

УДК 517.9

А.Ю. Сластухин

Снифферы и их применение в задачах перехвата сетевого трафика

Снифферы и их применение 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 ответственность 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 средств перехвата сетевого трафика - далее такие инструменты будем 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 объектом 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 - 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 ответ(500-й статус).

X Headers Payload Preview Response Initiator Timing	
▼ General	
Request URL:	https://stats.avito.ru/api/1/stats/frontend/generic
Request Method:	POST
Status Code:	● 200 OK
Remote Address:	146.158.48.2:443
Referrer Policy:	strict-origin-when-cross-origin
▼ Response Headers	
Access-Control-Allow-Origin:	*
Content-Encoding:	gzip
Content-Type:	application/json; charset=utf-8
Date:	Tue, 07 Nov 2023 16:09:29 GMT
Server:	nginx
X-Content-Type-Options:	nosniff
X-Frame-Options:	SAMEORIGIN
X-Xss-Protection:	1; mode=block
▼ Request Headers	
:authority:	stats.avito.ru
:method:	POST
:path:	/api/1/stats/frontend/generic
: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:	864
Content-Type:	text/plain;charset=UTF-8

Рис. 1: Пример заголовков запроса

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

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.

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, т.к. многие сервисы отвлеживаются на запросы, которые отличаются от запросов браузера(запросов целевого пользователя).

Политика сервиса может быть разной. Кто-то понимает, что его сервис удобнее использовать с помощью программ и не ограничивает запросы от бота, иногда даже предоставляет API, которое облегчает работу с сервисом, позволяя отправлять запросы напрямую, без имитации пользователя. Но есть и другие, которые готовы предоставить функции своего сервиса только целевому пользователю, например, покупателю в интернет магазине, т.к. 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. Но он отсеивает тех, кто не обладает нужными знаниями. Если зайти на авито, то можно увидеть часть объявлений 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, причём 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 пример на основе анализа трафика в браузере, но это далеко не всё. Существуют утилиты(снифферы), позволяющие отслеживать 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 sniffing 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.

Обращаем 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. Для подробного изучения SQL инъекций рекомендую обратиться на сайт Burp suite[5] - инструмент позволяет проводить эффективные атаки такого рода.

Главное - сейчас человек не старается вникать в те вещи, которые вокруг него происходят, возлагая эту ответственность 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.

Антиснифферы

Снифферы возволяют анализировать весь проходящий через устройство трафик. В прошлой главе мы просматривали этот трафик для своих целей, но представим, что доступ к этим данным получит злоумышленник - что если на устройстве будет работать сниффер, пересылающий трафик пользователя третьему лицу?

Могут быть похищены данные о сайтах, на которые заходит жертва, о её 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: Обратим 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 подробнее:

1. Cookie можно найти в средствах разработчика в браузере, они представляются соотношением (хост, Cookie).
2. Перейдём на сайт площадки RTS Tender.
3. Обратим внимание, что площадка относится к тем, которые блокируют программы, которые пытаются зайти к ним. Поэтому браузеру нужно будет пройти 5 секундную проверку, т.к. у него ещё нет файлов Cookie, которые позволили бы сервису сразу удостовериться, что пользователь не бот.
4. Теперь при повторном переходе на страницы площадки, проверки не будет, т.к. в первоначальный запрос будет добавлен заголовок Cookie из пункта 1, который будет проверен на сервере, пользователя пропустят без проверки.
5. Затем попробуем перенести файл Cookie на устройство с другим IP адресом и повторите пункт 2. Проверка не будет пройдена, т.к. сервис соотносит Cookie и устройство, которому их выдавал.

Cookie встречаются на уровне Web приложений, ещё используется механизм предоставления токенов доступа, подробнее о них[6]. Это хороший пример того, как нужно заботиться о пользовательской безопасности, но далеко не все это делают.

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 Fishing 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 отправляет все вводимые пользователем данные злоумышленнику.
3. Сторонние 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 структуры при лицензировании ПО требуют проверить не только код, но и используемые внешние библиотеки.

Лучше не допускать появления подобного сниффера-шпиона. Рассмотрим, что делать, если он уже оказался 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 Wireshark[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 сетевой активности. Если какое-то приложение будет вести сетевое взаимодействие без ведома пользователя - это можно будет заметить. Считаю, что в случае поддержки серьезных систем самостоятельный мониторинг сети необходим, ведь человеческий глаз порой замечает то, что не замечает программа.

Косвенные признаки наличия угрозы

Рассмотрим косвенные признаки, указывающие на работу вредоносного ПО.

1. Система стала тормозить - вероятно, что что-то потребляет её ресурсы для собственной деятельности - хорошо понять, что это.
2. Уровень соединения 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 устройство, можно перенаправить трафик через сетевой узел, который будет анализироваться.
3. Вырастает энергопотребление(на аккумуляторных устройствах это заметно - разряжаются в несколько раз быстрее) - т.к. 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 пользовательских системах, поступая творчески, поэтому для борьбы с угрозами нельзя опираться только на ПО, которое магическим образом отловит весь вредоносный трафик. 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 механизмов определения угроз по известным сигнатурам, но не стоит полагаться только на него - нужно подходить творчески и подобно злоумышленникам находить уязвимости там, где их, кажется, нет.

На GitHub репозитории[3] дополнительно опубликована презентация, использованная автором на выступлении по этой теме.

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