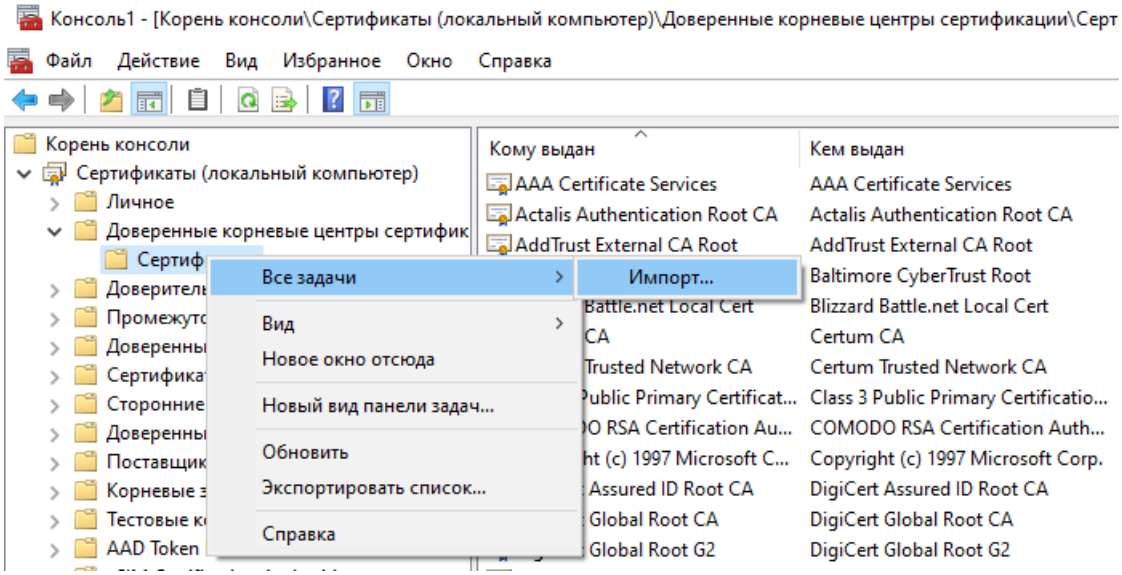
После того как сгенерированный сертификат скопирован на рабочий стол этот сертификат необходимо установить на каждое устройство которое будет использовать данный Fiddler в качестве прокси-сервера.

Для установки сертификата используем консоль управления MMC: в коммандной строке вводим команду "mmc".

В меню файл выбираем "*Добавить или удалить оснастку*". Из доступных оснасток выбираем "*Сертификаты*" и с помощью кнопки "*Добавить*" выбираем данную оснастку. Нажимаем "Ок" и выбираем "*учетной записи компьютера*". Это нужно чтобы открыть сертификаты которые установлены для всего компьютера, а затем установить сертификат Fiddler именно в это хранилище. Если открыть сертификаты "*моей учетной записи пользователя*", то после установки сертификата Fiddler в это хранилище другие пользователи данного компьютера не смогут подключиться к Fiddler.



Установку сертификата производим в "Доверенные корневые центры сертификации".



Если ваши компьютеры находятся в домене, то используйте инструменты домена для установки сертификата каждому пользователю или на каждый компьютер сети.

Настройка прокси на Android

Чтобы отображались запросы приложения Android, вам необходимо разрешить удаленное подключение. Для этого перейдем во вкладку Tools -> Options. В открывшемся диалоговом окне нам необходимо выбрать вкладку Connections.

Далее необходимо выбрать чекбокс «Allow remote computers to connect» и «Capture FTP requests». Fiddler теперь прослушивает порт 8888 (это порт по умолчанию, вы можете изменить его из приведенной выше настройки). Для применения настроек, нажимаем «OK» и перезапускаем Fiddler.

Теперь нам необходимо настроить наше Android устройство. Возьмите в руки телефон, откройте Свойства сети → Название сети WiFi → Прокси-сервер → Вручную → Имя хоста: *ваш IP* / Порт: *8888* → Сохраните измененные свойства сети (указывается IP адрес вашего ПК, на котором установлен Fiddler).

Перейдем по ссылке http://ipv4.fiddler:8888 и скачаем FiddlerRoot certificate, далее начнется автоматическая загрузка сертификата. Откройте его, задайте имя сертификата, и теперь у вас есть доступ к трафику Android-приложения.

Не забываем нажать на «Response body is incoded. Click to decode.», чтобы декодировать ответ.

Настройка прокси на iOS

Возьмите iPhone, откройте Свойства сети → Название сети WiFi → Прокси-сервер → Вручную → Имя хоста: *Наш IP* / Порт: *8888* → Сохраните измененные свойства сети.

Теперь необходимо перейти по ссылке http://ipv4.fiddler:8888, скачать FiddlerRoot certificate, «Разрешить» загрузку профиля конфигурации.

Далее перейдите в Настройки → Профиль загружен → Установить.

Затем перейдите в Настройки → Основные → Об этом устройстве → Доверие сертификатам → найдите установленный сертификат и сделайте его «Доверенным».

Операции над запросами

Справа, в окне, где находятся request и response, располагаются дополнительные инструменты:

Statistics — позволяет получать различные статистики как по одному запросу, так и по пачке выделенных;

Inspectors — дает возможность просматривать в различном виде заголовки и данные запроса;

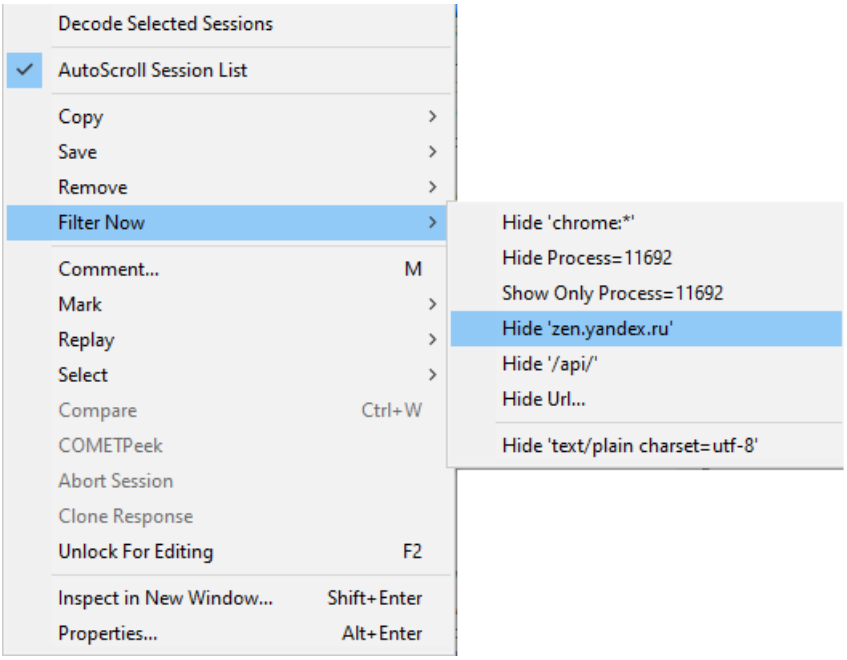
Filters — позволяет следить за конкретными запросами;

Timeline — визуальное представление выполненных запросов на временной шкале.

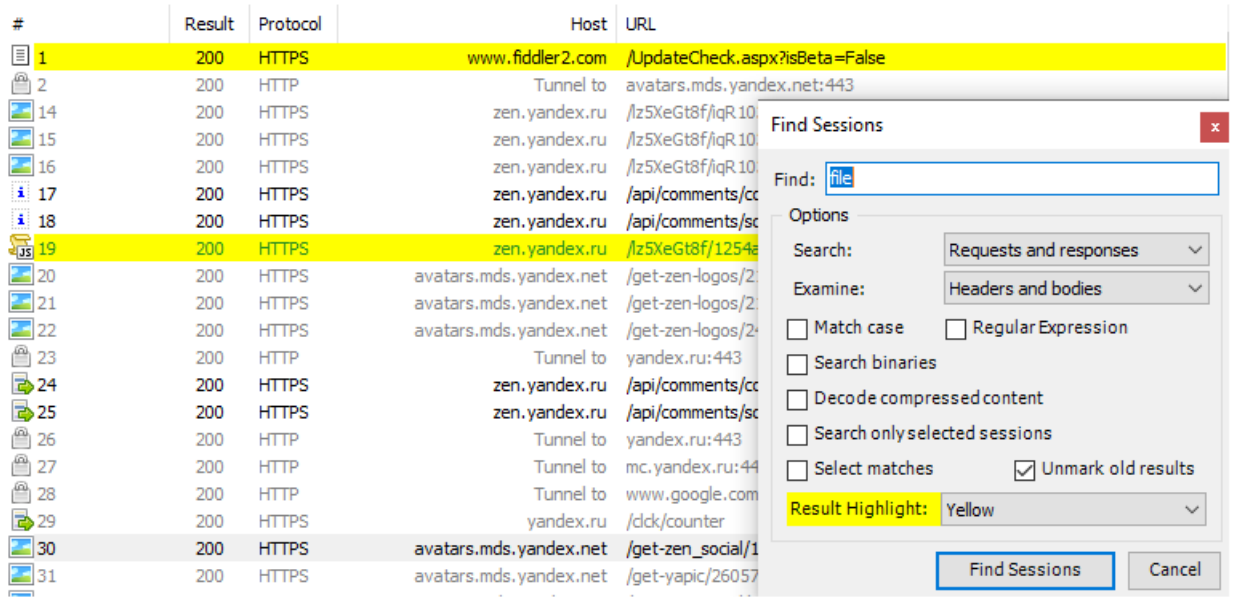
Анализ трафика

В процессе работы Fiddler сниффит все HTTP-запросы и их обычно много. Для поиска необходимых запросов можно использовать фильтры. Правой кнопкой мыши

выбираем лишний запрос, выбираем "*Filter Now*" и "*Hide '...'*" чтобы скрыть запросы к данному домену. Можно удалять вручную выделенные запросы используя кнопку "*Delete*".



Кроме использования фильтров можно искать отдельный текст в теле запросов/ответов: "*Ctrl+F*" для открытия меню поиска. Найденные запросы подсвечиваются по умолчанию желтым цветом.



Подмена данных

В Fiddler Classic есть несколько вариантов подмены данных, рассмотрим некоторые из них:

Automatic Breakpoints

Breakpoint — это некая точка остановки запроса. Когда обнаруживается запрос из заданного списка, он отображается справа и с ним можно взаимодействовать.

Можно поставить breakpoint на request. Перейдем в **Rules** -> **Automatic Breakpoints** и выберем «**Before requests**». Выполним запрос - request запроса отображается справа. Мы его можем отредактировать. Для того, чтобы отправить запрос, нажмем **Run to Completion**. Наш запрос отправится на сервер.

А можно изменить ответ от сервера. Выберем «**After Responses**» в **Rules** -> **Automatic Breakpoints**. После запроса в левой части экрана мы видим ответ от сервера. Внесите необходимые нам изменения и нажмите «**Run to Completion**».

AutoResponder — это некая точка остановки запроса. Когда обнаруживается запрос из заданного списка, он отображается справа и с ним можно взаимодействовать.

Моделирование скорости

Данная возможность предоставляется в скрипте. Вы можете изменить его поведение, перейдя в настройки **Rules** -> **Customize Rules** и отредактировав значения блока скрипта:

```
if (m_SimulateModem) {
// Задержка отправки на 300 мсек на каждый загруженный КБ.
oSession ["просачивание-задержка"] = "300";
// Задержка приема 150 мс на загруженный КБ.
oSession ["задержка-отклик"] = "150";
}
```

Для включения данной функции перейдите в **Rules** -> **Performance** и выберите пункт «**Simulate Modem Speeds**».

В Fiddler существует инструмент **"*Fiddler ScriptEditor*" (Редактор скриптов)** для создания правил модификации трафика. Запуск редактора скриптов через "*Ctrl+R*" или выбора пункта меню "*Rules -> Customize Rules...*".

В редакторе скриптов есть два основных метода: "*OnBeforeRequest*" и "*OnBeforeResponse*":

"*OnBeforeRequest*" - выполнение скриптов в этом методе происходит перед отправкой пакетов на веб-сервер.

"*OnBeforeResponse*" - выполнение скриптов в этом методе происходит после получения ответа от веб-сервера.

Ниже приведены примеры скриптов с указанием в каком методе их расположить.

Задача 1: Запрет сайта

Запрещаем переход на адрес сайта содержащий строку.

```
// OnBeforeRequest
if (oSession.uriContains("//ya.ru/")) oSession.host =
'access.denied';
```

Задача 2: Запрет загрузки ресурса

Запрещаем загрузку "*.svg*" файлов для заданного адреса сайта.

```
// OnBeforeRequest
if (oSession.uriContains("yastatic.net") &&
oSession.url.EndsWith(".svg"))
{
            oSession.host = 'na.derevnu.dedushke';
}

// или

// OnBeforeRequest
if (oSession.uriContains("yastatic.net") &&
oSession.url.EndsWith(".svg"))
{
            oSession.responseBodyBytes = new byte[0];
}

// OnBeforeResponse
if (oSession.uriContains("yastatic.net") &&
oSession.url.EndsWith(".svg"))
{
            oSession.ResponseBody = null;
}
```

Задача 3: Переадресация запроса

Переадресация запроса на адрес сайта содержащий строку.

```
// OnBeforeRequest
if (oSession.uriContains("//ya.ru/")) oSession.url = "yandex.ru"
```

Задача 4: Сбор данных

Пользователи подключаются через данный прокси-сервер и делают в браузерах некоторые запросы вида "*https://myhost.ru?key=abcd&vin=VF38BLFXE81078232&lang=ru*". Задача записать в базу данных событие поиска и передать значение vin-номера. Данный скрипт создает файлы с названием включающем vin-номер. Кроме скрипта необходимо создать утилиту/службу, которая раз в заданный интервал читает каталог "*C:\vinsearch*" и записывает данные в базу данных.

```
// OnBeforeResponse
if(oSession.uriContains("https://myhost.ru?key=") &&
oSession.uriContains("&vin="))
{
            oSession.utilDecodeResponse();
            // поиск позиции индекса начала vin-номера
            var startVin = oSession.url.IndexOf("vin=") + 4;

            // поиск позиции индекса конца vin-номера
            var endVin = oSession.url.IndexOf("&", startVin);

      // поиск подстроки зная индекс начала и индекс конца подстроки
            var vin = oSession.url.Substring(startVin, endVin -
startVin);
```

```
            // создание файла с именем типа
vin_текущееЗначениеМиллисекунд.txt
            oSession.SaveResponseBody("C:\\vinsearch\\" + vin + "_"
+ DateTime.Now.Millisecond + ".txt");
}
```

Задача 5: Изменить текст в ответе

В данном примере меняем текст "*Иванов*" на "*Петров*".

```
// OnBeforeResponse
if (oSession.uriContains("https://myhost.ru"))
{
            oSession.utilDecodeResponse();
      oSession.utilReplaceInResponse('Иванов','Петров');
}
```

Задача 6: Заменить ресурс веб-портала на локальный ресурс

Заменим картинку веб-портала на картинку расположенною на локальном диске.

```
// OnBeforeResponse
if (oSession.uriContains("/Static/app/img/world.svg"))
{
            oSession.LoadResponseFromFile("c:/scripts/lang.png");
}
```

Задача 7: Изменение свойств HTML-объектов

Например, есть картинка с заданными размерами в HTML и нужно эти размеры изменить.

```
// OnBeforeResponse
oSession.utilReplaceInResponse('/Static/app/img/world.svg"
height="15" width="15" style="height:
15px','/Static/app/img/world.svg" height="1" width="1"
style="height: 1px');
```

Задача 8: Скрыть элементы по className меняя css-файлы

В данном примере скрываем элементы зная их className в css-файле добавляя свойство "*visibility: hidden;*"

```
// OnBeforeResponse
oSession.utilDecodeResponse();
oSession.utilReplaceInResponse("#header_Area_Right {",
"#header_Area_Right { visibility: hidden; ");
```

Задача 9: Заставить страницу открываться в текущем окне

Пример: существует JavaScript, который открывает ссылку в новом окне. Нужно сделать чтобы ссылка открывалась в текущем окне.

```
// OnBeforeResponse
if (oSession.uriContains("myhost.ru") &&
oSession.uriContains(".js"))
{
            oSession.utilDecodeResponse();
            oSession.utilReplaceInResponse("window.open(url,
'_blank', option);", "window.open(url);");
}
```

Задача 10: Выполнение скриптов для определенных IP

В данном примере меняем текст "*Иванов*" на "*Петров*" только для IP = "*192.168.0.100*"

```
// OnBeforeResponse
if (oSession.clientIP == '192.168.0.100')
{
            oSession.utilDecodeResponse();
            oSession.utilReplaceInResponse('Иванов','Петров');
}
```

Задача 11: Меняем css-стили портала

Css-файлы веб-портала можно сохранить на локальном диске, отредактировать и настроить скрипт отдавать стили с локального диска, а не с портала.

```
// OnBeforeResponse
if (oSession.uriContains("/banner.css"))
{
            oSession.LoadResponseFromFile("c:/scripts/banner.css");
}
```

Задача 12: Запрет PUT-команды и аналогичных

Запрет команды по ее типу: "*PUT*", "*DELETE*", etc.

```
// OnBeforeRequest
if (oSession.HTTPMethodIs("PUT") &&
oSession.uriContains("https://myhost.ru/"))
{
            oSession.host = 'access.denied';
}
```

Задача 13: Изменение тела POST-запроса

Изменить тело POST-запроса для заданного портала. При авторизации на данном портале вне зависимости от введенных пользователем данных на веб-портал отправятся данные из скрипта.

```
// OnBeforeRequest
if (oSession.uriContains("https://myhost.ru/") &&
oSession.RequestMethod == "POST")
{
```

```
        oSession.utilSetRequestBody("username=xxx&password=yyy");
}
```

Задача 14: Меняем заголовки HTTP-пакета

Заголовки пакетов можно легко редактировать: удалять, добавлять, изменять.

```
// OnBeforeRequest

// Удалить заголовок с именем 'User-Agent'
oSession.oRequest.headers.Remove("User-Agent");

// Добавить заголовок 'xxx' со значением 'yyy'
oSession.oRequest.headers.Add("xxx", "yyy");

// Изменить значение заголовка с именем 'User-Agent' на значение
'xxx'
oSession.oRequest.headers["User-Agent"] = "xxx";
```

Задача 15: Меняем Cookie

Работа с Cookie: добавление, удаление, редактирование

```
// OnBeforeRequest - добавить в запрос Cookie
oSession.oRequest["Cookie"] = (oSession.oRequest["Cookie"] +
";mycookie=xxx");

// OnBeforeRequest - изменить значение Cookie 'JSESSIONID' на 'xxx'
oSession.oRequest['Cookie'] =
oSession.oRequest['Cookie'].Replace("JSESSIONID=","ignoredCookie=")
+ ";JSESSIONID=xxx";

// OnBeforeRequest - удалить Cookie 'JSESSIONID'
oSession.oRequest['Cookie'] =
oSession.oRequest['Cookie'].Replace("JSESSIONID=","ignoredCookie=");
```