DOI: 10.15514/ISPRAS-2026-38(4)-22

Определение эффективности обнаружения DNS-туннелей при помощи нейронной сети в системе обнаружения вторжений

[1] *Н.Д. Маринин, ORCID: 0009-0001-1589-2772 <Marinin.N.D@hse.ru>*
[1,2,3,4] *А.И. Гетьман, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

[1] *Национальный исследовательский университет «Высшая школа экономики», Россия, 101978, Мясницая ул., д. 20.*

[2] *Институт системного программирования им. В.П. Иванникова РАН, Россия, 109004, г. Москва, ул. Александра Солженицына, д. 25.*

[3] *НИУ Московский Физико-Технический институт, Россия, 141701, Московская область, г. Долгопрудный, Институтский пер., д. 9.*

[4] *Московский государственный университет им. М.В. Ломоносова, Россия, 119991, г. Москва, ГСП-1, Ленинские горы, д. 1.*

Аннотация. В работе рассматривается метод обнаружения туннелей, реализованных с помощью протокола DNS в сетевом трафике при помощи нейронной сети. Для этого был произведен анализ актуальных методов. Подготовлен набор данных для обучения нейронной сети. Предложенная модель использует на входе последовательность символов, извлеченная из DNS ответа. Обученная модель показала F1-меру близкую к единице. Для проверки работоспособности доработаны модули системы обнаружения вторжений с открытым исходным кодом Snort3. Обученная модель исполнялась при помощи совместимого модуля LibML. Результаты эксперимента показывают точность близкой к единице и практически полное отсутствие ложных срабатываний. Время обработки DNS пакета с активированным модулем обнаружения при помощи нейронной сети в среднем увеличилось на 13%, а для смешанного трафика, состоящего из различных протоколов время увеличилось на 2%. Анализ экспериментальных данных подтверждает, что использование нейронной сети эффективно дополняет классические средства безопасности, позволяя эффективно обнаруживать скрытые каналы, инкапсулированные в DNS, не оказывая существенного влияния на производительность сигнатурной подсистемы.

Ключевые слова: DNS-туннелирование; нейронная сеть; система обнаружения вторжений (IDS); скрытые каналы.

Для цитирования: Маринин Н.Д., Гетьман А.И. Определение эффективности обнаружения DNS-туннелей при помощи нейронной сети в системе обнаружения вторжений. Труды ИСП РАН, том 38, вып. 4, часть 2, 2026 г., стр. 123–142. DOI: 10.15514/ISPRAS–2026–38(4)–22.

Evaluating the effectiveness of DNS tunnel detection using a neural network in an intrusion detection system

[1] *N.D. Marinin, ORCID: 0009-0001-1589-2772 <Marinin.N.D@hse.ru>*
[1,2,3,4] *A.I. Getman, ORCID: 0000-0002-6562-9008 <ever@ispras.ru>*

[1] *National Research University «Higher School of Economics», 20, Myasnitskaya ulitsa, Moscow, 101000, Russia.*

[2] *Ivannikov Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.*

[3] *Moscow Institute of Physics and Technology (National Research University), 9 Institutskiy per., Dolgoprudny, Moscow Region, 141701, Russia.*

[4] *Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russia.*

Abstract. This paper investigates a method for detecting DNS tunnels in network traffic using a neural network. To achieve this, an analysis of current detection approaches was conducted, and a dataset for neural network training was prepared. The proposed model takes as input a sequence of characters extracted from DNS responses. The trained model demonstrated an F1-score close to one. To validate the proposed approach, modules of the open-source intrusion detection system Snort 3 were extended and modified. The trained model was executed using the compatible LibML module. Experimental results demonstrate detection accuracy close to one with an almost complete absence of false positives. The average DNS packet processing time with the neural network-based detection module enabled increased by 13%, while for mixed traffic consisting of multiple protocols, the processing time increased by only 2%. The analysis of the experimental data confirms that the use of neural networks effectively complements traditional security mechanisms, enabling efficient detection of covert channels encapsulated within DNS traffic without significantly affecting the performance of the signature-based detection engine.

Keywords: DNS tunneling; neural network; intrusion detection system (IDS); covert channels.

For citation: Marinin N.D., Getman A.I. Evaluating the effectiveness of DNS tunnel detection using a neural network in an intrusion detection system. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 4, part 2, 2026, pp. 123-142 (in Russian). DOI: 10.15514/ISPRAS-2026-38(4)-22.

1. Введение

Обеспечение информационной безопасности сетей является одной из ключевых задач современных информационных систем. Важным и сложным является обнаружение скрытых каналов передачи данных, позволяющие злоумышленникам обходить средства защиты и организовывать управление вредоносным программным обеспечением. Одним из распространенных примеров использования таких каналов является DNS-туннелирование – передача данных посредством протокола DNS. Пакеты протокола DNS являются критически важной службой в обеспечении работы сети. Пакеты DNS используются пользователями для разрешения имен в сети интернет, как правило, свободно пропускаются между сетями, поскольку DNS-трафик необходим для нормальной работы сети [1]. Утилиты создающие туннели с использованием протокола DNS, инкапсулируют полезную нагрузку в полях запроса/ответа, что делает трафик внешне похожим на легитимные DNS-запросы. Злоумышленники часто используют туннели для создания скрытого канала (например, для обмена данными или выполнения команд управления зараженным устройством) сквозь межсетевые экраны и прокси-серверы [2-3].

Практическое использование и опасность DNS-туннелей подтверждается реальными инцидентами информационной безопасности. Например, вредоносное программное обеспечение FrameworkPOS [4] и др. использовали DNS-туннели для кражи платёжных данных. В случае атаки на компанию Home Depot, вредоносный модуль отправлял данные на

серверы злоумышленников через DNS, что привело к компрометации десятков миллионов записей банковских карт [5-6]. Аналогичные техники применяются и в современных APT-кампаниях (Advanced Persistent Threat – персистентная устойчивая угроза, то есть спланированная многоэтапная кибератака), что делает DNS-туннелирование устойчивым вектором атак.

Классическим методом обнаружения DNS-туннелей является поиск по специфическим шаблонам трафика и статистический анализ содержимого (длины домена, энтропия, частоты запросов и тому подобное). Использование статического анализа даёт неплохие результаты (например, анализ частоты биграмм в доменных именах показал точность (Precision) ~98.7%), но его легко обойти [7]. Однако все статистические детекторы имеют ограниченную универсальность и склонны к ложным срабатываниям при сложном трафике [8].

Важным ограничением является использование клиентами технологии DoH (DNS over HTTPS). Такой трафик использует протокол TLS (Transport Layer Security) для обеспечения безопасности, в то время как традиционные DNS пакеты передаются в открытом виде. Чтобы проанализировать содержимое зашифрованного трафика необходимо выполнить перехват и расшифровку (атака "человек-посередине", Man-in-the-Middle – MiTM) или опираться на поведенческие и статистические данные [9].

Современные подходы основаны на машинном и глубоком обучении. Чаще всего используют модели на основе опорных векторов (Support Vector Machine, SVM), случайных лесов, рекуррентных и сверточных нейронных сетей (Convolutional Neural Networks, CNN), а также гибридные архитектуры. Важным в применении таких подходов является выделение признаков. Признаки могут быть получены из самих пакетов (например, значение поля, характеристика доменов и т.д.) или собраны из статистики многих пакетов (средний размер, количество, частота запросов). Признаки на основе извлеченных данных из пакетов позволяют обнаружить скрытый канал сразу в то время, как статистические и поведенческие признаки покажут скрытый канал с задержкой и часть данных может быть уже передана злоумышленникам. Модель DNS Sentinel использует ансамбль алгоритмов, который показал полноту (Recall) обнаружения DNS-туннелей, равную 1.00, и F1-меру ≈ 0.83, используя признаки, извлеченные из полей пакетов [8]. Метод случайного леса в гибриде с усиленным алгоритмом оптимизации достиг точности 99.82 % и показателя F1 = 99.79 % на зашифрованном наборе данных DoH при использовании поведенческих признаков трафика [9]. Анализ признаков на основе статистических и поведенческих данных позволяет обнаружить скрытый канал даже на зашифрованном трафике, однако допускает задержку обнаружения, обусловленную временем достижения порога срабатывания. Некоторые исследователи используют комбинированные архитектуры на основе CNN. В частности, авторы работы [10] интегрировали выход CNN с деревом решений, а также исследовали комбинацию CNN и SVM, при использовании которой была достигнута точность классификации более 99%. Авторы работы [11] использовали генетический алгоритм для отбора признаков построенных на энтропии данных внутри пакета DNS и классификатор SVM, что дало значение F1-меры≈0.946.

2. Обзор существующих методов обнаружения

Методы обнаружения DNS-туннелирования условно можно разделить на три основные группы: сигнатурные, статистические и модели машинного/глубокого обучения.

2.1 Сигнатурные методы

Традиционные средства обнаружения (Intrusion Detection System / Intrusion Prevention System, IDS/IPS) используют сигнатурный анализ. Такие методы выявляют заранее известные шаблоны трафика: характерные строки в поддоменах, специфические типы DNS-записей (текстовые – TXT, пустые – NULL), аномальную длину доменного имени или нестандартные

последовательности символов, характерные для конкретных утилит, создающие скрытый канал.

Для такого метода существенными ограничениями являются:

- Низкая устойчивость к модификации туннеля. Например, смена схемы представления данных с Base32 на Base64, добавление дополнительных этапов или случайных символов.
- Пороговые значения требуют калибровку под конкретную инфраструктуру сети.
- Возможные ложные срабатывания из – за доменов, созданные CDN или системами телеметрии.
- Необходимость открытого трафика для анализа пакета.

Таким образом, сигнатурные методы демонстрируют высокую эффективность лишь для известных реализаций туннелей и в контролируемых условиях.

2.2 Методы машинного обучения с ручной инженерией признаков

Более современные подходы используют машинное обучение. Бубнов и др. [12] обучили полносвязную нейронную сеть на наборе признаков (длина имени, энтропия, тип записи и пр.) и получили 83.5% правильных классификаций (Accuracy), при этом точность (Precision) составила 84.1%, а полнота (Recall) 81.8% [12]. Другие работы применяют гибридные алгоритмы: генетические алгоритмы вкупе с SVM достигли значения F1-меры ≈ 0.946, а простые свёрточные сети выявляют туннели с точностью свыше 92% при очень низком уровне ложных срабатываний [11].

Выявленные недостатки:

- зависимость модели от качества выбора признаков;
- необходимость дополнительного этапа выделения и подсчета признаков;
- система зависима от экспертных знаний и не гарантирует устойчивость к новым атакам.

2.3 Методы глубокого обучения

В последние годы активно применяются архитектуры глубокого обучения: CNN, рекуррентные нейронные сети (Recurrent Neural Network, RNN) и их частные случаи LSTM (Long Short-Term Memory) и GRU (Gated Recurrent Unit), а также их комбинации с векторным представлением (эмбеддинг, embedding) строк из полей пакетов.

Подход основан на последовательной обработке строки доменного имени: слой векторного представления преобразует каждый символ в вектор, а LSTM-модель выявляет зависимости между символами и характерные последовательные шаблоны, свойственные DNS-туннелированию. Это даёт модели возможность самостоятельно учить релевантные признаки из данных, а не опираться на заранее выбранные метрики. Аналогичные архитектуры, сочетающие слой векторного представления и LSTM-слой подтвердили высокую эффективность для анализа сетевого трафика: например, Авторы работы [13] построили IDS на таких сетях и получили 99.7% точности на общих наборах данных.

Недостатки системы с моделью глубокого обучения:

- высокая ресурсоемкость;
- высокая зависимость от обучающей выборки;

2.4 Выводы по разделу

Сравнение и анализ существующих методов показывает, что сигнатурные и статистические методы возможно обойти. Классические алгоритмы машинного обучения имеют ограничения, вносимые ручной инженерией признаков. Модели глубокого обучения

показывают высокую точность, но требуют значимых ресурсов. Следовательно, остается актуальным вопрос поиска эффективного метода обнаружения DNS туннелей с точки зрения полноты, точности и затрачиваемых ресурсов (времени выполнения).

Важным является проверка, тестирование возможности применения модели в модуле, средства обнаружения вторжения, работающем на трафике. Исследования, упомянутые выше, не рассматривают контекст реального внедрения в процесс выявления вторжений IDS и анализа влияния на производительность и качество обнаружения.

Таким образом, несмотря на значительные исследования в области обнаружения DNS-туннелей, как классическими способами, так и методами машинного и глубокого обучения (Machine Learning / Deep Learning, ML/DL), сохраняется проблема создания средства автоматического выявления скрытых каналов протокола DNS с высокой точностью, минимальным числом ложных событий и приемлемыми вычислительными затратами в условиях реального сетевого трафика.

3. Подготовка данных

В основе набора данных для обучения модели лежит публичный набор записей DNS-трафика, включающий в себя туннели и легитимные запросы [14-15]. Легитимные запросы-ответы генерировались по списку популярных доменов (OpenDNS Top Domains). Туннельный трафик создавался с использованием нескольких утилит: Iodine [16], dns2tcp [17], dnscapy [18] и tuns [19]. Для последующих экспериментов было создано два файла формата PCAP, путем объединения мелких файлов, один содержащий только туннели, другой – легитимную DNS активность.

Из записей трафика был создан набор данных, путем извлечения данных из DNS ответов. Набор данных состоит из типа DNS-записи, метки класса и нормализованного (сырые байты были представлены в шестнадцатеричном виде) значения поля записи. Нормализация необходима из-за того, что извлекаемые значения полей имеют разные типы данных. Итого, выделено записей с туннелями – 179404, легитимных – 974444. Извлеченные данные были сбалансированы, путем случайной выборки заданного количества образцов, обеспечивая равное количество записей для классов. Итоговым результатом является набор данных, содержащий по 150 000 записей каждого класса. Итоговое распределение типов записей показано на рис. 1, рис. 2 отражает кумулятивное распределение длин доменов.

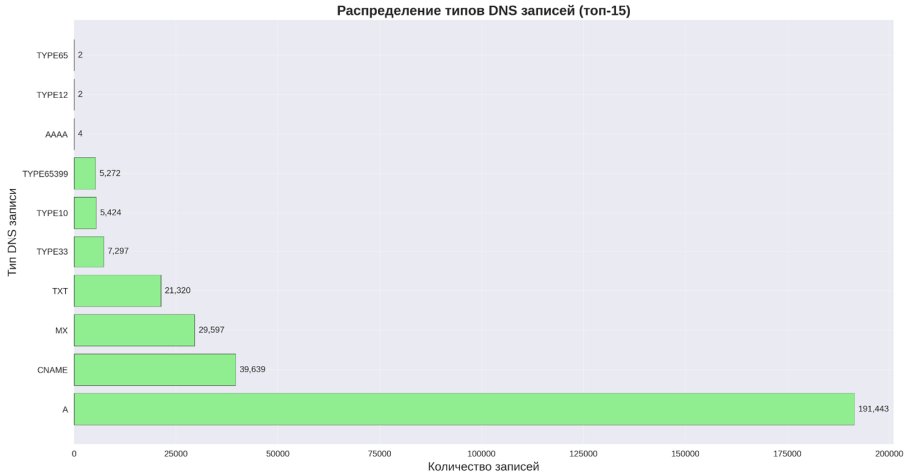

Рис. 1. Распределение типов DNS записей.
Fig. 1. Distribution of DNS record types.

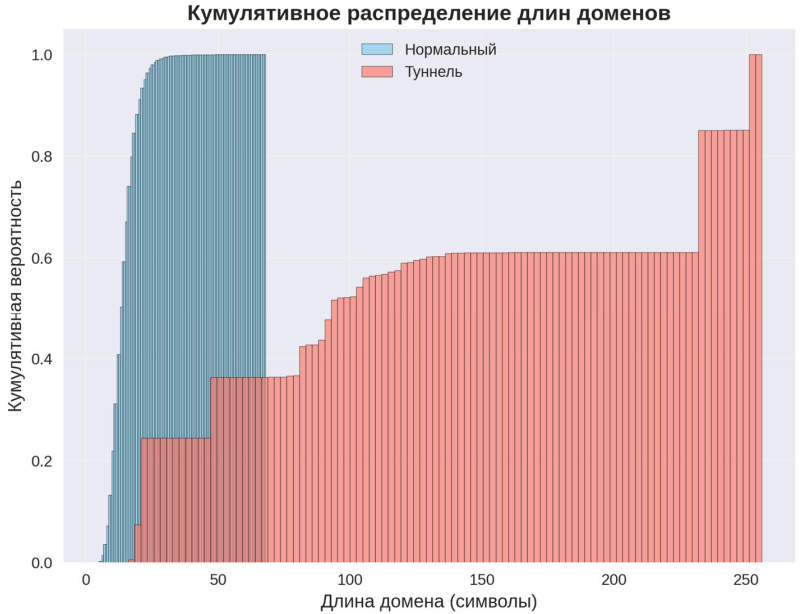

Рис. 2. Кумулятивное распределение длин доменов.
Fig. 2. Cumulative distribution of domain lengths.

Анализ доменных имен показал, что большое подмножество доменные имен, используемые в туннеле, содержали одинаковые корневые домены, поэтому была применена аугментация данных путем замены корневого домена на случайный, без изменения доменов следующих уровней, что характерно для DNS-туннелей. Такая обработка данных служит для предотвращения утечки данных из выборки.

Обучение производилось на процессоре Intel Core i7-12700h, использовалась подготовленная выше выборка из 300 тысяч записей, по 150 тысяч случайно выбранных строк каждого класса. Время, затраченное на обучение модели, не превышало пяти минут. Набор данных был разделен на обучающий и валидационный: в пропорции 80/20, случайным образом.

Дополнительно был сгенерирован независимый набор данных для проверки устойчивости модели к новым типам атак. Для проведения экспериментов был разработан программный модуль генерации синтетических сетевых пакетов и их записи в файлы в формате PCAP, моделирующий как легитимный DNS-трафик, так и различные реализации DNS-туннелей. Основной целью генератора являлось получение статистически контролируемого набора данных, в котором различия между классами обусловлены исключительно особенностями реализации туннелей, а не случайными параметрами сетевого взаимодействия.

Набор данных состоит из двух основных классов: легитимный DNS-трафик, DNS-туннели. Каждый класс представлен набором независимых сценариев, моделирующих различные варианты поведения клиентов и серверов. Для каждого сценария автоматически формируется отдельный PCAP-файл, содержащий полный набор трафика, полный набор туннелей и общий набор данных для последующей обработки средствами машинного обучения.

Одной из особенностей генератора является моделирование широкого спектра различных реализаций DNS-туннелей. Вместо воспроизведения поведения одной конкретной утилиты реализованы многочисленные варианты, имитирующие особенности наиболее распространённых DNS-туннельных протоколов.

В набор включены сценарии, приближённые к следующим типам решений:

- dnscat2 [20];
- iodine [16];
- dns2tcp [17];
- несколько авторских вариантов реализаций DNS – туннелей.

Для каждого варианта изменяются:

- типы DNS-запросов (A, AAAA, TXT, MX, CNAME, SRV);
- формат ответов DNS-сервера;
- способ кодирования полезной нагрузки;
- максимальная длина DNS-меток;
- структура DNS-имён.

В результате модель сталкивается не с единственным шаблоном туннеля, а с множеством различных вариантов организации скрытого канала передачи данных.

Для разнообразия полезной нагрузки туннеля генератор использует несколько типов передаваемой информации совместно с разными типами кодирования информации. Для кодирования используются схемы Base64 и Base32, шестнадцатеричное представление, хэш значения.

В частности моделируются:

- обычный текст;
- JSON-документы;
- псевдослучайные бинарные данные;
- журналы событий;
- смешанные структуры, содержащие одновременно текстовую и случайную информацию.

Использование различных типов полезной нагрузки и способов ее кодирования приводит к изменению энтропии DNS – имён и распределения символов, что делает набор данных более разнообразным и приближает его к условиям эксплуатации реальных DNS-туннелей.

Созданный генератор формирует легитимные DNS-запросы, отражающие различные типы пользовательской активности. Для каждого сценария автоматически генерируются реалистичные доменные имена, соответствующие характерным шаблонам запросов для следующих типов сервисов:

- веб-браузеров;
- сетей доставки контента (CDN);
- почтовых сервисов;
- систем обновления программного обеспечения;
- мобильных приложений;
- потоковых мультимедийных сервисов;
- DNSSEC;
- обратного DNS;
- IoT-устройств;
- VoIP-инфраструктуры;
- облачных платформ;

- аналитических сервисов;
- корпоративной инфраструктуры;
- социальных сетей;
- картографических сервисов;
- новостных ресурсов;
- платёжных систем.

Полученный набор данных представляет собой PCAP-файлы, содержащие по 680 пар пакетов (запрос-ответ). Таким образом, полученный набор данных, корректно использовать, как контролируемый лабораторный набор данных для исследования способности моделей выделять структурные признаки DNS-туннелей.

4. Модель и обучение

В работе [21], была рассмотрена модель, состоящая из слоя Embedding и LSTM, показывающая наилучший результат обнаружения скрытых каналов в DNS. Выполнение такой модели требует значительных ресурсов из-за наличия слоя LSTM. В исследовании [22] используется схожая модель, но основным является сверточный слой, значительно более быстрый, чем LSTM. Для использования модели в IDS, необходимо обеспечить высокую производительность, далее рассматриваются только относительно лёгкие модели. Дополнительным ограничением является формат моделей библиотеки машинного обучения TFLite [23] используемый в LibML [24], который ограничивает набор поддерживаемых архитектур. Например, перспективные модели на основе градиентного бустинга [25] не поддерживаются программной платформой и не будут далее рассматриваться. Архитектуры LSTM, Transformer, CNN-LSTM имеют существенно большее время исполнения, чем MLP и CNN [26-28]. Архитектура CNN предлагает отличный баланс между задержкой и точностью за счет оптимизации аппаратных сверток [27].

В настоящей работе предлагается использовать сверточную нейронную сеть со слоем Embedding, преобразующим последовательность символов в векторные представления для выявления DNS-туннелей. Входом модели является строка размером до 256 символов, являющаяся значением поля DNS-записи (A, MX, TXT и др.). Такой подход позволяет оценивать комплексные закономерности всего доменного имени сразу, без разделения на предопределенные признаки. Целью работы является показать, что модель улучшает точность детектирования по сравнению с классическими средствами, без значимого ухудшения производительности. В работе используется облегчённая архитектура сверточной нейронной сети (1D CNN), предназначенная для классификации символьных последовательностей, архитектура показана на рис. 3.

Упрощенная модель была создана для уменьшения нагрузки и скорости выполнения. Архитектура модели принимает на вход – последовательность из 256 символов, полученных из значения поля записи в DNS пакете. Сначала применяется слой Embedding, который отображает каждый символ в вектор. Полученные векторы поступают на сверточный слой, состоящий из 16 сверточных ядер размера 3 с функцией активации ReLU (Rectified Linear Unit). Ядра перемещаются по последовательности с помощью скользящего окна, выделяя локальные признаки. Сверточный слой является наиболее ресурсоемким в данной сети. Далее, идет слой MaxPooling, уплотняющий значения признаков. Для уменьшения риска переобучения в модели используются механизмы регуляризации. Во-первых, к весовым коэффициентам сверточного и выходного полносвязного слоя применяется L2-регуляризация с коэффициентом 0,001. Во-вторых, после слоя глобального максимального объединения GlobalMaxPooling1D используется слой Dropout с вероятностью отключения нейронов 0,3, что способствует повышению обобщающей способности модели за счет предотвращения чрезмерной адаптации к обучающей выборке.

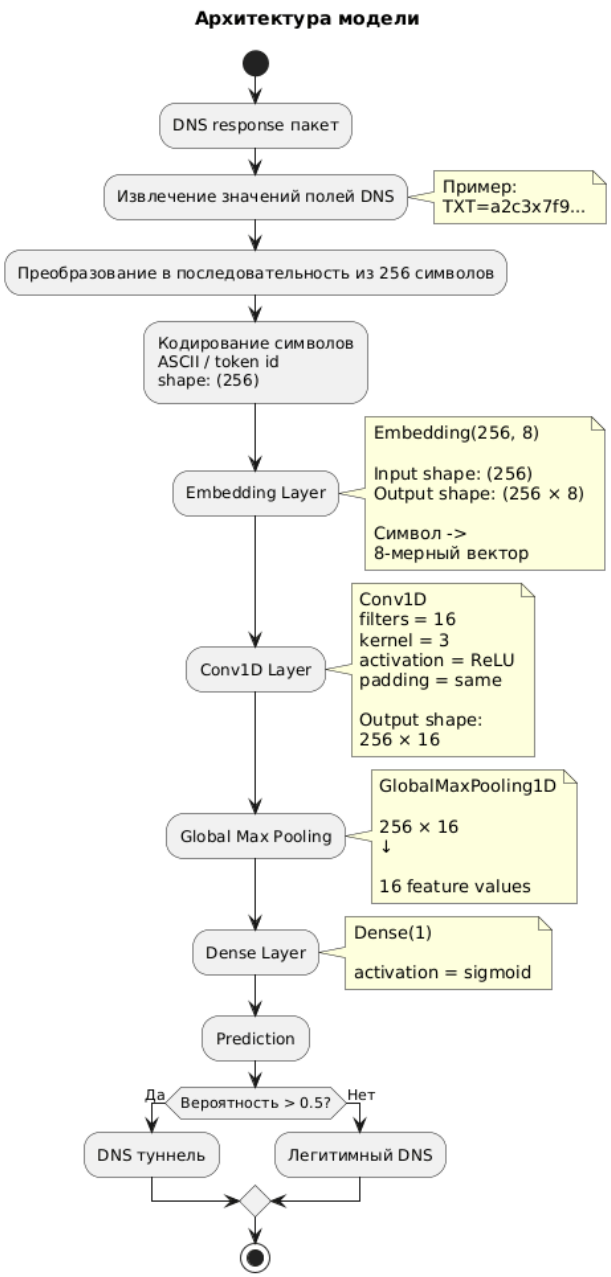

Рис. 3. Архитектура модели.
Fig. 3. Model Architecture.

На выходе используется классический выходной слой бинарной классификации, с функцией активации – сигмоида, которая обладает выходом в интервале [0,1], что хорошо подходит для задачи классификации [29]. В качестве функции потерь используется бинарная кросс-энтропия (обеспечивает корректное обучение вероятностных предсказаний), обучение со стандартным оптимизатором типа Adam, который демонстрирует высокую эффективность и стабильность обучения [30]. Использование функции активации является стандартным решением для бинарной классификации [31]. Обучение модели выполнялось с использованием размера Batch size, равным 256. Максимальное количество эпох обучения выбрано равное 10, однако процесс обучения завершился досрочно при отсутствии улучшения функции потерь на валидационной выборке. В качестве критерия ранней остановки использовался мониторинг значения валидационной функции потерь с параметром patience = 3, при котором обучение прекращалось после трех последовательных эпох без улучшения, а веса модели автоматически восстанавливались к состоянию, соответствующему наилучшему значению функции потерь. Сеть обрабатывает каждую запись DNS, как предложение из символов, что позволяет выявлять паттерны туннельного трафика [21]. Данная архитектура является упрощённым вариантом стандартной сверточной модели для обработки последовательностей и широко используется в задачах анализа текстов и сетевого трафика, обеспечивая хороший баланс между скоростью работы и точностью классификации [32-34].

Отметим, что архитектура модели различается на этапах обучения и последующего использования. На этапе обучения модели используется слой Dropout, выполняющий случайное отключение 30 % нейронов после слоя глобального максимального объединения. Данный механизм применяется исключительно во время обучения и способствует повышению обобщающей способности модели за счет предотвращения совместной адаптации нейронов и снижения риска переобучения [35]. На этапе экспорта модели для последующего применения в системе обнаружения вторжений слой Dropout исключается из вычислительного графа. Это позволяет сократить вычислительные затраты и обеспечить полностью детерминированную работу модели, поскольку механизм случайного отключения нейронов во время классификации не используется [36]. При этом веса обученной модели полностью переносятся, вследствие чего результаты классификации итоговой модели остаются такими же, как у обученной модели.

Архитектура применяемой модели имеет статический размер выходного слоя, равный единице, поскольку задача классификации формулируется как бинарная классификация для каждого DNS ответа, содержащего запись. Такой подход обусловлен задачами практического применения системы обнаружения скрытых каналов. Используется один нейрон на выходном слое с сигмоидальной функцией активации для формирования модели вероятностью принадлежности входного объекта к одному из двух классов, например «нормальный трафик» и «туннельный трафик. Однако применение модели, принимающей решение для каждого отдельного пакета, увеличивает требования к вычислительным ресурсам системы. Это связано с тем, что в сетях с высокой нагрузкой количество DNS-запросов достаточно велико. В модели не используется групповая обработка данных (batching, батчинг). Батчинг может обеспечить большой выигрыш в скорости обработки за счет параллельной обработки, но потребует механизма буферизации, в котором будет скапливаться необходимое количество данных. Если система работает в режиме предотвращения вторжений, то применение буфера внесет задержку в обработку пакетов, что негативно скажется на использовании сети.

Модель была запущена на подготовленном сгенерированном наборе данных для изучения качества обучения. Основные метрики качества сохранились на высоком уровне, что свидетельствует о хорошем качестве обучения. Сведения о полученных результатах представлены в табл. 1.

Табл. 1. Сравнение метрик обучения модели на разных наборах данных.
Table 1. Comparing model training metrics on different datasets.

Тестовая выборка	Accuracy	Precision	Recall	F1	ROC-AUC	PR-AUC
Валидационный набор данных	0.9982	0.9977	0.9982	0.9980	1.0000	1.0000
Сгенерированный набор данных	0.9435	0.9888	0.9118	0.9487	0.9919	0.9931

Для качественного практического применения модели важно выбрать пороговое значение, с помощью которого будет определяться обнаруженный класс. Важно отметить, что данный параметр позволяет изменять баланс между полнотой обнаружения и количеством ложных срабатываний, что делает предложенную модель пригодной для различных сценариев эксплуатации IDS. Графики зависимостей оценки качества модели от порога показаны на рис 4 и 5.

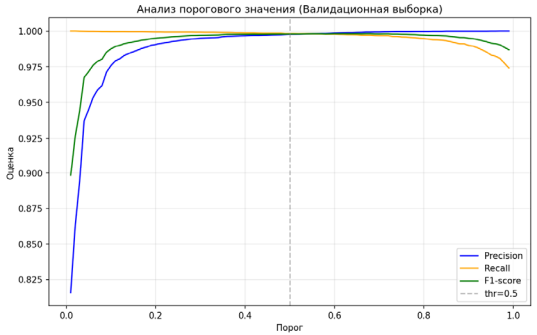

Рис. 4. Анализ порогового значения для валидационной выборки.
Fig. 4. Threshold value analysis for the test sample.

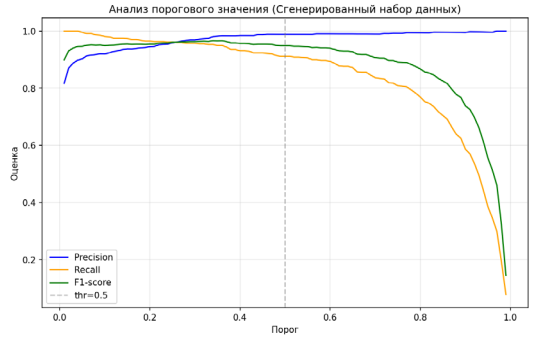

Рис. 5. Анализ порогового значения для независимой выборки.
Fig. 5. Threshold value analysis for a generated independent sample.

Для определения оптимального порогового значения классификации применялась процедура перебора значений порога в диапазоне от 0,01 до 0,99 с шагом 0,01. Для каждого значения порога выходной результат модели преобразовывался в бинарный класс, после чего

рассчитывались значения Precision, Recall и F1-score. В качестве критерия выбора порогового значения использовалось максимальное значение F1-score, обеспечивающее компромисс между полнотой обнаружения DNS-туннелей и количеством ложных срабатываний.

На валидационном наборе данных при подборе порогового значения для максимизации F1-меры оптимальным оказалось значение 0,60, при котором F1-мера составила 0,998. При проверке модели на полностью независимом сгенерированном наборе данных аналогичная процедура подбора порога позволила определить оптимальное значение 0,34, при котором F1-мера составила 0,949. Таким образом, переход к независимому набору данных сопровождается снижением F1-меры и изменением порогового значения, обеспечивающего ее максимальное значение.

При этом модель сохранила высокую способность к разделению двух классов, что подтверждается значением ROC-AUC, равным 0,992. Изменение оптимального порога при сохранении высокого значения ROC-AUC указывает на различия в распределении выходных оценок модели между наборами данных. Следовательно, способность модели различать легитимный и туннельный трафик сохраняется, однако порог, обеспечивающий наилучший баланс точности и полноты, может зависеть от характеристик анализируемого трафика.

В связи с этим фиксированное пороговое значение не следует рассматривать как универсальное. В дальнейших экспериментах в качестве рабочего используется порог 0,5 как нейтральное значение при отсутствии предварительной информации о профиле анализируемого трафика. При этом для конкретных условий эксплуатации значение порога может быть дополнительно скорректировано с учетом требований к соотношению полноты обнаружения и количества ложных срабатываний.

5. Интеграция модуля в Snort3

Свободно распространяемая система обнаружения вторжений Snort3 [37] заслужила популярность у пользователей, в том числе благодаря широко используемым и часто обновляемым правилам от сообщества [38] для системы Snort3. Однако, этот набор правил не включают в себя сигнатуры для обнаружения туннелей. Для того чтобы последующее сравнение было более объективным, на основе [39] были подготовлены 9 правил способные детектировать некоторые DNS туннели.

Snort3 написан на языке программирования C++, архитектура представляет собой модульную архитектуру состоящая из множества встроенных и внешних плагинов служащих для расширения функциональности. Основными типами плагинов являются инспекторы, кодеки, логгеры, препроцессоры [40]. Для практического эксперимента модель встроена в движок Snort3. Был разработан модуль, перехватывающий DNS пакеты и передающий значения полей в модель для классификации. Если ответ модели, то есть вероятность наличия туннеля превышает пороговое значение (выбрано стандартное значение при решении задачи бинарной классификации 0.5 [41]), то генерируется событие о детектировании DNS туннеля. Модуль основан на библиотеке LibML от Cisco Talos [42]. Упрощенная диаграмма компонентов показана на рис. 6, где обозначенные модули разделены по цветам в соответствии с уровнем доработок. Синим цветом показаны модули без доработок, желтым - частично доработанные, зеленым – разработанные модули.

По умолчанию в Snort3 есть утилиты позволяющие отслеживать производительность правил и их нагрузку на процессор, но формат представления данных не позволяет провести сравнительный анализ с конфигурацией, использующей модуль машинного обучения. Дополнительно были доработаны другие модули для подсчета статистики по времени работы модулей машинного обучения и сигнатурного анализа на базе Hyperscan от Intel [43]. Счетчики времени были добавлены в конвейер, обрабатывающий пакеты, для определения времени обработки данных на каждом этапе, включая парсинг пакета, сопоставление правил, исполнение модели машинного обучения и другие процессы.

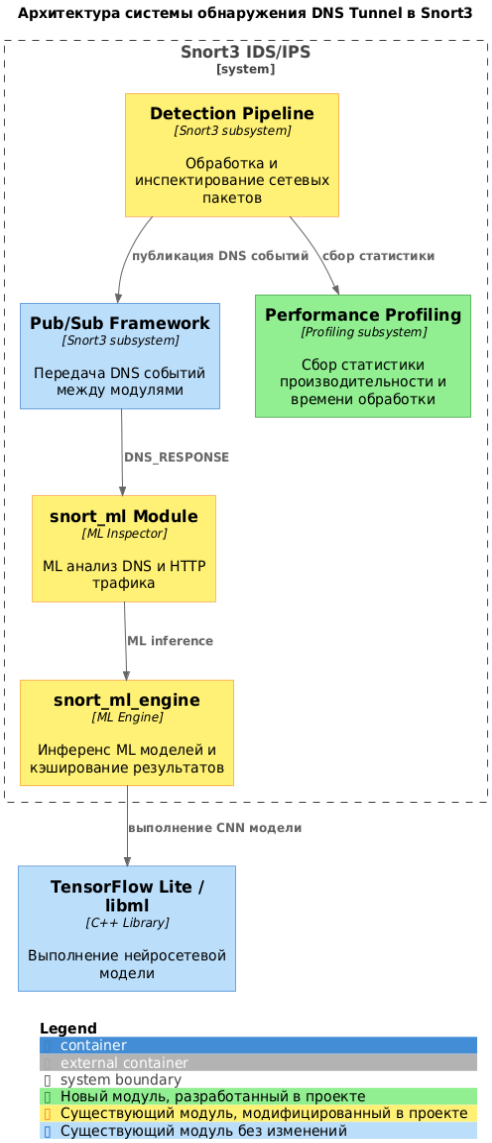

Рис. 6. Упрощенная архитектура модуля в Snort3.
Fig. 6. Simplified module architecture in Snort3.

6. Результаты эксперимента

Для определения влияния на работу в условиях разнообразного трафика был использован набор данных CIC-IDS-2017 [44]. Выбранный набор разбит на части по дням недели с указанием активности. Для эксперимента выбраны записи данных за понедельник и четверг,

т.к. они содержат веб - трафик и DNS запросы. Трафик понедельника содержит только легитимный трафик, включающий в себя DNS запросы. Запись трафика за четверг включает в себя легитимную активность, веб-атаки, вредоносную активность, выгрузку данных и так далее.

Для оценки вычислительных затрат ML-подхода по сравнению с традиционным сигнатурным обнаружением был проведен дополнительный эксперимент, в котором режимы работы IDS с использованием только правил и только ML-модуля рассматривались раздельно.

При обработке легитимного DNS-трафика использование только сигнатурных правил обеспечило среднее время обработки около 9,3 мкс/пакет, тогда как при использовании только ML-модуля данный показатель составил около 10,6 мкс/пакет. Таким образом, применение ML-модели увеличило среднее время обработки пакета примерно на 13 %.

Для трафика, состоящего только из DNS-туннелей, среднее время обработки пакета при использовании сигнатурного подхода составило около 6,6 мкс, а при использовании только ML-модуля — около 5,8 мкс. В этом случае оказалось, что в данном эксперименте обработка пакетов с использованием ML-модуля выполнялась в среднем примерно на 13 % быстрее сигнатурного подхода.

Для оценки влияния ML-модуля в условиях, близких к практической эксплуатации сетевой системы обнаружения вторжений, дополнительно был проведен эксперимент на смешанном трафике, содержащем легитимные DNS-запросы и DNS-туннели. В данном режиме среднее время выполнения ML-модели составило около 0,2 мкс на пакет, а сопоставление сигнатурных правил занимало около 2,2 мкс. Еще около 0,3 мкс приходилось на декодирование и разбор пакетов. Общее время инспектирования пакетов составляло около 6,7 мкс, а суммарное время обработки одного пакета — около 10 мкс. Временная последовательность показана на рис. 7.

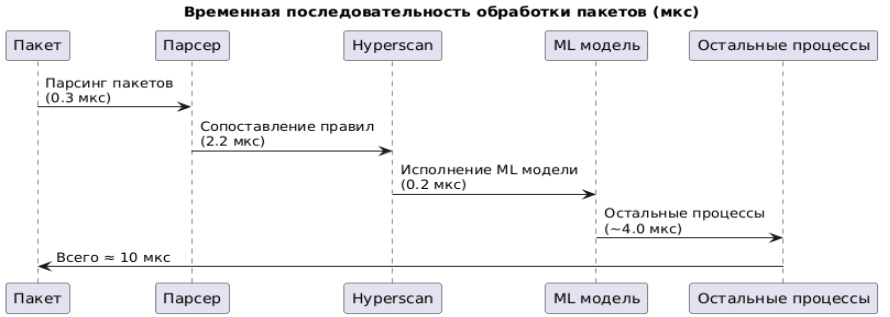

Рис. 7. Временная последовательность обработки пакетов.
Fig. 7. Time sequence of packet processing.

Таким образом среднее время обработки пакета увеличилось на 2%, но дало возможность обнаружения DNS туннелей.

Результаты исследования зависимости производительности IDS от профиля сетевого трафика показаны в табл. 2. Сравнение конфигураций Snort 3 с использованием только сигнатурных правил и совместно с сигнатурными правилами и ML-моделью показывает, что влияние модели на производительность зависит от профиля обрабатываемого трафика.

Для наборов данных из CIC-IDS 2017, соответствующих трафику понедельника и четверга, добавление ML-модели практически не изменяет среднее время обработки пакета. Для понедельника данный показатель увеличивается с 10,0 до 10,4 мкс, а для четверга, напротив, уменьшается с 10,3 до 10,2 мкс. При этом среднее время обработки пакета ML-моделью

составляет около 0,2 мкс в обоих случаях. Таким образом, при данных профилях трафика дополнительные вычислительные затраты, связанные с применением модели, оказывают незначительное влияние на общую производительность системы.

Для набора данных, содержащего только DNS-туннели, среднее время обработки пакета при использовании конфигурации Snort 3 + Rules + ML составило 10,1 мкс против 6,5 мкс при использовании только правил. В данном случае каждый обрабатываемый пакет является DNS-пакетом, поэтому ML-модель запускается для каждого пакета. Среднее время её выполнения составило 2,2 мкс на пакет. В результате добавление ML-модуля привело к увеличению среднего времени обработки примерно на 55 %.

Для легитимного DNS-трафика влияние ML-модуля оказалось еще более заметным: среднее время обработки увеличилось с 9,3 до 15,6 мкс на пакет, при этом среднее время исполнения модели составило 4,7 мкс. Поскольку весь рассматриваемый трафик представлен DNS-пакетами, модель вызывается для каждого ответного пакета, что приводит к увеличению времени работы системы примерно на 68 %. При этом среднее время сопоставления правил изменилось незначительно — с 4,5 до 5,0 мкс.

Табл. 2. Сравнение среднего времени обработки пакета в разных профилях трафика.
Table 2. Comparison of average packet processing time in different traffic profiles.

Набор данных	Конфигурация	Среднее время обработки пакета, мкс.	Среднее время обработки ML модели на пакет, мкс.	Среднее время на сопоставление правил, мкс.
CIC-IDS 2017 Понедельник	Snort3 + Rules	10.0	-	2.4
	Snort3 + Rules + ML	10.4	0.2	2.4
CIC-IDS 2017 Четверг	Snort3 + Rules	10.3	-	2.3
	Snort3 + Rules + ML	10.2	0.2	2.2
Набор данных – туннели	Snort3 + Rules	6.5	-	3.2
	Snort3 + Rules + ML	10.1	2.2	3.4
Набор данных – легитимные	Snort3 + Rules	9.3	-	4.5
	Snort3 + Rules + ML	15.6	4.7	5.0

Результаты показывают, что влияние ML-модуля существенно зависит от состава обрабатываемого трафика. На профилях, содержащих различные типы сетевого трафика, среднее время обработки пакета при включении ML-модуля практически не изменилось. В то же время при обработке трафика, состоящего только из DNS-пакетов, ML-модель вызывается для каждого пакета с ответом, вследствие чего увеличение времени обработки становится более выраженным. Таким образом, величина вычислительных затрат, вносимых ML-модулем, определяется не только временем выполнения самой модели, но и долей пакетов, передаваемых на ML-анализ. Поскольку модель применяется только к DNS-пакетам, при обработке смешанного сетевого трафика её влияние на среднее время обработки всего трафика снижается пропорционально доле DNS-пакетов.

В рамках задачи обнаружения сетевых атак положительным классом (positive class) выбран класс «туннель», поскольку он соответствует целевому событию – наличию аномального или вредоносного трафика. Это позволяет корректно интерпретировать метрики точность, полноту и F1-меру с точки зрения способности системы обнаруживать атаки.

Матрицы ошибок на рис. 8 и 9 показывают результаты обнаружения туннелей на размеченном трафике, который использовался для обучения модели. Классическое решение дает некоторое число ложных обнаружений детектирует около 16% туннелей, модуль машинного обучения дает единичные ложные срабатывания и детектирует 100% туннелей. Модель использовалась с порогом классификации 0.64 оптимизированный по всему набору данных при помощи процедуры подбора порога, описанной выше.

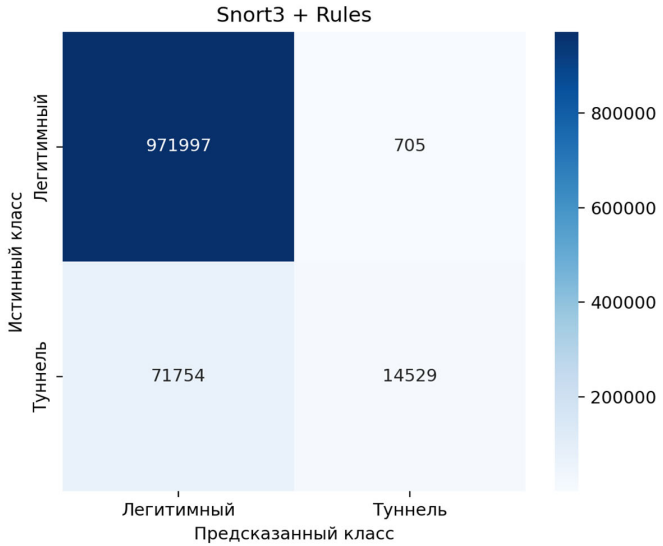

Рис. 8. Матрица ошибок Snort3 + Rules.
Fig. 8. The Snort3 Error Matrix + Rules.

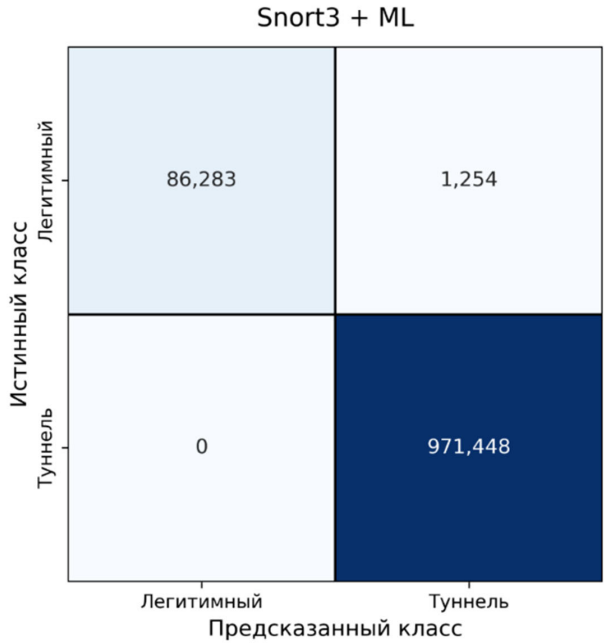

Рис. 9. Матрица ошибок Snort3 + ML.
Fig. 9. The Snort3 + ML error Matrix.

В табл. 3 представлено сравнение метрик качества с моделью из работы [22]. Результаты экспериментов демонстрируют, что сигнатурный подход обеспечивает высокую точность классификации, однако характеризуется крайне низкой полнотой обнаружения атак, что делает его недостаточным для практического применения. Введение методов машинного обучения позволяет устранить данный недостаток, обеспечивая практически полное обнаружение атак при сохранении высокой точности. Модель, полученная в данной работе, демонстрирует сопоставимое или более высокое качество по сравнению с моделью Byte-level CNN [22]. Следует отметить, что приведенное сравнение с моделью Byte-level CNN не является прямым экспериментальным сопоставлением, поскольку результаты получены на наборах данных и при экспериментальных условиях, которые не являются полностью идентичными. Несмотря на использование сопоставимых наборов данных и близких сценариев обнаружения, различия в разбиении данных, предварительной обработке, архитектурах моделей и вычислительных платформах могут влиять на итоговые показатели. Поэтому приведенные значения используются для общего качественного сопоставления.

Табл. 3. Сравнение F1-меры –решений.
Table 3. Comparison of F1 measures.

Решение	F1-мера	Accuracy (%)	Precision (%)	Recall (%)
Snort3 + Rules	≈ 0.286	93.16	95.37	16.84
Snort3 + ML	0.99996 (~ 1.0)	99.99 (~ 100.0)	99.99 (~ 100.0)	100.0
Byte-level CNN[22]	0.9998	99.98	100.00	99.96

7. Заключение

В работе рассмотрен вопрос практического применения обнаружения DNS-туннелей с помощью машинного обучения. Показано, что классические методы обнаружения имеют ограничения в обнаружении скрытых каналов. Создана модель способная обнаруживать туннели с высокой точностью без необходимости предварительной выработки признаков. В экспериментах модель обнаружила все атакующие DNS-запросы при очень малом числе ложных тревог, значительно превосходя сигнатуры. При этом скорость обработки после интеграции модели сопоставима со скоростью обработки правил при использовании современного решения Hyperscan, которое превосходит используемый по умолчанию алгоритм Ахо—Корасика [45].

Важным преимуществом модели является предсказуемое время исполнения. За счет использования входной последовательности фиксированной длины и неизменной структуры вычислительного графа количество операций при обработке каждого входного объекта остается постоянным. Это обеспечивает более стабильное время обработки по сравнению с сигнатурным подходом, в котором производительность может зависеть от количества проверяемых правил и профиля сетевого трафика. Проведенные эксперименты подтверждают незначительный разброс времени выполнения модели при обработке различных входных данных.

Список литературы / References

[1]. Wang Y., Zhou A., Liao S., Zheng R., Hu R., Zhang L. A comprehensive survey on DNS tunnel detection. Computer Networks, 2021, vol. 197, p. 108322. DOI: 10.1016/j.comnet.2021.108322.

[2]. Wendzel S., Zander S., Fechner B., Herdin C. Trends and Challenges in Network Covert Channels Countermeasures. Applied Sciences, 2021, vol. 11, no. 4, p. 1641. DOI: 10.3390/app11041641.
[3]. What Is a DNS Proxy? How Do DNS Proxies Work? Available at: https://www.akamai.com/glossary/what-is-a-dns-proxy, accessed 22.02.2026.
[4]. Rascagneres P. New FrameworkPOS variant exfiltrates data via DNS requests. Available at: https://www.gdatasoftware.com/blog/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests, accessed 15.02.2026.
[5]. Zhang Z., Li H., Zhao Y., Lin C., Liu J. POS: An Operator Scheduling Framework for Multi-model Inference on Edge Intelligent Computing. Proceedings of the 22nd International Conference on Information Processing in Sensor Networks (IPSN '23). New York, NY, USA: Association for Computing Machinery, 2023, pp. 1-12. DOI: 10.1145/3583120.3586966.
[6]. Nadler A., Aminov A., Shabtai A. Detection of malicious and low throughput data exfiltration over the DNS protocol. Computers & Security, 2019, vol. 80, pp. 36-53. DOI: 10.1016/j.cose.2018.09.006.
[7]. Qi C., Chen X., Xu C., Shi J., Liu P. A Bigram based Real Time DNS Tunnel Detection Approach. Procedia Computer Science, 2013, vol. 17, pp. 852-860. DOI: 10.1016/j.procs.2013.05.109.
[8]. Amirov N., Isik B., Tuncer B., Bahtiyar Ş. DNS Tunneling: Threat Landscape and Improved Detection Solutions. DNS Tunneling, 2025. DOI: 10.1007/978-3-031-91548-4_5.
[9]. Sammour M., Othman M.F.I., Hassan A., Bhais O., Talib M.S. Advanced DNS tunneling detection: a hybrid reinforcement learning and metaheuristic approach. Frontiers in Computer Science, 2026, vol. 7. Article 1626646. DOI: 10.3389/fcomp.2025.1626646.
[10]. Lal A., Prasad A., Kumar A., Kumar S. DNS-Tunnet: A Hybrid Approach for DNS Tunneling Detection. 2022 4th International Conference on Advances in Computer Technology, Information Science and Communications (CTISC), 2022, pp. 1-6.
[11]. Al-Ibraheemi F.A., AL-Ibraheemi S., Amintoosi H. A hybrid method of genetic algorithm and support vector machine for DNS tunneling detection. International Journal of Electrical and Computer Engineering, 2021, vol. 11, no. 2, pp. 1666-1674. DOI: 10.11591/ijece.v11i2.pp1666-1674.
[12]. Бубнов Я.В. Модели и алгоритмы для обнаружения сетевых атак на основе анализа характеристик DNS-запросов: автореф. дисс. канд. техн. наук: 05.13.19. Минск: БГУИР, 2021. 21 с.
[13]. Gwon H., Lee C., Keum R., Choi H. Network Intrusion Detection based on LSTM and Feature Embedding. DOI: 10.48550/arXiv.1911.11552.
[14]. DNS-Tunnel-Datasets: tunnel. Available at: https://github.com/ggyggy666/DNS-Tunnel-Datasets/tree/main/tunnel, accessed 07.02.2026.
[15]. Gao G., Niu W., Gong J., Gu D., Li S., Zhang M., Zhang X. GraphTunnel: Robust DNS Tunnel Detection Based on DNS Recursive Resolution Graph. IEEE Transactions on Information Forensics and Security, 2024, vol. 19, pp. 7705-7719. DOI: 10.1109/TIFS.2024.3443596.
[16]. Ekman E. iodine. Available at: https://github.com/yarrick/iodine, accessed 07.08.2026.
[17]. dns2tcp. Kali Linux Tools. Available at: https://www.kali.org/tools/dns2tcp/, accessed 07.08.2026.
[18]. dnscapy. Google Code Archive. Available at: https://code.google.com/archive/p/dnscapy/, accessed 07.08.2026.
[19]. Nussbaum L. tuns. Available at: https://github.com/lnussbaum/tuns, accessed 07.08.2026.
[20]. dnscat2. Kali Linux Tools. Available at: https://www.kali.org/tools/dnscat2/, accessed 07.08.2026.
[21]. Chen S., Lang B., Liu H., Li D., Gao C. DNS covert channel detection method using the LSTM model. Computers & Security, 2021, vol. 104, p. 102095. DOI: 10.1016/j.cose.2020.102095.
[22]. Liu C., Dai L., Cui W., Lin T. A Byte-level CNN Method to Detect DNS Tunnels. 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC), 2019, pp. 1-8. DOI: 10.1109/IPCCC47392.2019.8958714.
[23]. Module: tf.lite. TensorFlow v2.16.1. Available at: https://www.tensorflow.org/api_docs/python/tf/lite, accessed 07.08.2026.
[24]. Snort 3.0 Team. snort3/libml. Available at: https://github.com/snort3/libml, accessed 07.08.2026.
[25]. Chen T., Guestrin C. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). New York, NY, USA: Association for Computing Machinery, 2016, pp. 785-794. DOI: 10.1145/2939672.2939785.
[26]. Pettersson J., Falkman P. Comparison of LSTM, Transformers, and MLP-mixer neural networks for gaze based human intention prediction. Frontiers in Neurorobotics, 2023, vol. 17, p. 1157957. DOI: 10.3389/fnbot.2023.1157957.
[27]. Zhao Y., Wang G., Tang C., Luo C., Zeng W., Zha Z.-J. A Battle of Network Structures: An Empirical Study of CNN, Transformer, and MLP. DOI: 10.48550/arXiv.2108.13002.

[28]. Abukhousa E., Zonouz S., Meliopoulos A.P.S. Latency-Aware Deep Learning Benchmark for Real-Time Cyber-Physical Attack and Fault Classification in Inverter-Dominated Power Grids. 2026 IEEE/PES Transmission and Distribution Conference and Exposition (T&D), 2026, pp. 1-5.
[29]. Pratiwi H., Windarto A.P., Susliansyah S., Aria R.R., Susilowati S., Rahayu L.K., Fitriani Y., Merdekawati A., Rahadjeng I.R. Sigmoid Activation Function in Selecting the Best Model of Artificial Neural Networks. Journal of Physics: Conference Series, 2020, vol. 1471, no. 1, p. 012010. DOI: 10.1088/1742-6596/1471/1/012010.
[30]. Kingma D.P., Ba J. Adam: A Method for Stochastic Optimization. DOI: 10.48550/arXiv.1412.6980.
[31]. Abualghanam O., Alazzam H., Elshqeirat B., Qatawneh M., Almaiah M.A. Real-Time Detection System for Data Exfiltration over DNS Tunneling Using Machine Learning. Electronics, 2023, vol. 12, no. 6, p. 1467. DOI: 10.3390/electronics12061467.
[32]. Johnson R., Zhang T. Effective Use of Word Order for Text Categorization with Convolutional Neural Networks. DOI: 10.48550/arXiv.1412.1058.
[33]. Kim Y. Convolutional Neural Networks for Sentence Classification. DOI: 10.48550/arXiv.1408.5882.
[34]. TextConvoNet: a convolutional neural network based architecture for text classification. Available at: https://pmc.ncbi.nlm.nih.gov/articles/PMC9589611/, accessed 21.03.2026.
[35]. Srivastava N., Hinton G., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 2014, vol. 15, no. 56, pp. 1929-1958. DOI: 10.5555/2627435.2670313.
[36]. Baldi P., Sadowski P. The Dropout Learning Algorithm. Artificial Intelligence, 2014, vol. 210, pp. 78-122. DOI: 10.1016/j.artint.2014.03.008.
[37]. Snort 3.0 Team. snort3/snort3. Available at: https://github.com/snort3/snort3, accessed 07.08.2026.
[38]. What are Community Rules? Available at: https://www.snort.org/faq/what-are-community-rules, accessed 16.02.2026.
[39]. Adiwal S., Rajendran B., Shetty D.P., Sudarsan S.D. DNS Intrusion Detection (DID) – A SNORT-based solution to detect DNS Amplification and DNS Tunneling attacks. Franklin Open, 2023, vol. 2, p. 100010. DOI: 10.1016/j.fraope.2023.100010.
[40]. Thorarensen C. A Performance Analysis of Intrusion Detection with Snort and Security Information Management: Independent thesis, Advanced level, degree of Master (Two Years). Linköping: Linköping University, Department of Computer and Information Science, Database and Information Techniques, 2021, 84 p. Available at: https://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-177602, accessed 07.08.2026.
[41]. Freeman E.A., Moisen G.G. A comparison of the performance of threshold criteria for binary classification in terms of predicted prevalence and kappa. Ecological Modelling, 2008, vol. 217, no. 1, pp. 48-58. DOI: 10.1016/j.ecolmodel.2008.05.015.
[42]. Stultz B. Talos launching new machine learning-based exploit detection engine. Snort Blog, 2024. Available at: https://blog.snort.org/2024/03/talos-launching-new-machine-learning.html, accessed 11.05.2026.
[43]. Hyperscan. Intel Hyperscan Documentation. Available at: https://intel.github.io/hyperscan/, accessed 11.05.2026.
[44]. IDS 2017. Datasets. Research. Canadian Institute for Cybersecurity. University of New Brunswick. Available at: https://www.unb.ca/cic/datasets/ids-2017.html, accessed 07.02.2026.
[45]. Wang X., Hong Y., Chang H., Park K., Langdale G., Hu J., Zhu H. Hyperscan: A Fast Multi-pattern Regex Matcher for Modern CPUs. Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation (NSDI '19). USA: USENIX Association, 2019, pp. 631-648.

Информация об авторах / Information about authors

Никита Денисович МАРИНИН – аспирант, Департамент электронной инженерии МИЭМ НИУ ВШЭ. Сфера научных интересов: анализ сетевого трафика с помощью машинного обучения.

Nikita Denisovich MARININ – postgraduate student of the HSE Tikhonov Moscow Institute of Electronics and Mathematics. Research interests: network traffic analysis using machine learning.

Александр Игоревич ГЕТЬМАН – кандидат физико-математических наук, старший научный сотрудник ИСП РАН, ассистент ВМК МГУ, доцент ВШЭ и МФТИ. Сфера научных интересов: анализ бинарного кода, восстановление форматов данных, анализ и классификация сетевого трафика.

Aleksandr Igorevich GETMAN – Cand. Sci. (Phys.-Math.), senior researcher at ISP RAS, assistant at CMC MSU, associate professor at HSE and MIPT. Research interests: binary code analysis, data format recovery, network traffic analysis and classification.