УДК 517.9

А.Ю. Сластухин

Снифферы и их применения в задачах перехвата сетевого трафика

Снифферы и их применения в задачах перехвата сетевого трафика

Сластухин А. Ю. (ЯрГУ им. П. Г. Демидова, Ярославль)
Научный руководитель: , Старший преподаватель Савинов Д. А.

Информация всегда имела большое значение в жизни человека. Её носители были зачастую физическими, например, это могло быть письмо, в котором содержалась важная государственная информация или частная переписка. За носителями информации активно охотились, их перехватывали, заменяли, уничтожали. Пока носитель был простым(это мог быть гонец с письмом), многие понимали важность обеспечения безопасности носителя, но со временем всё координально изменилось: вместо письма теперь пакет, перемещающийся не по дороге, а по сети передачи - теперь люди не видят, как происходит передача, не понимают, как можно на неё повлиять, в этом и заключается проблема: теперь мы не заботимся о безопасности передаваемых данных, возлагая эту ответсвенность на тех, кто реализует инструменты, которыми мы пользуемся.

В данной статье мы рассмотрим инструменты, позволяющие просматривать передаваемые по сети данные, поговорим о том, как эти инструменты можно использовать во вред, как с их же помощью защищаться - то, что является средством нападения, может использоваться и для защиты. Начнём с встроенных в браузер средсв перехвата сетевого трафика - далее такие инструменты будет называть снифферами.

Встроенные средства браузера

Рассмотрим на примере HTTP запросов в браузере Google Chrome version 118.0.5993.89. В других браузерах различия будут незначительны. Заходим на любой сайт и открываем средства разработчика – F12. Переходим в параметры сети – Network. Мы увидим запросы, которые связаны с этим сайтом.

Запросы делятся на запросы клиента и ответа сервера.

1. Запросы(HTTP Requests) - отправляются на сервер с целью вызвать реакцию - обычно это доступ к ресурсу.

2. Ответы(HTTP Responses) - реакция сервера на запрос клиента.

Рассмотрим структуру HTTP запроса:

1. Стартовая строка - имеет описательный характер, в ней указывается версия протокола, код ответа и другая информация, которая часто помогает понять, назначение запроса. Например: GetCurrencyRates скорее всего запрашивает курсы валют.

2. HTTP заголовки - текстовые данные передаваемые в формате <Название заголовка>:<значение>. Заголовки бывают общего назначения. Заголовки запроса - заголовки, описывающие запрос, требования, дополнительный контент. Заголовки представления - описывают формат передаваемых данных и кодировку.

3. Тело запроса - завершающая часть запроса, которая есть у запросов определённых типов. В теле часто передают данные для запроса или ответа. Например: для передачи нескольких атомарных значений можно вполне обойтись GET запросом и передать параметры в строке запроса. Но, если тело большое или является сложным элементом, например, сериализованным обьектом в формате JSON, то используется POST запрос.

Структура ответа аналогична, но добавляется статус ответа, состоящий из кода ответа и текстового описания.

Статусы делятся на категории:

1. 1xx Информационные, не влияют на обработку запроса.

2. 2xx Статусы успешной обработки запроса.

3. 3xx Перенаправление по другому адресу(Redirect).

4. 4xx Ошибка в пользовательском запросе. Возможно переданы не все необходимые данные или они имеют неожидаемый вид - тут рекомендую следить за заголовками и кодировкой.

5. 5xx Внутренняя ошибка сервера. Возможно, пользователь нашёл ошибку в работе приложения на сервере или сервер не смог найти подходящий подходящего ответа(500-й статус).

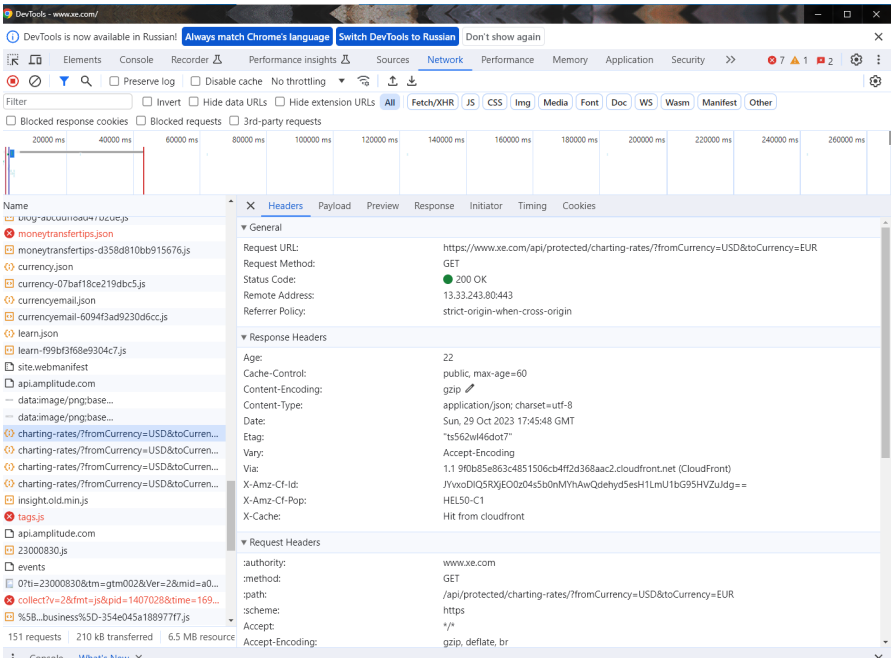


Рис. 1: Пример структуры запроса

Подробнее о протоколе HTTP и его структуре можете прочитать в статье [4].

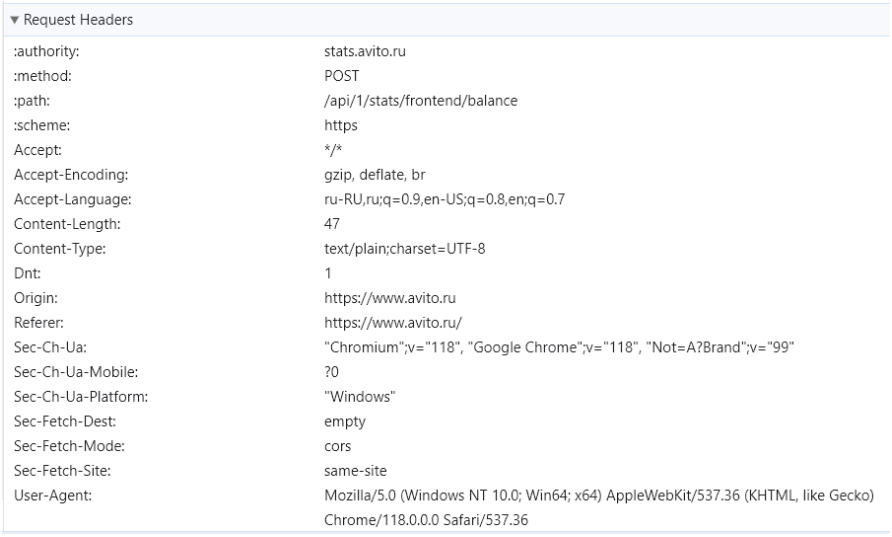
Важно понимать, что помимо заголовков для запроса браузер подставляет туда свои заголовки, в которых содержится используемая пользователем ОС, версия браузера, размер окна пользователя и т.п. Эта информация включается в FingerPrint.

FingerPrint браузера - это информация, собранная о удалённом устройстве с целью анализа и дальнейшего использования.

1. Данная информация зачастую используется для сбора статистики по пользователям, для выявления целевой аудитории - так, например, можно понять, на каких конфигурациях стоит тщательнее тестировать продукт.

2. Эта же информация может быть использована в качестве отпечатка устройства, с помощью которого можно будет идентифицировать отправителя.

3. Помимо живых пользователей к сайту могут обращаться и программы - боты, которые часто собирают информацию, которая публикуется. Поэтому хорошо отличать целевого пользователя от бота.



▼ Request Headers

:authority:	stats.avito.ru
:method:	POST
:path:	/api/1/stats/frontend/balance
:scheme:	https
Accept:	*/*
Accept-Encoding:	gzip, deflate, br
Accept-Language:	ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Content-Length:	47
Content-Type:	text/plain;charset=UTF-8
Dnt:	1
Origin:	https://www.avito.ru
Referer:	https://www.avito.ru/
Sec-Ch-Ua:	"Chromium";v="118", "Google Chrome";v="118", "Not=A?Brand";v="99"
Sec-Ch-Ua-Mobile:	?0
Sec-Ch-Ua-Platform:	"Windows"
Sec-Fetch-Dest:	empty
Sec-Fetch-Mode:	cors
Sec-Fetch-Site:	same-site
User-Agent:	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36

Рис. 2: Пример заголовков запроса

Мы обсудили вопросы просмотра запросов, теперь попробуем их сформировать самостоятельно с помощью Postman(или Postman Interceptor, добавляемый через расширения браузера).

Postman

Это инструмент, позволяющий отправлять запросы. Проанализированный запрос позволяет нам сформировать такие же заголовки, как в браузере, чтобы не было отличий от запроса пользователя, который сформировал браузер - очень важно следить за этим, тк многие сервисы отвлеживают запросы, которые отличаются от запросов браузера(запросов целевого пользователя).

Обратите внимание, что политика использования сервиса может быть разной. Кто-то понимает, что его сервис удобнее использовать с помощью программ и не ограничивает запросы от бота, иногда даже предоставляет платное API, которое облегчает работу с сервисом, позволяя отправлять запросы напрямую, без эмитации пользователя. Но есть и другие, которые готовы предоставить функции своего сервиса только целевому пользователю, например, покупателю в интернет магазине, тк боты не только не приносят прибыль, но зачастую уносят с сервиса данные, усиливающие конкурентов сервиса.

Пример: На avto.ru периодически пользователям приходится проходить проверку на то, что пользователь не бот. Это неэффективный способ проверить пользователя, тк тем, кому нужно, найдут способ обойти эту защиту,

но способ отсеивает тех, кто не готов заморочиться с обходом. Если зайти на авито, то можно увидеть часть обьявлений с avto.ru. Таким образом, сервисы конкурируя за пользователей, возможно, собирают друг у друга информацию, защищаются от попыток конкурентов получить нужную информацию.

Другим примером является тендерная ниша. Цель тендера найти заказчику исполнителя, при чём это хочется сделать быстро и эффективно, чтобы отсеять тех, кто не подходит. Тендеры публикуются на большом наборе площадок для своих клиентов, но существуют компании, которые собирают и агрегируют информацию о тендерах с целью предоставить своим клиентам(исполнителям) те тендеры, которые им подходят. Здесь происходит борьба за информацию. Площадки часто защищаются, но эти попытки редко имеют успех. Всё, что открыто пользователю априори невозможно гарантировано защитить.

Снифферы

Мы рассмотрели примитивный пример на основе анализа трафика в браузере, но это далеко не всё. Существуют утилиты(снифферы), позволяющие ослеживать весь сетевой трафик, проходящий через наше устройство(запросы браузера, вход в любимые игры, запрос обновления - всё, что использует сеть для своей работы).

Например:

1. Fiddler 4 - удобная утилита для сниффинга запросов, но платная. Есть бесплатные версии со сторонних сайтов.

2. WireShark - популярная утилита, активно используется, имеет массу возможностей, гайдов, стоит рассмотреть к использованию.

3. Burp suite - ПО для тестирования веб-приложений на проникновение, возможна функция перехвата и редактирования запроса. Можно использовать как для защиты, так и для атаки. На официальном сайте доступен широкий курс обучения обеспечению информационной безопасности.

Используя снифферы, можно перехватывать и анализировать трафик, искать уязвимости в существующих сервисах.

Пример(политика безопасности) из личной практики: существует европейский сервис, поставляющий по платной подписке данные о курсах различных мировых валют. Но на своём сайте, они выводят графики курсов

валют даже неавторизованному пользователю. Заметив это и захватив уходящий запрос и приходящий ответ, удалось выявить уязвимость, которая позволила выгрузить бесплатно информацию о курсах за последние 10 лет, потратив на по одному запросу на валюту. Уязвимость была в том, что сервер возвращал все данные по конкретному курсу, фильтрация точек и построение графика происходила на клиенте.

Помимо этого, неавторизованные клиенты отправляли запросы к API с одним авторизационным токеном для всех неавторизованных пользователей, который обновлялся, примерно, раз в сутки. Это мешает отследить число запросов неавторизованного клиента к API, ограничить действия клиента.

Очевидно, что такого не должно было быть.

Обратите внимание, что пример приведён для Web приложения, но любые приложения, работающие через сеть, отправляют запросы и их тоже можно анализировать и выявлять уязвимости. Так, например, можно перехватить запрос аутентификации пользователя в игре и посмотреть, какие данные и в каком виде отправляются на сервер - бывают случаи, когда проходит атака - SQL инъекция, в результате которой злоумышленник может получить критически важную информацию о сервисе, нарушить его работу. Если хотите узнать подробнее про SQL инъекции, рекоммендую обратить внимание на Burp suite[5] - позволяет проводить эффективные и быстрые атаки такого рода, имеет хорошее описание.

Поймите главное - сейчас человек не старается вникать в те вещи, которые вокруг него происходят, возлагая эту ответсвенность на разработчиков используемых инструментов, но ошибок в разработке очень много, разработчики не волшебники, а люди, которые порой принимают неудачные решения или забывают закрыть 'чёрные ходы', которые были удобны на этапе разработки.

Сниффер - это инструмент, который может помочь найти такие ошибки, закрыть их или использовать в своих целях.

Антисниффферы

Сниффферы возволяют анализировать весь проходящий через устройство трафик. В прошлой главе мы просматривали этот трафик для своих целей, но представьте, если доступ к этим данным получит взлоумышленник - что если на устройстве будет работать сниффер, пересылающий ваш трафик третьему лицу? Всё верно, сниффферы могут использоваться в целях шпионажа за пользователем.

Могут быть похищены данные о сайтах, на которые вы заходите, о ваших приложениях, которые используют сеть, о сетевой активности в целом.

Всё это злоумышленник может использовать в своих интересах. Например: перехватив хэш пароля, можно подобрать пароль, который будет иметь тот же хэш, использовать подобранный пароль для прохождения аутентификации под видом пользователя. Другой пример: обратите внимание на Cookie. Именно они отвечают за то, чтобы пропустить пользователя на сайт без необходимости повторной аутентификации. Хорошие сервисы привязывают Cookie к конкретному устройству или ip адресу, тем самым защищая пользователя от опасности получения Cookie третьими лицами.

Рассмотрим этот пример подробнее:

1. Найдите вкладку Cookie и изучите информацию, которую там найдёте: это будет сооношение (хост, Cookie).

2. Перейдите на сайт площадки RTS Tender.

3. Обратите внимание, что площадка относится к тем, которые блокируют программы, которые пытаются зайти к ним. Поэтому вашему браузеру нужно будет пройти 5 секундную проверку, тк у вас ещё нет файлов Cookie, которые позволили бы сервису сразу удостовериться, что вы не бот.

4. Теперь при повторном переходе на страницы площадки, проверки не будет, тк в первоначальный запрос будет добавлен заголовок Cookie из пункта 1, который будет проверен на сервере и вас пропустят без проверки.

5. Затем попробуйте перенести файл Cookie на устройство с другим IP адресом и повторите пункт 2. Проверка не будет пройдена, тк сервис соотносит Cookie и устройство, которому их выдавал.

Cookie встречаются на уровне Web приложений, ещё испольузется механизм предосталвения токенов доступа, подробнее о них[6]. Это хороший пример того, как нужно заботиться о пользовательской безопасности, но далеко не все это делают. В качестве интереса проделайте то же самое с каким-нибудь приложением.

Так же, можно использовать полученные данные о действиях пользователя в сети в методах социальной инженерии - например, узнать время активности пользователя, когда он дома, и когда его нет, предпочтения пользователя, адрес электронной почты, профиль социальной сети - использовать полученные данные для Fisching атак и т.п.

Рассмотрим, популярные места, где может скрываться угроза.

Где может скрываться угроза?

1. Нелицензированное ПО или ПО с закрытым исходным кодом (Мы не знаем заранее, какие зловредные модули внутри есть и какие средства защиты предусмотрены)

2. Драйвера устройств - лакомый кусочек для закладки любого опасного кода (их часто качают с неофициальных источников, работают они на низком уровне системы, интегрируя программный и физический уровни, для их установки зачастую требуются повышенные права). Например: логгер клавиатуры, спрятанный в драйвер клавиатуры, который отсылает все вводимые пользователем данные злоумышленнику.

3. Сторонних библиотеки с закрытым исходным кодом, использующиеся в рабочем проекте или открытые библиотеки, использующиеся без проведения проверок(как и в первом случае, мы не можем гарантировать безопасность их использования) - поэтому, например, наши гос структуры при лицензировании ПО требуют проверить не только код, но и используемые внешние библиотеки.

Лучше не допускать появления подобного сниффера-шпиона. Рассмотреть, что делать, если он уже есть в системе.

Что может помочь найти угрозу?

1. Снифферы могут выступать и в роли антиснифферов-контролируя трафик, они могут выявлять активность вредоносного ПО по известным зловредным сигнатурам - это одна из базовый мер противодействия таким угрозам. В качестве такого антисниффера предлагаю рассмотреть Wareshark[1].

2. Jenkins - программная система с открытым исходным кодом на Java, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения, которая так же содержит инструменты для выявления уязвимостей в используемых библиотеках и написанном коде.

3. Самостоятельный просмотр собственной сетевой активности - согласитесь, будет странно, если какое-то приложение будет вести сетевое взаимодействие без вашего ведома. Считаю, что в случае поддержания серьёзных систем самостоятельный мониторинг сети необходим, как и в случае с аномальными запросами к базе данных.

Косвенные признаки наличия угрозы

Рассмотрим косвенные признаки, указывающие на работу вредоностного ПО.

1. Система стала тормозить - вероятно, что что-то потребляет её ресурсы для собственной деятельности - хорошо понять, что это.

2. Уровень соединение с интернетом переключается с более высокого на более низкий и обратно. Например, для перехвата ваших данных спец службам не нужно ставить дополнительное программное обеспечение на ваше устройство, можно перенаправить ваш трафик через сетевой узел, который будет анализироваться и вы об этом не узнаете.

3. Вырастает энергопотребление(на аккумуляторных устройствах это заметно - разряжаются в несколько раз быстрее) - тк устройство взаимодействует с сетью больше обычного. Пример: Это может быть как встроенный в устройство механизм, который заложен производителем или распространителем для возможной слежки за покупателем - последнее время участились случаи, когда новые устройства содержали такие механизмы.

Злоумышленники находят уязвимости в наших системах,поступая творчески, поэтому нам для борьбы с угрозами нельзя опираться только на ПО, которое магическим образом отловит весь вредоносный трафик в нашей сети. Существуют угрозы, которые до сих пор не всегда отлавливаются антиснифферами, их можно обнаружить, только просматривая трафик самостоятельно.

Примером такой угрозы может служить BPFDoor-Berkley Packet Filter[2], которая была обнаружена исследователями Deep Instinct. Deep Instinct отмечает, что на момент анализа новая версия BPFDoor не определялась как вредоносная ни одним из доступных антивирусных движков на VirusTotal, хотя впервые она появилась на платформе еще в феврале 2023 года.

Заключение

Сниффер - полезный инструмент, который может использоваться как для изучения построенных систем с целью выявления уязвимостей, так и для шпионажа(анализатор трафика выступает в качестве вредоносного ПО на устройстве жертвы), так и для борьбы с ним за счёт анализа сетевой активности с использованием механизкмов определения угроз по известным сигнатурам.

Помните, что снифер это лишь инструмент, который мы можем использовать в работе, но нам не стоит палагаться только на него- нужно подходить творчески и подобно злоумышленникам находить уязвимости там, где их, кажется, нет.

Основная проблема связана с отношением современного человека к знаниям - мы не хотим вникать в "сложные как нам кажется, процессы, перекладываем ответственность за безопасность на других - поэтому старайтесь вникать в современные процессы, разбирайтесь с тем, как работает то или иное устройство или как устроен тот или иной процесс. Вспомните, с чего мы начинали - с передачи информации, хранимой в письме, проблемы обеспечения безопасности очевидны, а сейчас, как вы видите, то же самое, но другие средства представления и передачи информации, а необходимость обеспечить безопасность никуда не делась, однако мы не задумываемся сейчас об этом - и зря. Описанная проблема пересекается с проблемой отчуждения человека, описанной Карлом Марксом в 19-м веке - я наблюдаю эту проблему и в современном обществе, считаю её одной из самых важных проблем 21 века.

Если захотите подготовить на основе этой статью доклад, можете воспользоваться презентацией, которую я опубликовал вместе с текстом в GitHub репозиторий[3].

Литература

1. https://telegra.ph/Wireshark—uchimsya-skanirovat-set-04-01
2. https://xakep.ru/2023/05/15/new-bpfdoor
3. https://github.com/SkibaSAY/Doclad_Sniffers
4. https://selectel.ru/blog/http-request/
5. https://portswigger.net/web-security
6. https://habr.com/ru/articles/534092/

Ярославский государственный университет им. П.Г. Демидова