УДК 004.85

МОДУЛЬ РАСПОЗНАВАНИЯ СЕТЕВОЙ РАЗВЕДКИ

Н.П. ШАРАЕВ, С.Н. ПЕТРОВ

Белорусский государственный университет информатики и радиоэлектроники, Республика Беларусь

Поступила в редакцию 18 ноября 2021

Аннотация. Изучена эффективность метода дерева решений в задачах обнаружения сетевой разведки. Приведены блок-схемы обученного дерева решений и работы модуля обнаружения признаков сетевой разведки.

Ключевые слова: сетевая разведка, аномалии сетевого трафика, метрики признаков сетевой разведки, датасеты.

Введение

В последнее время наблюдается тенденция перехода от массовых кибератак отдельных злоумышленников к масштабным атакам киберпреступных группировок на конкретные организации (таргетированные атаки). Данные атаки в значительной мере опасны для организаций, в связи с созданием злоумышленниками вредоносного программного обеспечения с учетом специфики сетевой инфраструктуры организации.

В мировой практике задача обнаружения признаков сетевой разведки сводится к задаче обнаружения аномалий сетевого трафика, так как в обоих случаях наблюдается большое количество подозрительного трафика, преимущественно на третьем и четвертом уровнях модели OSI. Существуют датасеты для обучения моделей обнаружению аномалий сетевого трафика: Network Intrusion Detection [1]; UNSW_NB15 [2]; 2019 Trendmicro CTF Wildcard 400 [3]; Kitsune Network Attack [4]; NSL-KDD [5]; Network Traffic with Port Scanning Attack [6]. Путем анализа открытых источников были выбраны наиболее актуальные метрики и разработан датасет для обнаружения сетевой разведки [7].

Таким образом, актуальным является вопрос разработки модуля обнаружения признаков сетевой разведки и оценка эффективности его работы на созданном датасете.

Метод дерева принятия решений в решении задач обнаружения сетевой разведки

Созданный JSON-датасет конвертируется в двумерную матрицу признаков. На данном этапе множество ответов представлено в формате строки («good», «bad») и для дальнейшей работы требуется перевести их в бинарный формат (0, 1). Указанное реализуется использованием класса LabelEncoder и перекодирует ответы следующим образом: «good» – 1, «bad» – 0. Матрица признаков, содержащая перекодированные ответы, с помощью класса train_test_split разделяется на обучающее (80 %, или 800 событий) и тестовое (20 %, или 200 событий) множества с помощью функции псевдослучайных чисел. Анализ созданных множеств показал наличие 201 события сетевой разведки в обучающей выборке (25 %) и 49 событий в тестовом множестве (25 %), что позволяет судить о сбалансированности выборок.

В результате исследования эффективности методов машинного обучения [8], наиболее быстрым методом, верно классифицирующим признаки сетевой разведки, признан метод дерева принятия решения (Decision Tree) с параметрами criterion = «gini» и splitter = «random», обучающимся за 0,000912 секунды.

Блок-схема обученного алгоритма дерева принятия решений представлена на рис. 1.

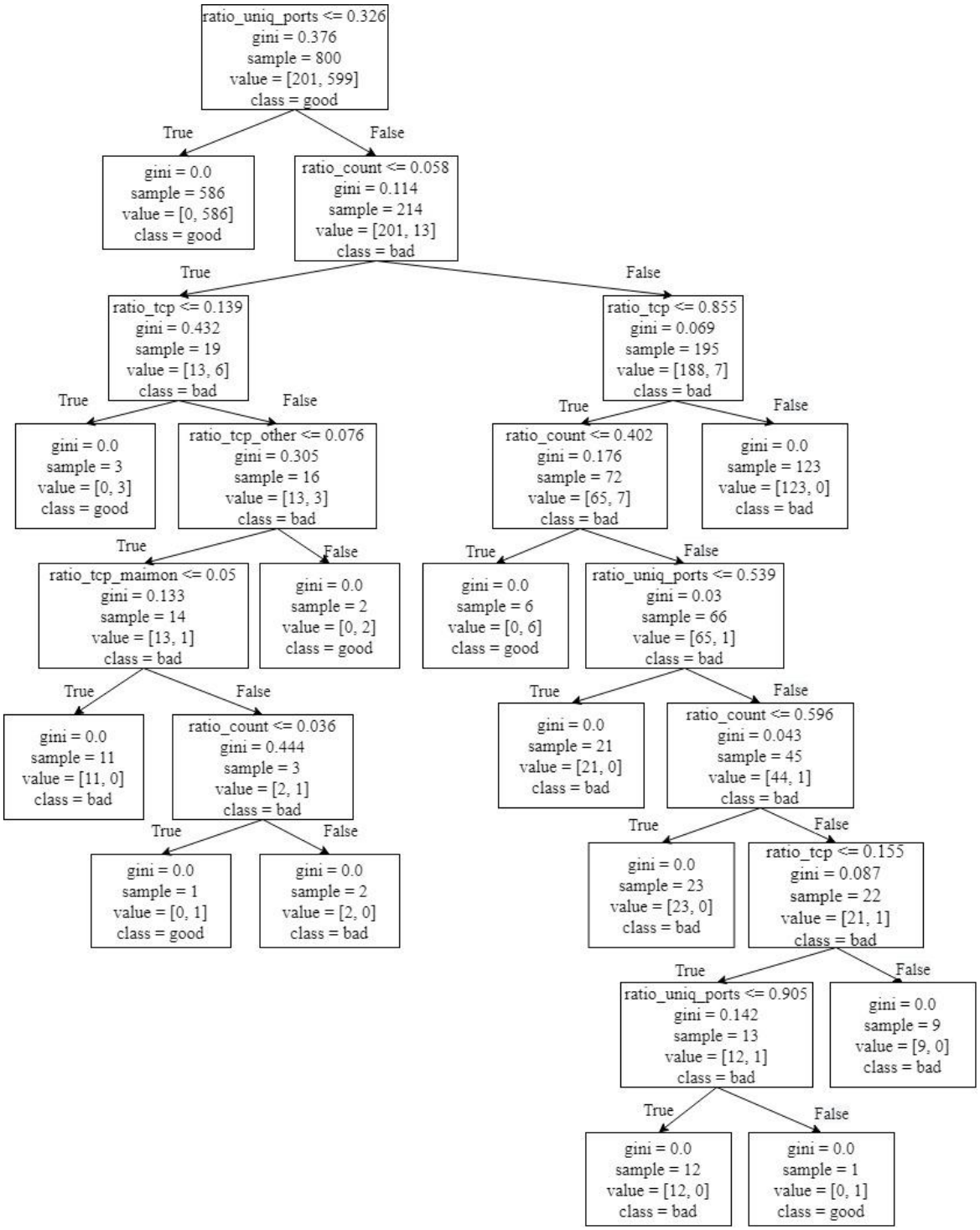

Рис. 1. Блок-схема обученного дерева принятия решений

Дополнительным плюсом применения указанного метода на практике является возможность описать его работу программным языком вместо применения обученной модели, что положительно скажется на производительности и объеме программного кода.

Принцип работы модуля обнаружения признаков сетевой разведки

При запуске модуль обнаружения признаков сетевой разведки импортирует базовые и расширенные библиотеки для работы программы. При разделении программы на множество файлов создавался один основной файл (main.py), в котором инициализировались библиотеки threading, json и logging. Далее импортируются основные модули: Sniffer, DataStore, Analyitcs и

Model. Модули представляют собой отдельные интерпретируемые файлы (*.py), содержащие в себе программный код и предназначенные для логического разделения программы. После создается объект библиотеки logging, предназначенный для создания журнала событий, определяется формат записи события, путь к лог-файлу, а также уровень информирования в журнале событий. Далее загружается файл конфигурации. Если данный файл отсутствует, происходит преднамеренное завершение программы и информирование об отсутствии файла. Из файла конфигурации импортируются следующие параметры: количество событий в анализируемой выборке (по умолчанию 30 событий), время жизни одного события (по умолчанию 5 мин), переключатель создания модели (по умолчанию модель создается), путь к датасету (по умолчанию «module/data/dataset.json»), путь к модели (по умолчанию «module/data/model.pkl»), анализируемый интерфейс (по умолчанию «eth0»), массив исключенных из мониторинга IP-адресов (по умолчанию только loopback). На основе представленных параметров создаются объекты DataStore(), Sniffer(), Analytics() и Model(). При этом объекты Sniffer() и Analytics() создаются в новых потоках библиотеки threading (sniffer и analytics соответственно) и выполняются параллельно основному потоку ввода-вывода Python, а объект Model() передается объекту Analytics(). Потоки запускаются и проверяются на наличие ошибок при выполнении. В случае наличия ошибок основной поток ввода-вывода досрочно завершается и останавливает созданные потоки библиотеки threading.

Внутри потока sniffer после записи параметров конфигурации в поля класса происходит выполнение двух команд операционной системы для получения списка прослушиваемых портов и IP-адресов интерфейса. После получения указанной информации создается и запускается объект асинхронного сниффера класса scapy. При получении пакета данный сниффер проверяет, является ли данный пакет из стека TCP/IP, направлен ли он на закрытые порты транспортного уровня и не находится ли IP-адрес, на который он отправлен в списке исключение. Если на все вопросы ответ «правда» (true), то пакет нормализуется и записывается (метод запись события) в созданный объект DataStore(). Если нет – пакет не обрабатывается и считается легитимным. Дополнительно в классе есть методы запуска и остановки асинхронного снифера, что связано с отладкой приложения.

В классе DataStore() реализованы два метода: запись события и чтение всех событий. При записи события происходит извлечение из тела события IP-адреса источника. Базируясь на данной информации создается новый словарь, содержащий в себе в качестве ключа IP-адрес источника, а в качестве значения – словарь, состоящий из времени получения первого пакета с указанным IP и массива событий. В случае, если IP-адрес нового пакета уже существует в словаре, событие заносится в массив событий. Метод чтения всех событий возвращает словарь всех событий, хранящихся в поле объекта DataStore().

Внутри потока analytics происходит чтение событий из объекта DataStore(), инициализация объекта Model() и запуск метода проверки события. Метод проверки события запущен в бесконечном цикле и для этого и запущен в отдельном потоке. В данном цикле проверяется наличие в базе данных не более 100 различных исходящих IP-адресов. Это применяется для защиты от DDoS атак, в целях защиты от неограниченного потребления оперативной памяти. Если количество исходящих IP-адресов превышает параметр, происходит очистка базы данных объекта DataStore(). Далее для каждого исходящего IP-адреса формируется выборка из 30 событий (настраивается в конфигурационном файле). В случае, если данное число событий за 5 минут (настраивается в конфигурационном файле) не набралось, создается множество из не менее чем половины исходного параметра (в данном случае 15 событий). Если же и половины выборки не набралось – выборка удаляется. Представленный механизм экономит оперативную память и позволит выявлять сетевую разведку, проводимую за значительный промежуток времени. Выбор параметра в 5 минут основан на желании злоумышленника провести анализ запущенных в информационной системе служб в обозримый промежуток времени. Так, для набора минимального количества событий для анализа достаточно отправлять одно событие в 20 секунд, что для анализа 1000 популярных портов составляет 5,5 часа, на что вполне может пойти злоумышленник. После, базируясь на сформированной выборке, происходит расчет метрик и создается полностью нормализованное событие, предназначенное для анализа методами машинного обучения. Данное событие передается объекту Model(). В случае, если полученный результат анализа относится к классу сетевой разведки, данное событие логируется и содержит

время получения первого пакета, исходящий IP-адрес и все параметры нормализованного события. Таким образом, можно назначить автоматическую блокировку подозрительно IP-адреса на 5 минут с помощью утилиты fail2ban.

Блок-схема работы модуля обнаружения признаков сетевой разведки представлен на рис. 2.

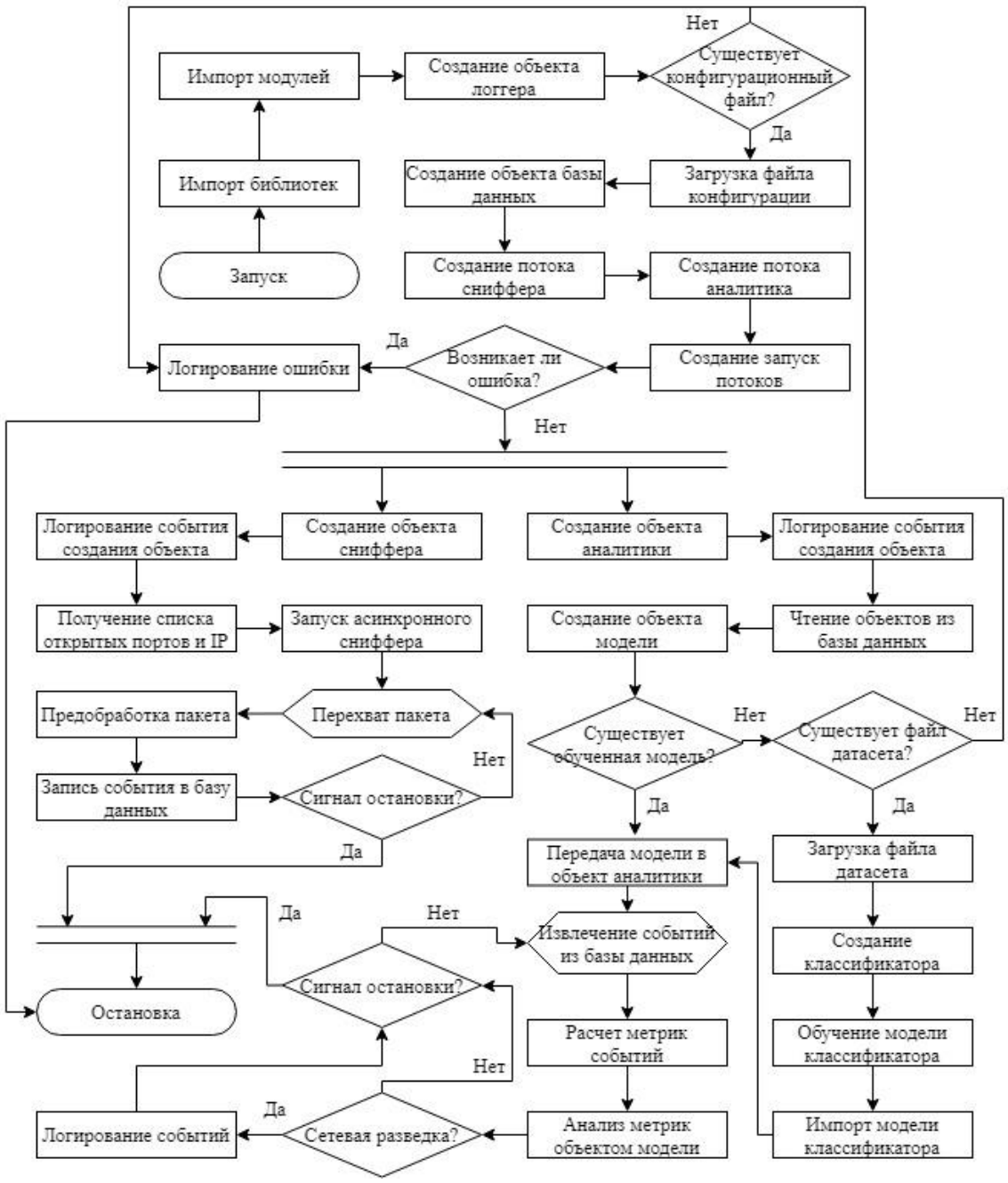

Рис. 2. Блок-схема работы модуля обнаружения признаков сетевой разведки

На момент создания объекта Model() происходит поиск существующей обученной модели машинного обучения. При ее наличии происходит загрузка в оперативную память. В случае ее отсутствия происходит создание новой модели, если этому не противоречит параметр конфигурационного файла (create_model). Когда параметр «create_model» переведен в состояние «false» или отсутствует файл датасета, выполнение программы завершается, журналируется событие ошибки и останавливаются все потоки. При создании новой модели происходит загрузка файла датасета, создание необходимого классификатора (в данном случае Decision Tree с параметрами criterion = «gini» и splitter = «random»), обучение классификатора и экспорт созданной модели классификатора для ускорения последующих загрузок программы. При

передаче в указанный объект события происходит его анализ классификатором и возврат результата проверки в объект Analytics().

Описанное программное обеспечение позволяет добиться значительной скорости работы в потоке, небольшого потребления оперативной и энергонезависимой памяти, качественного обнаружения признаков сетевой разведки и возможность интегрирования базовыми сторонними утилитами (типа fail2ban).

Заключение

Разработана блок-схема работы модуля обнаружения признаков сетевой разведки. Описаны необходимые компоненты и зависимости модуля. Проведен анализ работы модуля, показавший 100 % обнаружения попыток проведения сетевой разведки. Наилучшие результаты показал метод дерева принятия решений с параметрами criterion = «gini» и splitter = «random», с точностью 100 % и скоростью работы 0,912 мс. Дополнительно проведено представление алгоритма с наилучшими параметрами в виде программного кода, что позволило увеличить скорость работы приблизительно в 2 раза.

NETWORK INTELLIGENCE RECOGNITION MODULE

N.P. SHARAEV, S.N. PETROV

Abstract. The effectiveness of the decision tree method in network intelligence detection tasks has been studied. The flowcharts of the trained decision tree and the operation of the network intelligence feature detection module are given.

Keywords: network intelligence, network traffic anomalies, metrics of network intelligence features, datasets.

Список литературы

1. Network Intrusion Detection – Kaggle. [Electronic resource]. URL: https://www.kaggle.com/sampadab17/network-intrusion-detection/.
2. UNSW_NB15 – Kaggle. [Electronic resource]. URL: https://www.kaggle.com/mrwellsdavid/unsw-nb15/.
3. 2019 Trendmicro CTF Wildcard 400 – Kaggle. [Electronic resource]. URL: Режим доступа: https://www.kaggle.com/hawkcurry/2019-trendmicro-ctf-wildcard-400/.
4. Kitsune Network Attack Dataset – Kaggle. [Electronic resource]. URL: https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune/.
5. NSL-KDD dataset – Canadian Institute for Cybersecurity. [Electronic resource]. URL: https://www.unb.ca/cic/datasets/ nsl.html.
6. NTwPSA – Loughborough University Network Traffic with Port Scanning Attack. [Electronic resource]. URL: https://figshare.com/articles/dataset/Loughborough_University_-_Network_Traffic_with_Port_Scanning_Attack/4630282/3
7. Шараев Н.П., Петров С.Н. // Управление информационными ресурсами: материалы XVII Междунар. науч.-практ. конф., 12 мар. 2021 года. Минск. 2021. С. 238–240.
8. Sharaev N. // Education & applications on design and engineering during pandemic 2021: 7th Maltepe University International Student Congress (MUISC), 5–7 May 2021, Istanbul, Turkey, 2021. P. 47.