Снифферы и их применение в задачах перехвата интернет трафика.

Введение

Снифферы

Это программы - анализаторы, способные перехватывать и анализировать сетевой трафик.

Снифферы отличаются следующими особенностями:

* Поддерживаемые сетевые протоколы
* Качество декодирования протоколов
* Наличие пользовательского интерфейса
* Доступ к статистике
* Просмотр трафика в реальном времени

Начнём своё знакомство с встроенных в браузеры анализаторов сети.

Их преимущество заключается в том, что их не нужно устанавливать отдельно, можно использовать хоть сегодня вечером, читая статью на любимом форме.

Анализатор сети браузера отлавливают все отправленные и полученные запросы, фильтруют их и выводят нам те, которые относятся к хосту.

Использование даже встроенных анализаторов позволяет нам заниматься ревирс инжинирингом сайта, мы можем отследить, как клиентская сторона (наш браузер) взаимодействует с сервером.

Заходим на любой сайт и открываем средства разработчика – F12.

Переходим в параметры сети – Network.

Выполним любое действие на сайте, например, пройдём авторизацию или применим фильтры.

Возвращаемся на вкладку Сети и видим, что появилась информация о запросах:

-Тип запроса(Get, Post и т.д.)

-Адрес(какому API обращается клиент)

-Тело запроса(если оно есть)

-Заголовки(Headers)

1.Обратите внимание на заголовки, помимо служебной информации, тут вы найдёте данные о клиенте, такие как версия браузера, размер окна и тд. Данная информация зачастую нужна для сбора статистики по пользователям, для выявления целевой аудитории.

2.Но эта же информация может быть использована в качестве отпечатка вашего компьютера, с помощью которой можно будет идентифицировать отправителя (API получает ведь и ip, а зная ip, можно вычислить организацию, в которой определить компьютер отравителя по данным о его компьютере)

3.Помимо этого, на стороне сервера можно попытаться отсеять часть запросов, отправленных не людьми, но это скорее проверка на дурака, чем 100% решение такой проблемы как прасинга.

Следующим шагом будет имитация прочитанных запросов.

Можем использовать, например, Postman, в котором сформируем запрос, аналогичный запросу клиента.

Формируя и отправляю запрос, мы сможем понять, какие части критичны для получения ответа от сервера (это уникальный момент для каждого API, зависит от программиста, писавшего API).

Когда нам удалось построить валидный запрос, можем перенести его формирование в своё приложение, которое будет обрабатывать ответы сервера, агрегировать нужную нам информацию.

Вернёмся к теории.

Снифферы позволяют перехватывать весь сетевой трафик, идущий от вас и к вам. Поэтому с помощью них можно

-производить перехват данных, передаваемых между сервером и клиентом

-выявлять активность вредоносного ПО, паразитный трафик, который нагружает сеть (для этого существуют утилиты)

- искать уязвимости в архитектуре сервиса, изучая особенности клиент серверного взаимодействия

Я бы хотел остановиться на последнем применении. Это очень творческая и интересная задача, поскольку нужно видеть потенциальные уязвимости там, где их, на первый взгляд, нет.

1.Рассмотрим пример одного разработчика:

При авторизации в приложении он заметил кнопку выбора роли, это привлекло его внимание. Он запустил сниффер и посмотрел на структуру запроса: запрос содержал форму, в которой помимо логина и пароля содержалась роль = 0. Разработчик отредактировал запрос, начал посылать в поле «роль» различные значения. И на значении 6 сервер выдал успешный статус – пользователь авторизовался с правами администратора, не имея таковых.

Ошибка: Нарушение принципов проектирования клиент-серверного взаимодействия: пользователь отправляет данные на сервер, где они обрабатываются, пользователю возвращается результат. Проще говоря – все расчёты и логика работы должна быть скрыта от клиента за API интерфейсом.

В данной ситуации клиент сам себе присваивал роль, что недопустимо. Сервер должен был по полученному логину её определить и присвоить.

2.Рассмотрим другой пример из моей недавней практики:

Для расширения статистики продукта моей команде потребовалось найти историю курса конвертации различных валют(около 160 шт) в рубли за последние 5 лет с периодом агрегации в 1 дня. При этом нужен был проверенный источник, работающий чётко, без видимых ошибок в работе – ЦБ не подходил. Начал исследование возможных источников с зарубежной площадки, которая продаёт информацию о корректных курсах валют, я решил с известном нам площадки, которая предоставляет платный доступ к своему API с ограничением на число запросов – когда-то давно, у нас даже был доступ к их API, но его ограничений было недостаточно для выполнения задачи, а в сложившихся обстоятельствах, когда с русскоязычными компаниями не хотят работать, получить новое не представлялось возможным. Я обратил внимание на возможность посмотреть историю курса конвертации валюты в виде графика, будучи неавторизованным пользователем. Для построения графика скриптами нужны данные, которые клиент получает от сервера. Запустив Fiddler 4 и проанализировав трафик, мы обнаружили, что сервер присылает json с данными за 10 лет, при этом в качестве авторизации сервер принимает один и тот же авторотационный заголовок для всех неавторизованных пользователей – проверили опытным путём. Так же, из этого же заголовка удалось извлечь логин и пароль, для которых был сформирован заголовок. Таким образом, за 160 запросов удалось получить все необходимые данные, используя стандартный заголовок неавторизованного пользователя.

Ошибка: Архитектура позволяет клиенту парсить данные, обращаться к API со стандартным заголовком авторизации – считаю, что это потенциальная дыра, через которую можно организовать атаку «Отказ в обслуживании» (или DoS) – на запросы со стандартным заголовком не должно быть ограничений по количеству запросов (не считая ограничений на число запросов с конкретного Ip адреса, но DoS атака обычно производится с большого числа Ip адресов)

Считаю, что правильно использовать принцип белого листа – не разрешать работать с API неавторизованным пользователям.

До этого момента мы рассматривали использовали анализатор для перехвата своего трафика. Это легальный перехват трафика.

Но очевидно, что есть и нелегальный.

Нелегальный перехват заключается в шпионаже за пользователями сети. Злоумышленник может получить информацию о посещаемых сайтах, программах, которые работают на компьютере жертвы и используют сеть, отлавливать данные, получаемые и отправляемые пользователем, в частности, логинах и паролях, передаваемых в незашифрованном виде.

Любая организация и любой пользователь могут оказаться под угрозой сниффинга, особенно если они обладают ценной информацией.

Где, например, можно встретить такую угрозу?

* Нелицензированное ПО или ПО с закрытым исходным кодом (Мы не знаем заранее, что туда встроено)
* Драйвера на устройства (для их установки зачастую требуются повышенные права, а это потенциальный доступ для закладок, в частности, существуют опасность логирования клавиатуры с последующей отправкой злоумышленнику)
* Использование в рабочих проектах сторонних библиотек с закрытым исходным кодом или без проведения проверок.

В больших проектах, часто используют преддеплойные проверки, которые помимо компиляции, проведения сценарных и модульных тестов, включают в себя проверку использованных зависимостей на уязвимости по базам.  
Пример: Jenkins

Jenkins — программная система с открытым исходным кодом на Java, предназначенная для обеспечения процесса непрерывной интеграции программного обеспечения.

Существует несколько способов защиты:

1.Шифрование

2.Применение так называемых антиснифферов – программные средства, позволяющие выявить перехват трафика.

Используйте криптозащиту совместно с антиснифферами, поскольку шифрование не скрывает сам факт передачи вами информации по сети.