АВТОМАТИЧЕСКОЕ МАРКИРОВАНИЕ СЕТЕВОГО ТРАФИКА БРАУЗЕРА ДЛЯ АНАЛИЗА И КЛАССИФИКАЦИИ НА ПРИМЕРЕ ПЛАТФОРМЫ "REMOTETOPOLOGY"

DOI: 10.36724/2072-8735-2022-16-12-17-22

Manuscript received 02 October 2022;
Accepted 10 November 2022

Уймин Антон Григорьевич,
РГУ нефти и газа (НИУ) имени И.М. Губкина, Москва, Россия, ***au-mail@ya.ru***

Ключевые слова: *данные, соединение, http, сетевой трафик, анализ, ML модель*

Постановка задачи: Анализ сетевого трафика является стандартом безопасности в большинстве систем удаленной работы. Эта задача становится сложение с каждом днем в виду применения систем шифрования на возрастающем количестве устройств. Традиционный метод исследования трафика в модели OSI не дает должного уровня достоверности. Метод Machine learning – один из перспективных методов решения проблемы анализа сетевого трафика. На первом месте стоит выявление нежелательного трафика, который может быть интерпретирован, как инструмент обхода системы. Для того чтобы система могла использоваться в учебном и производственном процессе, необходимо учитывать вопрос эргономики решения. Влияние системы контроля и анализа на скорость отклика. Целью работы является: формирование маркированных данных HTTP/HTTPS трафика, для перехвата шифрованного трафика туннелей. Предлагается разработать плагин для браузера, задачами которого является сбор данных из браузера пользователя с использованием Chromium API. Собранные данные объединяются с сетевыми потоками для формирования маркированных данных используемых в дальнейшем для наборов данных обучение ML модели. Новизна: элементами новизны представленного решения являются использование Machine learning для классификации сетевого трафика основаны на маркированных данных. Результат: использование представленного решения по автоматическому маркированию сетевого трафика браузера для анализа и классификации Мы решили рассмотреть вариант работы каждой системы в независимом шифрованном канале. Запуск проекта RemoteTopology был переведен на HTTP, при этом каждое подключение выполняется в своей VPN сессии и гарантирует безопасность соединения. В рамках соединения выполняется маркировка потоков, что позволяет быстро и качественно обрабатывать данные для последующего ML анализа. Практическая значимость: представленное решение предлагается реализовать в рамках платформы "RemoteTopology" выявить в режиме реального времени отклонения между эталонной системой, которая уже опознает пользователя и "нежелательными системами", которые пользователь может использовать для обхода системы защиты.

Информация об авторе:
Уймин Антон Григорьевич, *старший преподаватель кафедры кафедра безопасности информационных технологий, Факультета комплексной безопасности ТЭК, университета ФГАОУ ВО "РГУ нефти и газа (НИУ) имени И.М. Губкина", Москва, Россия*

Для цитирования:
Уймин А.Г. Автоматическое маркирование сетевого трафика браузера для анализа и классификации на примере платформы "RemoteTopology" // T-Comm: Телекоммуникации и транспорт. 2022. Том 16. №12. С. 17-22.

For citation:
Uymin A.G. (2022) Automatic marking of browser network traffic for analysis and classification using the example of the RemoteTopology platform. *T-Comm*, vol. 16, no.12, pp.17-22. (*in Russian*)

Введение

В настоящее время анализ сетевого трафика стал стандартом безопасности в большинстве систем удаленной работы. Эта задача становится сложение с каждом днем в виду применения систем шифрования на возрастающем количестве устройств [1]. Традиционные методы анализа трафика на прикладном уровне модели OSI (The Open Systems Interconnection model) в настоящее время не дают должного уровня достоверности [2]. Инструменты ML (Machine learning) являются одним из перспективных методов решения проблем анализа сетевого трафика. Применение инструментов ML позволяет искать расширенные шаблоны в анализируемых данных.

В качестве платформы исследования рассмотрен проект Remote Topology, предназначенный для дистанционного доступа к ИТ инфраструктуре, обучения и проведения контроля по тогам обучения. Особой задачей стоит выявление «нежелательного трафика», который может быть со стороны хоста пользователя интерпретирован как инструмент обхода системы. В виду того, что система предназначена для повседневного использования в учебном и производственном процессе, необходимо учесть вопрос эргономики решения т.е. влияние системы контроля и анализа на скорость отклика.

Большое количество моделей ML для классификации сетевого трафика основаны маркированных данных [3,4,5], применяемых для обучения. Это является проблемой так как создание маркированных данных трудоемкий и длительный процесс, в том числе с применением ручного труда квалифицированных специалистов. Такая работа не застрахована от ошибок «человеческого» фактора, что приведет к невозможности проверок правильных меток, при указании доли неправильных.

Задачи и результаты исследования

В нашем исследовании ставится проблема, связанная с формированием маркированных данных HTTP HTTPS трафика, при перехвате шифрованного трафика туннелей. В ходе работы применяется разработанный плагин для браузера, задачами которого является сбор сбор данных из браузера пользователя с использованием Chromium API (Application Programming Interface) [6]. Собранные данные объединяются с сетевыми потоками для формирования маркированных данных используемых в дальнейшем для наборов данных обучение ML модели.

Вэб трафик представлен протоколом HTTP. Который не обеспечивает шифрования. Для повышения безопасности HTTP применяется протокол TLS, обеспечивающий шифрование и защиту данных на прикладном уровне модели OSI. Комбинация HTTP и TLS представляет собой Hypertext Transfer Protocol Secure (HTTPS). Данный подход не позволяет перехватывать открытые сообщения, в ходе обмена данными. Применение HTTPS усложняет анализ данных прикладного уровня сети и его дальнейшую классификацию. Наша работа решает эту проблему, путем перехвата данных браузера до шифрования средствами TLS. Применяя стандартные API, мы можем получить доступ к данным каждого запроса и анализировать ответы в режиме реального времени. Дополнительной проблемой, выявленной в ходе этой работы, является нахождение соответствия данных, полученных из браузера и из сетевого потока IP трафика. Проблема заключается в ограниченности информации от среды передачи, что не позволяет корректно связывать HTTP с передаваемым трафиком. Кроме этого, в потоке, как правило присутствует большое количество служебного трафика что зашумляет передачу. Отсутствует необходимая информация от сетевой среды. В качестве идентификатора используется связка имени узла и UUID – Linux или SID – Windows, что позволяет определить уникальную пару сетевого потока и вэб-интерфейса: формируется Host Name Indication (HNI). По возможности при работе используется HNI для уникальной пары сетевой поток вэб-интерфейс. Если HDI отсутствует, то будет применено, основанное на времени соединение. Такое соединение не будет являться уникальным в случае, если пользователь будет поддерживать несколько параллельных соединений.

Основываясь на анализе источников сети интернет, было выявлено отсутствие комплексного инструмента для получения доступа к данным браузера «из вне» т.е. со стороны интернета. Так же отсутствуют свободные инструменты, позволяющие подробно отображать информацию о каждом сетевом запросе.

Наиболее близким по необходимому функционалу, является OSQuery [7]. Он позволяет получить данные о всей операционной системе, собираемые в единую локальную базу данных. К базе данный можно получить доступ посредством SQL-like запросов для сбора информации о системе [8]. В частности, мы можем получить информацию из сетевых протоколов об исходном приложении и процессе в рамках операционной системы. Мы работаем с вэб-соединением, т.е. частным случаем процесса. При этом нам необходимо получать актуальную подпорную информацию о сетевом соединении HTTP/S от расширения браузера. Таким образом оба инструмента могут быть использованы одновременно, что повысит информативность маркированных данных, а также повысит нагрузку на систему.

В работе [9] описано поведение приложений, генерирующих сетевой трафик, приводится методика анализа зашифрованного трафика мобильных приложений, описан подход через парадигму Deep Learning (DL) к анализу сетевого трафика мобильных устройств, приведена критика подхода ML. Рассматривается общая структура систем классификации шифрованного трафика и приводится оценка на наборах данных мобильных пользователей.

В работе [10] рассматриваются задачи классификации сетевого трафика. Определяются характеристики, используемые классификации трафика. Приводятся существующие подходы к решению данной проблемы. Приведен перечень прикладных задач, требующих привлечения компонента классификации и дополнительные требования, проистекающие из особенности основной задачи. Анализируются свойства сетевого трафика, обусловленные особенностями среды передачи, а также применяемых технологий, так или иначе влияющие на процесс классификации.

В работе [11] определяются параметры снятия данных с манипулятора «мышь». Определяются требования для создания dataset, эффективность анализа и извлечения основных признаков из необработанных данных. Рассмотрены модели позволяющие верифицировать пользователя, производится

сравнение результатов их обработок. Рассмотрен порядок предобработки данных для их анализа. На основе предобработки собран dataset для анализа. Описана платформа Remote Topology в рамках которой проводится эксперимент.

Предлагаемая архитектура разделена на две части. Части отличаются по среде, в которой они работают. Пример архитектуры изображён на рисунке 1. Первая среда – это устройство пользователя, где веб-трафик собирается и назначается соответствующему сетевому потоку. Вторая часть находится в сети и часть контрольного зонда, где он маркирует захваченные потоки с использованием собранных данных. Нами используется существующая реализация зонда сети CESNET IPFIXprobe в качестве основы для нашего модуля маркировки [12]. Эта система может даже работать в автономном режиме без необходимости в сетевой связности.

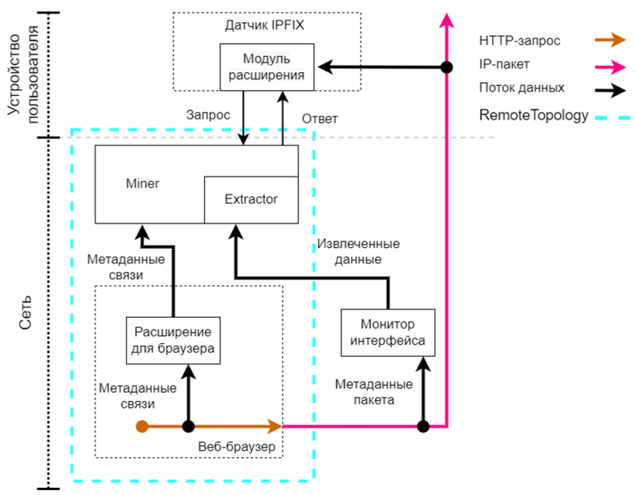

Рис. 1. Архитектура предлагаемой системы

Первая часть системы маркировки - расширение браузера. Его основная задача – перенаправить трафик, чтобы браузер предоставлял данные промежуточным процессам для объединения. В настоящий момент большинство современных браузеров строится на базе Chromium, поэтому API функции браузера Chromium, компании Google можно определить как стандарт «де факто» современного интернета [13]. Это позволяет нам выбрать один браузер для основы эксперимента. Таким образом, разработанное приложение может использоваться в большом перечне браузеров, без необходимости отдельной поддержки каждого из них.

Вторая часть системы маркировка также находится на устройстве пользователя, и его основная задача состоит в том, чтобы контролировать входящий/исходящий сетевой трафик. Предполагаемый монитор интерфейса будет искать сообщения, которые объявляют о начале или конце соединения, используемого для передачи связи HTTP. Это обеспечивает получение советующей информацией из сетевой среды, которая будет использоваться для сопряжения с HTTP.

Третьей частью системы является процесс Miner & Extractor. Они производят объединение данных из браузера и сетевого интерфейса. Эти данные будут доступны не только в режиме реального времени, но и ретроспективно для дальнейшего использования системы.

В качестве источника данных используется проект Remote Topology [14]. В качестве данный обрабатываются данные: сессии telnet, сессии ssh, сессии VMware Remote Console (VMRC), сессии VNC. Данный сессии маркируются в системе средствами API webRequest из API расширения браузера Google. В API webRequest мы выбираем события, которые будут вызваны на различных этапах жизненного цикла запроса. Событие onBeforeSendHeaders инициируется до того, запускается перед отправкой любых данных HTTP и после того, как все заголовки HTTP доступны. Данное состояние можно определить как идеальное, для прослушивания трафика. Пример синтаксиса приведен ниже.

```
browser.webRequest.onBeforeSendHeaders.addListener(
  listener,          // функция
  filter,            // объект
  extraInfoSpec      // необязательный массив строк
)
browser.webRequest.onBeforeSendHeaders.removeListener(listener)
browser.webRequest.onBeforeSendHeaders.hasListener(listener)
```

Событие onCompleted запускается, когда документ, включая ресурсы, на которые он ссылается, полностью загружен и инициализирован в системе. Это эквивалентно событию load [15]. Пример синтаксиса приведен ниже.

```
browser.webNavigation.onCompleted.addListener(
  listener,          // function
  filter             // optional object
)
browser.webNavigation.onCompleted.removeListener(listener)
browser.webNavigation.onCompleted.hasListener(listener)
```

В каждом из этих событий мы можем получить доступ другое подмножество данных в браузере, основанных на описании события. Событие onSendHeaders запускается перед отправкой запроса браузером. На этом этапе мы представляем идентификатор запроса, заголовки, источник запроса и другие данные, предоставленные в соответствии с API webRequest. Мы соберем все доступные данные и временно сохраняем их в массив внутри расширения браузера с идентификатором запроса в качестве ключа, который будет использоваться позже.

Событие onResponseStarted запускается при получении первого байта ответа. В этом случае у нас есть понимание, что соединение установлено и еще не закрыто, что поможет нам вовремя соединение процессов. Предшествующие события могут содержать этапы перенаправления запроса или аутентификации и более поздние события могут запускаться после того, как соединение уже закрыто. Предоставленные данные содержат идентификатор запроса, который отображает ответ на соответствующий запрос и является причиной сохранения данных запроса в массив, определенный ранее. На этом этапе мы должны верифицировать, в рамках Remote Topology, внутренний IP-адрес и порт сервера, который отправил ответ, то есть является частью полезной информации для соединения. Собранный пакет данных затем отправляется в Miner.

Работа монитора: определим основными пакетами, для обработки следующие:

- TLS ClientHello
- HTT GET or POST request

• T P Fin packet
• T P Rst packet

После определения пакетов для захвата, начинается процесс мониторинга сетевого трафика. HTTP мы опознать, получив HTTP GET или POST запрос, который содержит все данные, необходимые для соединения (включая дополнительные заголовки). В случае TLS мы можем определить начало соединения путем захвата пакета Client Hello, который содержит индикатор HNI, который будет также использоваться для склеивания. Оба протокола совместно используют уровень TCP, поэтому мы можем определить конец передачи путем захвата пакетов TCP Fin или TCP Rst.

В рамках захвата пакетов, нам необходимо отслеживать установленные соединения TLS и HTTP, чтобы корректно интерпретировать соответствующие данные HTTP. Это происходит в виду того, что браузер может повторно использовать подключение для нескольких HTTP-запросов. В случае HTTP/1.1 мы могли бы легко определить при повторном использовании соединения на основе наличия заголовка Connection: keep-alive в заголовках запросов [16], но при работе с HTTP/2 и HTTP/3 ситуация усложняется соединениями, где не только запросы могут повторно использовать уже установленные соединения, но и несколько соединений могут быть созданы к одному и тому же серверу одновременно [17]. Из-за этого мы сможем только создать 1:N соединение, в котором одному HTTP будет назначено несколько подключений, которые были активны во время передачи ответа.

Miner & Extractor: оба компонента совместно используют один и тот же процесс, поскольку они оба имеют доступ к общему хранилищу, где Miner сохраняет результаты процесса соединения, а Extractor предоставит эти данные без маркирования. Доступ к этому хранилищу должен быть взаимоисключающим для обеспечения полной доступности данных для Extractor.

В Miner источниками данных для соединения являются расширение браузера и монитор интерфейса. Прежде начать процедуру соединения, мы должны синхронизировать оба потока данных. При ретроспективном представлении потока данных, требования к синхронизации могут отсутствовать, так как мы могли бы сортировать события на основе временных меток и в соответствии с ними обрабатывать их. Наша система маркирования работает в режиме реального времени, поэтому мы должны гарантировать что мы будем обрабатывать входящие сообщения правильном порядке.

Применяется синхронизация на основе меток времени с буферизацией сообщений от монитора интерфейса. Когда Miner получает сообщение из монитора интерфейса он сохраняет его в очереди для последующей обработки. Когда Miner получает сообщение от расширения браузера, он в начале сравнивает свою временную метку с первым сообщением очереди, затем обрабатывает сообщение с меньшей временной меткой и повторяет это до тех пор, пока сообщение из расширения браузера обрабатывается.

После синхронизации событий мы переходим к соединению. Первый шаг соединения - создание локального пула активных сетевых подключений, который зеркалирует тот же пул, который содержится внутри браузера. Этот пул будет называться активным пулом подключений. Записи пула будут созданы на основе полученных данных от монитора интерфейсов и используются для соединения с HTTP.

Данные внутри пула хранятся в следующей форме: TCP socket (remote server) – > HNI – > TCP socket (local server). Дополнительно хранится сигнатура соединения. Записи в этот пул добавляются при поступлении сообщения HTTP GET/POST или TLS ClientHello. Если Miner получает сообщение от монитора интерфейса, с информацией о присутствии сообщения TCP Fin или TCP Rst, Miner может реагировать двумя способами. В случае сообщения TCP Rst записи с одинаковыми IP-адресами и портами будут удалены. В случае сообщения TCP Fin необходимо контролировать появление этих сообщений (без учета подтверждения) и каждое из них должно исходить со второй стороны подключения.

Если обе стороны посылают сообщение TCP Fin, мы удаляем соединение из активного пула подключений. При получении сообщения от расширения браузера выполняется внутренний поиск, если пул был создан с помощью TCP socket (IP-адрес и порт удаленного сервера) в качестве первого ключа и HNI в качестве второго ключа. Данный поиск предоставит нам список активных соединений, которые могут использоваться для передачи сообщений HTTP. Результат этого поиска сохраняется в списке для доступа Extractor.

Extractor используется для считывания данных из системы маркирования и предоставления внешних процессов. Нам необходимо предоставить все собранные данные для одного определенного сетевого потока, идентифицированного таким образом, чтобы запрос содержал шесть блоков ключевой информации: 0. Signature 1. Source IP address 2. Source port 3. Destination IP address 4. Destination port 5. Activity timestamp. Нулевой блок необходим для идентификации информации. Блоки 1-4 позволяют идентифицировать сам сетевой поток. Блок Activity timestamp отмечает точку активности сетевого потока, что позволит распознать два идентичных потока, имеющих одинаковые параметры 1-4 но не совпадающих по 5.

В случае поступления запроса Extractor будет просматривать список обрабатываемых соединений и сравнить их с запросом. Каждое соединение в этом списке имеет ссылку на HTTP, которая была передана по этому соединению. При этом, может оказаться, что соединение было не единственным для конкретной передачи. Необходимо учитывать остальные связи хоста. Затем, HTTP данные записи, которая соответствует запрошенным параметрам, посылаются как ответ на запрос.

Заключение

Предложенная система маркирования была внедрена и прошла лабораторные испытания, в качестве компонента системы Remote Topology. Было проведено измерение как наличие системы влияет на время отклика компонентов системы Remote Topology. Основной целью ставилось получение информации будет ли критичным воздействие системы при работе Remote Topology. После нескольких измерений времени загрузки Remote Topology, может который использовать HTTP или HTTPS для передачи данных: с запущенной системой маркирования или без нее, было выявлено, что разница в скорости была на 0,7% медленнее для HTTP-трафика и на 4% медленнее для HTTPS-трафика.

Если принять во внимание воздействие сетевой среды для этого измерения, можно сказать, что использование системы маркирования оказывает существенное влияние на время

загрузки web-страниц Remote Topology, что приводит понижению эргономики работы. По результатам работы было принято решение рассмотреть вариант работы каждой системы в независимом шифрованном канале. Работа проекта Remote Topology переведена на HTTP, при этом каждое подключение выполняется в своей VPN сессии, которая обеспечивает безопасность соединения. Внутри VPN соединения выполняется маркирование потоков, что позволяет выполнить быструю и качественную предобработку данных для последующего ML анализа.

Подготовленные таким образом данные могут применяться в рамках системы [18]. Так же нами произведена оценка пригодности предлагаемой системы маркирования. Данный проект рассматривает применение системы маркирования в рамках конкретной задачи, это означает, что мы не можем точно измерить общее воздействие на весь трафик, который может наблюдаться глобально, так как система маркирования зависит от веб-трафика, генерируемого хостом пользователя.

При этом мы можем создавать категории веб-трафика и измерять, сколько сетевых потоков может быть маркировано с использованием предлагаемой системы. Это позволит выявить в режиме реального времени отклонения между эталонной системой, которая уже опознает пользователя и «нежелательными системами», которые пользователь может использовать для обхода системы защиты в Remote Topology.

Литература

1) *Trofimova Y., Fesl J., Moucha A. M.* Performance Analysis of Neural Network Approach for Evaluation of Trust in Ad-hoc Networks // 2021 11th International Conference on Advanced Computer Information Technologies (ACIT). IEEE, 2021. С. 691-695.

2) *Trofimova Y., Tvrdík P.* Enhancing Reactive Ad Hoc Routing Protocols with Trust // Future Internet. 2022. Т. 14. №. 1. С. 28.

3) *Zeng J., Shan S., Chen X.* Facial expression recognition with inconsistently annotated datasets // Proceedings of the European conference on computer vision (ECCV). 2018. С. 222-237.

4) *Minervini M. et al*. Finely-grained annotated datasets for image-based plant phenotyping // Pattern recognition letters. 2016. Т. 81. С. 80-89.

5) *Favorskaya M.N., Jain L.C*. Saliency detection in deep learning era: trends of development // Информационно-управляющие системы. 2019. №3 (100). URL: https://cyberleninka.ru/article/n/saliency-detection-in-deep-learning-era-trends-of-development (дата обращения: 27.10.2022).

6) *Sivanesan A. P., Mathur A., Javaid A. Y.* A google chromium browser extension for detecting XSS attack in html5 based websites // 2018 IEEE International Conference on Electro/Information Technology (EIT). IEEE, 2018. С. 0302-0304.

7) *Haas S., Sommer R., Fischer M.* Zeek-osquery: Host-network correlation for advanced monitoring and intrusion detection // IFIP International Conference on ICT Systems Security and Privacy Protection. Springer, Cham, 2020. С. 248-262.

8) *Wagner P.* DiaSys: A Method and Tool for Non-Intrusive Runtime Diagnosis of Embedded Software : дис. – Technische Universität München, 2019.

9) *Aceto G. et al*. Toward effective mobile encrypted traffic classification through deep learning // Neurocomputing. 2020. Т. 409. С. 306-315.

10) *Гетьман А.И., Маркин Ю.В., Евстропов Е.Ф., Обыденков Д.О.* Обзор задач и методов их решения в области классификации сетевого трафика // Труды ИСП РАН. 2017. №3. URL: https://cyberleninka.ru/article/n/obzor-zadach-i-metodov-ih-resheniya-v-oblasti-klassifikatsii-setevogo-trafika (дата обращения: 27.10.2022).

11) *Уймин А. Г., Морозов И. М*. Сравнительный анализ инструментов непрерывной онлайн-аутентификации и систем обнаружения аномалий для постоянного подтверждения личности пользователя // T-Comm-Телекоммуникации и Транспорт. 2022. Т. 16. №. 5. С. 48-55.

12) *Tropková Z., Hynek K., Čejka T.* Novel HTTPS classifier driven by packet bursts, flows, and machine learning //2021 17th International Conference on Network and Service Management (CNSM). IEEE, 2021. С. 345-349.

13) Статистика использования браузеров в RUнете за январь 2022 года [Электронный ресурс] URL: http://alexvaleev.ru/browserstat/2022/1/ (дата обращения: 27.10.2022)

14) Свидетельство о государственной регистрации программы для ЭВМ № 2021614803 Российская Федерация. Программный модуль-тренажер подготовки к демонстрационному экзамену профессионального мастерства для обучения студентов СПО по специальности "Системное и сетевое администрирование" : № 2021613749 : заявл. 24.03.2021 : опубл. 30.03.2021 / А. Г. Уймин, В. О. Антонов, Д. А. Шерунтаев, М. М. Агафонова ; заявитель Федеральное государственное бюджетное образовательное учреждение высшего образования «Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых». – EDN CNHXZQ.

15) Window: load event [Электронный ресурс] URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event (дата обращения: 27.10.2022)

16) *Bressoud T., White D.* The HyperText Transfer Protocol //Introduction to Data Systems. Springer, Cham, 2020. С. 609-648.

17) *Velazquez L. M*. HTTP/2 vs HTTP/3: Toolsets for Measuring Tradeoffs. 2021.

18) *Уймин А. Г*. Интеллектуальный анализ динамики трехпозиционного графического манипулятора типа "мышь" как элемента поведенческой биометрии // Системы управления и информационные технологии. 2022. № 2(88). С. 92-96. DOI 10.36622/VSTU.2022.88.2.018. EDN XGHBWO.

AUTOMATIC MARKING OF BROWSER NETWORK TRAFFIC FOR ANALYSIS AND CLASSIFICATION USING THE EXAMPLE OF THE REMOTETOPOLOGY PLATFORM

Anton G. Uymin, *University of the Gubkin Russian State University of Oil and Gas (NRU), Moscow, Russia*
au-mail@ya.ru

Abstract

Purpose: Network traffic analysis is a security standard in most remote work systems. This task is becoming more complex every day due to the use of encryption systems on an increasing number of devices. The traditional method of traffic research in the OSI model does not provide the proper level of reliability. Method: Machine learning is one of the promising methods for solving the problem of network traffic analysis. In the first place is the identification of unwanted traffic, which can be interpreted as a tool for bypassing the system. In order for the system to be used in the educational and production process, it is necessary to take into account the issue of ergonomics of the solution. The influence of the monitoring and analysis system on the response rate. The purpose of the work: The purpose of the work is: the formation of tagged HTTP/HTTPS traffic data, to intercept encrypted tunnel traffic. It is proposed to develop a browser plugin whose tasks are to collect data from the user's browser using the Chromium API. The collected data is combined with network streams to form labeled data used later for ML model training datasets. Novelty: The novelty elements of the presented solution are the use of Machine learning to classify network traffic based on labeled data. Result: using the presented solution for automatic marking of browser network traffic for analysis and classification, we decided to consider the operation of each system in an independent encrypted channel. The launch of the RemoteTopology project has been switched to HTTP, while each connection is performed in its own VPN session and guarantees the security of the connection. Within the connection, the marking of streams is performed, which allows you to quickly and efficiently process data for subsequent ML analysis. Practical relevance: the presented solution is proposed to be implemented within the framework of the RemoteTopology platform to identify in real time deviations between the reference system that already identifies the user and "undesirable systems" that the user can use to bypass the protection system.

Keywords: *data, connection, http, network traffic, analysis, ML model.*

References

1. Trofimova Y., Fesl J., Moucha A.M. (2021). Performance Analysis of Neural Network Approach for Evaluation of Trust in Ad-hoc Networks. *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)*. IEEE, pp. 691-695.
2. Trofimova Y., Tvrdik P. (2022). Enhancing Reactive Ad Hoc Routing Protocols with Trust *Future Internet*. Vol. 14. No. 1. P. 28.
3. Zeng J., Shan S., Chen X. (2018). Facial expression recognition with inconsistently annotated datasets. *Proceedings of the European conference on computer vision (ECCV)*, pp. 222-237.
4. Minervini M. et al. (2016). Finely-grained annotated datasets for image-based plant phenotyping. *Pattern recognition letters*. Vol. 81, pp. 80-89.
5. Favorskaya M.N., Jain L.C. (2019). Saliency detection in deep learning era: trends of development. *Information and control systems*. No.3 (100). URL: https://cyberleninka.ru/article/n/saliency-detection-in-deep-learning-era-trends-of-development (date of access: 27.10.2022).
6. Sivanesan A.P., Mathur A., Javaid A.Y. (2018). A google chromium browser extension for detecting XSS attack in html5 based websites. *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, pp. 0302-0304.
7. Haas S., Sommer R., Fischer M. (2020). Zeek-osquery: Host-network correlation for advanced monitoring and intrusion detection. *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, Cham, pp. 248-262.
8. Wagner P. (2019). DiaSys: A Method and Tool for Non-Intrusive Runtime Diagnosis of Embedded Software : diss. Technische Universitat Munchen.
9. Aceto G. (2020). et al. Toward effective mobile encrypted traffic classification through deep learning. *Neurocomputing*. Vol. 409, pp. 306-315.
10. Get'man A.I., Markin Yu.V., Evstropov E.F., Obydenkov D.O. (2017). Review of problems and methods of their solution in the field of network traffic classification. Proceedings of JNP RAN. No.3. URL: https://cyberleninka.ru/article/n/obzor-zadach-i-metodov-ih-resheniya-v-oblasti-klassifikatsii-setevogo-trafika (date of access: 27.10.2022).
11. Uymin A.G., Morozov I.M. (2022). Comparative analysis of continuous online authentication tools and anomaly detection systems to permanently confirm user identity. T-Comm. Vol. 16. No. 5, pp. 48-55.
12. Tropkova Z., Hynek K., Cejka T. (2021). Novel HTTPS classifier driven by packet bursts, flows, and machine learning. *2021 17th International Conference on Network and Service Management (CNSM)*. IEEE, pp. 345-349.
13. Statistics on the use of browsers in RuNet for January 2022. URL: http://alexvaleev.ru/browserstat/2022/1/ (date of access: 27.10.2022).
14. Certificate of state registration of the computer program No. 2021614803 Russian Federation. Software module-simulator of preparation for a demonstration exam of professional skills for teaching students of secondary vocational education in the specialty "System and Network Administration": No. 2021613749: Appl. 03/24/2021 : publ. 30.03.2021 / A. G. Uymin, V. O. Antonov, D. A. Sheruntaev, M. M. Agafonova; applicant Federal State Budgetary Educational Institution of Higher Education "Vladimir State University named after Alexander Grigorievich and Nikolai Grigorievich Stoletovs".
15. Window: load event. URL: https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event (date of access: 27.10.2022).
16. Bressoud T., White D. (2020). The HyperText Transfer Protocol. *Introduction to Data Systems*. Springer, Cham, pp. 609-648.
17. Velazquez L.M. (2021). HTTP/2 vs HTTP/3: Toolsets for Measuring Tradeoffs.
18. Uymin A.G. (2022). Intellectual analysis of the dynamics of a three-position graphic manipulator of the "mouse" type as an element of behavioral biometrics. *Control systems and information technologies*. No. 2(88), pp. 92-96. DOI 10.36622/VSTU.2022.88.2.018.

Information about author:
Anton G. Uymin, S*enior Lecturer of the Department of Information Technology Security, Faculty of Integrated Security of Fuel and Energy Complex, University of the Gubkin Russian State University of Oil and Gas (NRU), Moscow, Russia*