Научная статья
УДК 004.056
https://doi.org/10.31854/1813-324X-2026-12-2-74-91
EDN:GEUZWL

Представление промышленного трафика и кодирование протокольной семантики для трансформерных моделей в малых сетях Industrial Ethernet

Нгуен Чунг Буи[1] ✉, bui.nch@phystech.edu
Федор Федорович Пащенко[1, 2], pif-70@yandex.ru
Фи Тхань То[1], to.ft@phystech.edu

[1]Московский физико-технический институт (национальный исследовательский университет), Москва, 117303, Российская Федерация
[2]Институт проблем управления РАН им. академика В.А. Трапезникова, Москва, 117342, Российская Федерация

Аннотация

В современных индустриальных сетях Industrial Ethernet обнаружение аномалий и кибератак требует учета протокольной семантики трафика. ***Актуальность*** *исследования обусловлена ростом числа атак на промышленные системы управления, усилением интеграции производственных и корпоративных сетей, а также широким использованием промышленных протоколов, часть которых изначально не ориентировалась на современные требования к кибербезопасности. Традиционные сигнатурные средства обнаружения ограниченно выявляют ранее неизвестные и скрытные атаки, тогда как подходы, основанные только на статистических характеристиках потоков, нередко теряют важную информацию о логике обмена, ролях сообщений и особенностях прикладного уровня промышленных протоколов. Дополнительную сложность создает дефицит размеченных данных, характерный для реальных промышленных объектов, что затрудняет обучение устойчивых моделей обнаружения атак.*

Цель: *разработать способ представления промышленного трафика в виде токенизированных последовательностей, пригодных для применения трансформерных моделей в малых сетях Industrial Ethernet.*

Методы. *Использованы потоково-сессионное агрегирование пакетов, семантическая токенизация полей протоколов Modbus/TCP и OPC UA, эмбеддинги с позиционным кодированием, предварительное обучение в self-supervised режиме и последующее дообучение модели на задаче классификации сессий. Оценка подхода выполнена на наборе данных Malware in Smart Factory.*

Результаты. *Разработано протокольно-ориентированное представление сетевых сессий, сохраняющее контекст обмена и особенности промышленных протоколов. На наборе данных Malware in Smart Factory трансформерная модель достигла F1-меры 99,3 % при обнаружении атак и превзошла модели LSTM и CNN; предварительное обучение дополнительно повысило качество классификации.*

Теоретическая и практическая значимость. *Предложенный подход формирует единый входной формат для анализа трафика Modbus/TCP и OPC UA и может использоваться при построении систем обнаружения аномалий и вторжений в малых промышленных сетях.*

Ключевые слова: *Industrial Ethernet, Modbus/TCP, OPC UA, трансформер, сетевой трафик, токенизация, позиционное кодирование, обнаружение вторжений, self-supervised обучение*

Ссылка для цитирования: Буи Н.Ч., Пащенко Ф.Ф., То Ф.Т. Представление промышленного трафика и кодирование протокольной семантики для трансформерных моделей в малых сетях Industrial Ethernet // Труды учебных заведений связи. 2026. Т. 12. № 2. С. 74–91. DOI:10.31854/1813-324X-2026-12-2-74-91. EDN:GEUZWL

Original research
https://doi.org/10.31854/1813-324X-2026-12-2-74-91
EDN:GEUZWL

Representation of Industrial Traffic and Encoding of Protocol Semantics for Transformer Models in Small Industrial Ethernet Networks

Nguyen Trung Bui[1] ✉, bui.nch@phystech.edu
Fedor Fedorovich Pashchenko[1, 2], pif-70@yandex.ru
Phi Thanh To[1], to.ft@phystech.edu

[1]Moscow Institute of Physics and Technology,
Moscow, 117303, Russian Federation
[2]Institute of Control Sciences RAS,
Moscow, 190031, Russian Federation

Annotation

Relevance. *In modern Industrial Ethernet networks, anomaly and cyberattack detection requires taking the protocol semantics of network traffic into account. The relevance of this study is determined by the growing number of attacks on industrial control systems, the increasing integration of production and corporate networks, and the widespread use of industrial protocols, some of which were not originally designed to meet modern cybersecurity requirements. Traditional signature-based detection tools are limited in their ability to identify previously unknown and stealthy attacks, while approaches based solely on statistical flow characteristics often lose important information about communication logic, message roles, and application-layer features of industrial protocols. An additional challenge is the shortage of labeled data typical of real industrial environments, which complicates the training of robust attack detection models.*

Objective. *To develop and evaluate a method for representing industrial traffic as tokenized sequences suitable for transformer models in small Industrial Ethernet networks.*

Methods. *Flow- and session-level packet aggregation, semantic tokenization of Modbus/TCP and OPC UA protocol fields, embeddings with positional encoding, self-supervised pre-training, and subsequent fine-tuning of the model for session classification were used. The approach was evaluated on the Malware in Smart Factory dataset.*

Results. *A protocol-aware representation of network sessions that preserves communication context and the specific features of industrial protocols was developed. On the Malware in Smart Factory dataset, the transformer model achieved an F1-score of 99.3 % in attack detection and outperformed LSTM and CNN models; pre-training further improved classification performance.*

Theoretical and practical significance. *The proposed approach provides a unified input format for analyzing Modbus/TCP and OPC UA traffic and can be used in anomaly and intrusion detection systems for small industrial networks.*

Keywords: *Industrial Ethernet, Modbus/TCP, OPC UA, transformer models, network traffic, tokenization, positional encoding, intrusion detection, self-supervised learning*

For citation: Bui N.T., Pashchenko F.F., To P.T. Representation of Industrial Traffic and Encoding of Protocol Semantics for Transformer Models in Small Industrial Ethernet Networks. *Proceedings of Telecommunication Universities.* 2026;12(2):74–91. (in Russ.) DOI:10.31854/1813-324X-2026-12-2-74-91. EDN:GEUZWL

1. Введение

Индустриальные системы управления (ICS, *аббр. от англ.* Industrial Control Systems) и сети Industrial Ethernet широко используются для соединения контроллеров, датчиков и исполнительных механизмов на производственных объектах. В таких сетях передаются данные технологического процесса, команды управления, телеметрия и другие критически важные сообщения. Надежность и безопасность обмена информацией являются приоритетными, однако многие промышленно-коммуникационные протоколы исторически разрабатыва-

лись без учета современных требований к кибербезопасности [1]. Например, протокол Modbus изначально не предусматривал аутентификацию или шифрование, из-за чего трафик уязвим для перехвата и подмены. С другой стороны, более новые протоколы, такие как OPC UA, проектировались с встроенными механизмами безопасности, но их сложность выше. В последние годы значительно возросло число кибератак на ICS, включая случаи вредоносного вмешательства (Stuxnet, Triton и др.), что подтверждает актуальность разработки эффективных средств мониторинга и обнаружения аномалий в промышленном трафике [1].

Классическим средством защиты сетей являются системы обнаружения вторжений (IDS, аббр. от англ. Intrusion Detection System), анализирующие сетевой трафик на наличие известных сигнатур или аномальных паттернов. Однако стандартные IDS с сигнатурным анализом плохо справляются с ранее неизвестными атаками и зашифрованным трафиком. Машинное обучение и, в особенности глубокие нейронные сети, рассматриваются как перспективный инструмент для построения IDS нового поколения, способных выявлять скрытые закономерности в сетевых данных [1]. Ранее в задачах классификации трафика применялись статистические методы и алгоритмы машинного обучения (деревья решений, SVM и др.), а также модели на основе сверточных (CNN, *аббр. от англ*. Convolutional Neural Network) и рекуррентных (RNN / LSTM, *аббр. от англ*. Recurrent Neural Network / Long Short-Term Memory) нейронных сетей [2, 3]. Эти подходы дали определенные улучшения по сравнению с ручными правилами, позволяя автоматически извлекать признаки из исходных данных. Тем не менее, они обладают рядом ограничений. В частности, RNN склонны терять долгосрочные зависимости при обработке длинных последовательностей и могут требовать тщательной настройки для разных протоколов, тогда как CNN эффективны для локальных шаблонов, но с трудом улавливают глобальный контекст без значительного увеличения глубины сети.

Трансформерная архитектура, изначально разработанная для обработки последовательностей в задачах естественного языка, в последние годы привлекла внимание исследователей сетевой безопасности. Трансформеры способны параллельно обрабатывать длинные последовательности и учитывать отношения между любыми позициями посредством механизма самовнимания. Это создает предпосылки для лучшего учета контекста в сетевых коммуникациях, где важны взаимосвязи между событиями, удаленными в последовательности по времени. Кроме того, трансформерные модели позволяют эффективно использовать self-supervised обучение, то есть обучение на больших объемах неразмеченных данных, что особенно актуально для промышленных сетей, где размеченные данные об атаках ограничены или получение их затруднительно [2, 4]. Уже появляются работы, демонстрирующие успешное применение трансформеров для сетевого трафика: например, предложены гибридные модели, сочетающие трансформер и LSTM для классификации вредоносного трафика с высокой точностью (около 99 %) [2], а также специализированные сетевые трансформеры для аномалий в ICS, способные выявлять нештатное поведение устройств без ручной разметки благодаря самообучению на структуре сетевого обмена [4].

Анализ предметной области и обзор литературы

Промышленные сети Industrial Ethernet характеризуются высокой долей периодического и детерминированного обмена, что позволяет строить IDS как на основе правил и спецификаций, так и на основе данных. В литературе подходы к обнаружению атак в ICS условно можно разделить на несколько классов. Общие обзоры методов IDS и применения машинного обучения для кибербезопасности и киберфизических систем приведены, например, в [5, 6]. Для ICS и операционных технологий (OT, *аббр. от англ.* Operational Technology) важны также экспериментальные стенды и наборы данных, обеспечивающие воспроизводимость исследований; среди наиболее известных – SWaT и WADI [7, 8], а примеры IDS, ориентированные на реальные промышленные сети, представлены в [9].

Сигнатурные и правило-ориентированные IDS (например, на базе Snort / Suricata и протокольных правил) хорошо выявляют известные шаблоны, но требуют постоянного ручного сопровождения и обычно слабо применимы к ранее неизвестным атакам. В промышленной среде дополнительно используются «белые списки» допустимых команд и коммуникационных сценариев согласно рекомендациям по OT-безопасности [10].

Модельно-ориентированные и протокольно-специфичные методы используют особенности промышленных протоколов и типовые сценарии взаимодействия между операторскими интерфейсами и программируемыми логическими контроллерами. Так, для Modbus/TCP предлагается строить формальные модели допустимых последовательностей сообщений, вплоть до детерминированных автоматов, что повышает интерпретируемость и точность в стабильных средах [11]. Ограничением является необходимость настройки под конкретный объект и протокол, а также чувствительность к изменениям конфигурации.

Методы машинного и глубокого обучения, применяемые к агрегированным потокам и сессиям, предполагают предварительное сворачивание пакетов в потоки и расчет статистических признаков (размеры, интервалы, направления, количество пакетов

и т. д.), после чего используются классические модели (RF, SVM) или нейросети. Такой подход обеспечивает компактность и удобство обучения, однако часть прикладной семантики протокола может теряться [1, 12]. Для онлайн-обнаружения аномалий по потоковым признакам также применяются ансамбли автокодировщиков, например Kitsune [13].

Глубокое обучение на низкоуровневых представлениях (байты, первые *N* байт пакета, «карты» байтов) позволяет автоматически извлекать признаки и широко используется для классификации сетевого трафика, в том числе в сценариях с ограниченной видимостью содержимого [14]. Однако в контексте ICS такой подход сталкивается с разнородностью протоколов и риском переобучения на артефакты конкретного стенда или реализации стека.

Трансформерные и self-supervised модели опираются на способность трансформера учитывать дальние зависимости в последовательностях [15], а также на идеи предобучения, характерные для моделей семейства BERT, в частности на маскированное моделирование токенов (MTM, *аббр. от англ.* Masked Token Modeling) [16]. Для сетевого трафика предлагаются решения семейства BERT (TSFN [2], ET-BERT [17], NetGPT [18]) и специализированные архитектуры Network Transformer для обнаружения аномального поведения устройств и сервисов [4]. Их ключевое достоинство – возможность использовать большие объемы неразмеченного трафика и переносить знания на задачи с малым числом меток.

Для Industrial Ethernet принципиально важно сохранить протокольный смысл сообщений. Поэтому перспективным направлением является протокольно-осмысленная токенизация, при которой признаки выбираются так, чтобы отражать роли сообщений и их контекст (коды функций, типы сервисов, размеры, параметры, временны́е характеристики). Предлагаемый в статье подход следует этой линии: формируются семантические токены на основе полей Modbus/TCP и OPC UA, затем они агрегируются в сессионные последовательности, после чего обучаются трансформерные модели для выявления аномалий в малых промышленных сетях.

Цель данной статьи – описать метод представления промышленного сетевого трафика и кодирования протокольной семантики, который позволяет эффективно применять трансформерные модели в малых сетях Industrial Ethernet. Под «малыми» понимаются локальные промышленные сегменты с ограниченным числом узлов и предсказуемым профилем трафика, например сеть цехового контроллера и нескольких программируемых логических контроллеров. В таких условиях особенно важна способность модели учитывать регулярные шаблоны обмена и выявлять даже скрытные отклонения.

В работе рассматриваются:

– способы потокового и сессионного агрегирования пакетов (раздел 2);

– характеристики промышленных протоколов Modbus/TCP и OPC UA и вытекающие из них требования к представлению данных (раздел 3);

– метод семантического кодирования трафика и формализация представления пакетов и их последовательностей как токенизированных данных (разделы 4–5);

– варианты реализации эмбеддингов признаков и позиционного кодирования (раздел 6);

– аргументы в пользу использования трансформерной архитектуры по сравнению с RNN / CNN для анализа сетей управления (раздел 7);

– подходы self-supervised обучения на сетевых данных (раздел 8);

– экспериментальная оценка предложенного подхода на реальном наборе данных промышленного трафика (раздел 9).

2. Агрегирование трафика по потокам и сессиям

Промышленный сетевой трафик представляет собой последовательность отдельных пакетов, обмен которыми происходит между множеством устройств (контроллеры, датчики, приводы и т. д.). Для корректного анализа поведения системы важно не рассматривать каждый пакет изолированно, а агрегировать пакеты в логические группы, соответствующие коммуникационным потокам или сессиям. Под потоком обычно понимается совокупность пакетов с общими атрибутами низкого уровня, например одинаковыми IP-адресами отправителя и получателя, номерами портов и т. п., в пределах определенного тайм-аута. Сессия может включать один или несколько потоков и отражать законченную операцию или диалог между узлами, например выполнение команды чтения регистра программируемого логического контроллера и передачу ответа. Агрегирование на уровне сессий позволяет восстановить контекст: какие запросы и ответы составляют единый обмен, каков временно́й интервал между ними и какова последовательность действий.

В промышленных сетях часто наблюдается цикличность и периодичность трафика: например, контроллер опрашивает датчики через фиксированные интервалы, обновляет значения, записывает команды на исполнительные устройства. Такой регулярный фон образует нормальный профиль сессий. При агрегации пакетов в потоки эти повторяющиеся коммуникации проявляются как схожие последовательности пакетов. Любое отклонение (новая последовательность команд, изменение длины или частоты пакетов, незапланированный запрос) может указывать на аномалию или атаку. Таким образом, потоково-сессионное представление повышает информативность данных для алгоритма обнаружения – вместо разрозненных пакетов модель видит целостную картину взаимодействия.

Формирование сессий из исходных дампов трафика осуществлялось с помощью анализа заголовков и временны́х меток пакетов. В данной работе под сессией понимается одно устойчивое соединение между парой узлов по определенному протоколу верхнего уровня. Для протокола Modbus/TCP это может соответствовать одному TCP-соединению, в рамках которого происходит серия запросов и ответов Modbus. Для OPC UA сессия начинается с установки соединения и инициализации сессионного контекста, включающей обмен HEL / ACK и открытие сессии, после чего следуют операционные сообщения, а завершается она закрытием сессии. Если соединение является длительным и включает множество циклов обмена, его можно разбить на подсессии по дополнительным признакам, например по временно́му окну или по окончанию логической транзакции. В общем случае критерии агрегирования задаются так, чтобы каждая сессия представляла логически завершенную последовательность взаимодействий.

Подход с использованием потоков и сессий принят в ряде современных наборов данных для IDS ICS. Например, в наборе данных ICS-Flow представлены как исходные сетевые пакеты, так и агрегированные записи потоков, что облегчает применение методов машинного обучения [1]. В разделе 9 используется именно такое агрегирование: исходные файлы трасс сетевого трафика в формате PCAP предварительно обработаны утилитами, выделяющими отдельные сессии обмена. Это позволяет далее токенизировать каждую сессию как самостоятельную последовательность (см. раздел 5). Таким образом, агрегирование трафика служит первым этапом предварительной обработки данных перед подачей в модель на этапе обучения или обнаружения.

3. Протокольные особенности Modbus/TCP и OPC UA

Промышленные сети используют специализированные протоколы прикладного уровня, которые накладывают определенную структуру на передаваемые данные. Рассмотрим два характерных протокола: Modbus/TCP – один из традиционных промышленных протоколов полевого уровня, и OPC UA – современный стандарт интеграции в промышленных системах.

Modbus/TCP

Modbus – один из старейших и наиболее распространенных промышленных протоколов, изначально разработанный в 1979 г. для связи между контроллерами, датчиками и исполнительными устройствами. Изначально данный протокол работал поверх последовательных линий (Modbus RTU по RS-485), однако вариант Modbus/TCP инкапсулирует Modbus PDU (*аббр. от англ*. Protocol Data Unit) в пакеты TCP/IP, что позволяет использовать его в Ethernet-сетях без изменений в логике протокола. Modbus имеет простую структуру «запрос–ответ»: ведущий узел, например программируемый логический контроллер или SCADA-система, отправляет запрос ведомому устройству с указанием кода функции (чтение, запись и т. п.) и необходимых параметров (адрес регистра, число регистров, данные), на что получает ответ, содержащий запрошенные данные или подтверждение записи. Каждый Modbus PDU имеет фиксированный формат: заголовок (транзакционный идентификатор, идентификатор протокола, длина, адрес устройства) и последующий блок данных приложения с кодом функции и байтами данных. Пример: запрос чтения регистра 0×1000 содержит функцию 0×03 (Read Holding Registers), адрес 0×1000 и количество 1; ответ содержит функцию 0×03, байт количества данных и затем значение регистра.

Особенности Modbus-трафика состоят в том, что это короткие сообщения фиксированного формата, которые чаще всего передаются в открытом виде, без шифрования. Modbus не имеет встроенных механизмов аутентификации, шифрования или контроля целостности, поэтому любой узел в сети, получивший пакет, может его прочитать или имитировать ответ. В нормальном режиме Modbus-трафик весьма регулярен: одни и те же запросы выполняются циклически, а значения могут повторяться или меняться плавно. Объем данных в одном пакете, как правило, невелик – несколько байтов полезной нагрузки. Временные задержки между запросом и ответом минимальны; для проводного Ethernet время реакции обычно составляет порядка 20 мс. С точки зрения представления для методов машинного обучения каждый Modbus-пакет несет семантически значимые поля – код функции, адрес, значение (для ответов), длину и др., – которые желательно использовать при токенизации (см. раздел 4). В то же время отсутствие шифрования позволяет анализировать содержимое напрямую, что упрощает анализ, но создает риски безопасности.

OPC UA

OPC Unified Architecture – современный протокол, разработанный OPC Foundation в 2006 г. как преемник устаревшего OPC DA (OLE for Process Control на базе DCOM). OPC UA спроектирован как универсальный механизм обмена данными между разнородными компонентами промышленной системы – от полевого уровня до облачных сервисов – с поддержкой богатой семантической модели данных и встроенных средств безопасности. OPC UA не привязан к конкретному транспорту: спецификация определяет несколько вариантов привязки к транспортной среде (TCP, HTTP/HTTPS, WebSockets и др.), однако наиболее распространен бинарный

протокол OPC UA поверх TCP. Коммуникационная модель OPC UA является клиент-серверной: клиенты, например SCADA-система или приложение мониторинга, устанавливают сессии с серверами, например OPC-сервером на контроллере, запрашивают или обновляют данные либо подписываются на изменения.

Особенности OPC UA-трафика состоят в следующем: сессия OPC UA начинается с рукопожатия, при котором клиент отправляет сообщение HELLO (HEL) с предложением параметров соединения, а сервер отвечает ACKNOWLEDGE (ACK). Далее происходит открытие безопасного канала – обмен сообщениями OPEN (OPN) с установлением уникального идентификатора сессии и согласованием режима безопасности (шифрование / подпись). После этого клиент создает сессию верхнего уровня в логике OPC UA, получает SessionID и может выполнять сервисные запросы: чтение атрибутов узлов адресного пространства, запись значений, подписку на изменения, вызов методов и т. п. Каждый такой запрос оформляется как сообщение MESSAGE (MSG) определенного типа сервиса, может включать сложно структурированные данные, например массивы и структуры, и получает соответствующий отклик MSG. В конце взаимодействия сессия завершается обменом CLOSE (CLO) и разрывом соединения.

Важное достоинство OPC UA заключается в наличии встроенных средств безопасности. При установлении сессии могут использоваться сертификаты для аутентификации клиента и сервера, а весь трафик сессии шифруется, если выбран режим SecureChannel с шифрованием. Это усложняет анализ содержимого трафика, так как полезная нагрузка становится недоступной для инспекции без ключей. Тем не менее даже зашифрованный OPC UA-трафик имеет отличительные характеристики на уровне метаданных: типы сообщений (HEL, OPN, MSG и др.), размеры пакетов, временны́е интервалы между сообщениями и последовательность запросов. Эти характеристики можно использовать для построения последовательности токенов, представляющих сессию, не раскрывая содержимого, например токенизировать типы сообщений и размеры. Если же трафик доступен в открытом виде, например в тестовой среде без шифрования или на уровне до шифрования, семантическое содержание OPC UA становится еще богаче: можно извлекать сами запрашиваемые узлы, методы и значения. В данной работе предполагается возможность токенизации как открытых протокольных полей для Modbus, так и ряда характеристик OPC UA. Например, учитывается последовательность типов сообщений OPC UA в сессии (HEL → OPN → несколько MSG → CLO), типы сервисов (Read, Write, Subscribe и т. д.) и базовые показатели объема данных.

Сравнивая Modbus/TCP и OPC UA, отметим следующие моменты, влияющие на выбор представления данных:

– длина и сложность сообщений (Modbus-пакеты короткие и фиксированные, пакеты OPC UA могут быть значительно больше и вариативнее, особенно при передаче массивов данных или сертификатов);

– структурированность данных (Modbus несет простые регистровые данные, например, числа, флаги, OPC UA – объектно-ориентированные: узлы адресного пространства с атрибутами, разными типами);

– частота обмена (опросы Modbus часто являются периодическими и высокочастотными, тогда как OPC UA может работать как в режиме опроса, так и по подписке, что формирует иной паттерн трафика);

– безопасность (OPC UA с шифрованием требует ориентироваться на метаданные – тайминги, размеры, вместо содержимого при анализе; Modbus – прямой анализ содержимого, но с риском подмены);

– транспорт (оба работают поверх TCP (в рассматриваемом в данной работе случае), потому с точки зрения сетевых потоков поведение сходно – есть установленное соединение, обмен парными сообщениями «запрос–ответ», хотя OPC UA может отправлять и асинхронные уведомления по подписке).

Учет всех этих особенностей диктует, что единый подход к кодированию трафика должен быть достаточно гибким, чтобы извлечь полезные признаки из различных протоколов. В следующем разделе описывается общий метод семантического представления, который может быть специализирован под конкретные протоколы путем настройки набора токенов и признаков.

4. Семантическое кодирование сетевого трафика

Непосредственное использование трафика в форме последовательности байтов, поступающей на сетевой интерфейс, неэффективно для обучения нейронных сетей, особенно таких сложных моделей, как трансформеры. Необходимо преобразовать исходные данные в форму, которая, с одной стороны, пригодна для подачи в модель в виде числовых векторов, а с другой – сохраняет как можно больше семантически значимой информации о передаваемых сообщениях. Семантическое кодирование трафика – это процесс преобразования пакетов и потоков в последовательности токенов, отражающих структурные поля и значения, существенные для конкретного протокола.

Подход, предлагаемый в данной работе, опирается на аналогию с обработкой естественного языка: сетевой обмен рассматривается как «текст», состоящий не из слов, а из токенов протокольных

сообщений. Каждый токен соответствует либо определенному полю / значению в пакете, либо определенному атрибуту коммуникации. Ключевая идея – вместо неструктурированной совокупности числовых признаков (длина, байты и пр.) предоставить трансформеру дискретные метки, которые имеют прямой смысл в контексте протокола. Такой подход успешно применялся, например, для трафика интернета вещей (IoT, *аббр. от англ.* Internet of Things): разработаны процедуры кастомной токенизации, которые превращают статистические характеристики потока в семантически осмысленные пары «признак–значение» [3]. Авторы адаптируют эту идею к пакетно-сессионному представлению.

Принципы выделения токенов

Идентификаторы протоколов и сессий. Каждый поток принадлежит определенному протоколу (Modbus или OPC UA в рассматриваемом случае). Имеет смысл вводить токен, обозначающий начало сессии конкретного протокола, например <MBUS> для начала сессии Modbus, <OPCUA> – для OPC UA. Это играет роль своеобразного «тега контекста».

Тип сообщения. Для Modbus все операционные сообщения – запросы или ответы, определяются кодом функции. Поэтому для Modbus целесообразно вводить токены вида Func:XX (где XX – номер функции). Например, Func:03 для функции чтения регистров, Func:16 для записи нескольких регистров и т. д. В OPC UA – токены для типов верхнеуровневых сервисов: Service:Read, Service:Write, Service:Subscribe и т. п., а также отдельные токены для системных сообщений HEL, OPN, CLO. Благодаря этому последовательность токенов сразу фиксирует последовательность выполняемых операций.

Параметры запросов. В Modbus ключевым параметром является адрес регистра и количество регистров, а для записи – еще и сами данные. Можно кодировать адрес как отдельный токен, например Addr:4096 (для десятичного адреса 4096) или категоризовать диапазоны адресов (например, <CoilAddr>, <InputRegAddr> и т. д.), если известно назначение диапазона. Данные (значения регистров) могут быть любыми числами; их можно дискретизовать (например, токены <Value:0>, <Value:NonZero> для флагов, или диапазонные для аналоговых величин) либо вообще опускать из токенизации, если численные значения малоинформативны сами по себе для выявления аномалий. В OPC UA параметры разнообразны – идентификаторы узлов, запрашиваемые атрибуты, значения. Можно использовать упрощенный подход: токенизировать, к примеру, размер передаваемых данных (<Payload:Small> или <Payload:Large>), или тип узла (Node:Temperature, Node:Pressure и т. п. по семантике тега, если известна). В общем случае, для OPC UA с зашифрованным содержимым, внутренняя семантика скрыта – тогда токенизируются лишь внешние признаки (тип сервиса, длина).

Структурные метки. Помимо собственно содержимого сообщений, важны структурные моменты: начало и конец сессии. Можно вводить специальные токены, аналогичные start-of-sequence и end-of-sequence в NLP. Например, <START_SESSION> и <END_SESSION>. Они помогут модели понимать границы последовательности, особенно если модель обучается на непрерывном потоковом трафике без заранее заданных границ сессий.

Временные интервалы. Хотя трансформер в базовой версии не учитывает неравномерность времени между токенами, поскольку оперирует позициями в последовательности, для сетевого трафика это существенный фактор. Можно встраивать дополнительные токены или метки, отражающие задержки, например <Delay:100ms>, если между пакетами i и $i + 1$ прошло 100 мс. В нашем случае промышленные сети малого масштаба обычно характеризуются регулярным опросом, поэтому значительные отклонения по времени сами по себе могут указывать на проблему. Однако для упрощения базовой модели авторы не стали явно токенизировать время: оно вводится косвенно через позиционное кодирование (см. раздел 6) или может быть добавлено отдельным каналом признаков.

Таким образом, сессия преобразуется в последовательность токенов различного типа. Упрощенно Modbus-сессия чтения нескольких регистров может быть представлена как: <START_SESSION>, <MBUS>, Func:03, Addr:4096, <RESP>, Func:03, DataLen:2, <END_SESSION>. Здесь <RESP> – условный токен, обозначающий переход от запроса к ответу (можно иначе кодировать, например, включая признак «ответ» в токен функции). Этот пример иллюстрирует, что даже короткий диалог разбивается на ряд осмысленных элементов. Для OPC UA начальная сессия может выглядеть следующим образом: <START_SESSION>, <OPCUA>, HEL, ACK, OPN, OPN, Service:Read, Service:Read, Service:Write, CLO, <END_SESSION> (представлена последовательность этапов и операций).

Подход семантической токенизации позволяет вручную инкорпорировать знания о протоколе в представление данных, что облегчает последующему обучению выявление аномалий. Например, если злоумышленник выполняет нетипичную команду (скажем, функцию 0×05 Force Coil в Modbus, редко используемую в нормальной работе), то в последовательности появится соответствующий токен Func:05 – модель может распознать его как аномальный токен последовательности. Или если атака приводит к рассинхронизации сессии OPC UA (неправильная последовательность OPN / HEL сообщений), то возникнет нелогичный порядок токенов, который трансформер также может уловить, обучаясь на нормальных данных.

Важно подчеркнуть, что семантическое кодирование должно быть разработано для каждого протокола (или класса протоколов) с учетом специфики. Данное исследование сфокусировано на Modbus и OPC UA, но общая методика применима и к другим промышленным протоколам (DNP3, IEC-104, S7 и др.) – для каждого можно определить набор значимых токенов (команды, коды функций, типы сообщений, параметры), которые затем включить в общий «словарь» модели.

5. Формализация представления пакетов и потоков как токенизированных последовательностей

Для строгого описания процедуры кодирования введем некоторый формализм. Пусть имеется сессия S, состоящая из последовательности n сетевых сообщений (пакетов) $S = \{P_1, P_2, \ldots, P_n\}$, упорядоченных по времени. Каждый пакет P_i характеризуется набором атрибутов или полей $\{x_{i1}, x_{i2}, \ldots, x_{im}\}$, где m зависит от протокола и уровня детализации (например, для Modbus можно выделить поля: тип сообщения («запрос–ответ»), код функции, адрес, значение, длина и т. д., для OPC UA: тип сервиса, флаг безопасности, размер полезных данных и т. п.).

Для целей токенизации определим функцию T, отображающую значение поля в дискретный токен:

$$T(x_{ij}) = t_{ij},\ t_{ij} \in V,$$

где V – множество всех токенов (словарь) для рассматриваемых протоколов; например, если x_{ij} – это код функции Modbus со значением 3, то:

$$T(x_{ij}) = t_{\text{Func:03}},$$

а если x_{ij} – индикатор типа сообщения OPC UA со значением «HEL», то:

$$T(x_{ij}) = t_{\text{HEL}}.$$

Каждому пакету P_i может соответствовать не один, а последовательность токенов. В общем случае определим набор токенов для пакета P_i как объединение токенов по всем его полям:

$$t(P_i) = [T(x_{i1}), T(x_{i2}), \ldots, T(x_{im})],$$

где квадратные скобки обозначают упорядоченный список токенов.

Таким образом, пакет с множеством полей разворачивается в последовательность токенов. Если пакет содержит поля, которые решено не токенизировать, например числовое значение регистра, обрабатываемое как непрерывный признак, то такие поля можно исключить из $t(P_i)$ либо заменить обобщенным токеном, например <VAL> без указания конкретного значения.

После этого для всей сессии S получим суммарную последовательность токенов путем конкатенации токенов всех пакетов по порядку:

$$T(S) = [t(P_1);\ t(P_2);\ \ldots;\ t(P_n)],$$

где точка с запятой обозначает конкатенацию списков; $T(S)$ – это конечное дискретное представление всей сессии, готовое для подачи в трансформерную модель.

Длиной получившейся последовательности будет сумма токенов по всем пакетам. В худшем случае (если каждый пакет порождает несколько токенов) длина может быть значительно больше числа пакетов. Чтобы ограничить размер последовательностей (что важно для уменьшения вычислительной сложности трансформера), можно использовать два подхода:

1) сократить максимальное число пакетов n в сессии, например рассматривать не более 100 сообщений, усекая или объединяя слишком длинные сессии;

2) ограничить число токенов на пакет (например, токенизировать только ключевые поля, игнорируя второстепенные).

В ходе проведения экспериментов был установлен предел длины последовательности в 128 токенов, который оказался достаточным для типичных сессий в целевом наборе данных. Длинные последовательности, встречавшиеся редко, в основном при длительных OPC UA-сессиях укорачивались: авторы отбрасывали самые ранние части сессии, полагая, что для обнаружения актуальных атак важнее последние события. Альтернативно такие последовательности можно было бы разбивать на несколько сессий.

После токенизации необходимо преобразовать дискретные токены $T(S)$ в числовые векторы для подачи в нейросеть.

Введем отображение E (эмбеддинг токенов):

$$V \to \mathbb{R}^d,$$

которое каждому токену $t \in V$ сопоставляет обучаемый вектор:

$$z = E(t) \in \mathbb{R}^d,$$

где d – размерность эмбеддинга.

В дальнейшем t_l обозначает l-й токен в последовательности $T(S)$, а z_l – его векторное представление.

Применяя эмбеддинг ко всей последовательности $T(S)$, получаем последовательность векторов:

$$E(T(S)) = [z_1, z_2, \ldots, z_L],$$

где $L = |T(S)|$ – длина токенизированной последовательности (число токенов в сессии); z_l – d-мерный эмбеддинг l-го токена:

$$z_l = E(t_l) \in \mathbb{R}^d.$$

Полученная последовательность $Z = [z_1, z_2, \ldots, z_L]$ далее дополняется позиционной информацией и подается на вход трансформерной модели.

Дополнительно введем функцию позиционного кодирования P: $\{1,\ldots,L\} \to \mathbb{R}^d$, которая каждой позиции l в последовательности сопоставляет позиционный вектор $p_l = P(l)$.

Тогда окончательный вход в трансформер формируется как сумма эмбеддинга токена и позиционного вектора:

$$h_l = z_l + P(l),$$
$$H = [h_1, h_2, \ldots, h_L],$$

где h_l – итоговый вектор на позиции l (эмбеддинг токена с учетом позиции).

Последовательность $H = [h_1, h_2, \ldots, h_L]$ используется как вход для последующих слоев трансформера.

6. Эмбеддинги и позиционное кодирование

6.1. Эмбеддинг токенов

После токенизации (раздел 4) и формальной укладки последовательности (раздел 5) каждый токен представляет собой некоторый символ из словаря V. Используется стандартный подход, принятый в NLP-моделях: каждому уникальному токену соответствует обучаемый параметр-вектор размерности d (гиперпараметр модели). Матрица эмбеддингов E является частью параметров модели, инициализируется случайно (либо заимствуется из предварительно обученной модели, если таковая имеется) и оптимизируется в процессе обучения трансформера.

Размерность эмбеддинга d выбирается с учетом баланса между выразительностью и сложностью модели. В проведенных экспериментах использовались значения $d = 64$ или $d = 128$; этого достаточно для кодирования нескольких десятков или сотен возможных токенов без перенасыщения модели избыточными параметрами. Некоторые токены могут не нести большого объема информации, например <START_SESSION> или <END_SESSION>, однако они все равно получают собственные векторы, которые модель может настроить по необходимости.

Особенность промышленного трафика – потенциально ограниченный размер словаря токенов по сравнению, скажем, с естественным языком. Если рассматривать только Modbus и OPC UA, общее число видов токенов обычно составляет порядка сотен: все возможные коды функций Modbus (чуть более 10 часто используемых), несколько категорий адресов / данных; типы сервисов OPC UA (десятки) и системные сообщения; а также технические токены начала / конца сессии, задержек и т. п. В итоге $|V|$ может оказаться <200. Это означает, что даже случайная инициализация эмбеддингов при обучении достаточно быстро начнет разделять токены. В отличие от слов естественного языка, где смысл часто проявляется в геометрии пространства эмбеддингов, здесь скорее ожидается, что модель сама соотнесет токены с их полезностью для задачи (например, токен опасной команды может получить вектор, который повышает отклик на выходном слое «аномалия»). Теоретически можно заложить дополнительную структуру, используя раздельные подэмбеддинги для разных типов токенов (например, отдельные параметры для Func:XX и для адресов / параметров), а затем объединять их в общий вектор. Однако такой ручной дизайн усложняет модель и не всегда дает выигрыш. Поэтому в данной работе используется единое пространство эмбеддингов для всех токенов, полагаясь на способность модели различать их тип по контексту (например, токены Func:XX обычно встречаются в характерных позициях сообщения, что может быть уловлено позиционным кодированием).

6.2 Позиционное кодирование

В архитектуре трансформера базовый механизм самовнимания сам по себе не учитывает порядок следования токенов, если явно не добавить позиционную информацию [3]. Классический подход состоит в добавлении к каждому эмбеддингу токена позиционного вектора $p_l = P(l)$, где $l = 1, \ldots, L$ – индекс позиции токена в последовательности, а L – ее длина. Вариант $P(l)$ может быть детерминированным, например задаваться синусоидальными функциями разной частоты, как в работе [15], либо обучаемым. Для сетевых данных позиция в последовательности соответствует порядку поступления пакетов по времени; интервалы могут быть неравномерными, однако в рамках выбранного представления кодируется именно порядковый номер появления токена.

Использовались два варианта позиционных кодов.

Вариант 1. Синусоидальное позиционное кодирование задается формулами:

$$P(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{\frac{2i}{d}}}\right),$$
$$P(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{10000^{\frac{2i}{d}}}\right).$$

где pos – позиция токена в последовательности (pos = 1, …, L), $i = 0, \ldots, d/2 - 1$; d – размерность эмбеддинга (d_{model}).

Вариант 2. Обучаемое кодирование позиций: в этом варианте вектор позиции, также как и эмбеддинг токена, является обучаемым параметром.

Для этого вводится таблица позиций:

$$P \in \mathbb{R}^{L\_max \times d},$$

где L_{max} – максимальная длина последовательности.

Вектор p_l берется из этой таблицы и оптимизируется в ходе обучения. Подход более гибок, однако требует осторожности при генерализации на длины, превышающие L_{max} или редко встречающиеся при обучении.

В реализации было выбрано обучаемое позиционное кодирование, так как максимальная длина известна и относительно невелика: L_{max} = 128. Это позволило модели потенциально скорректировать представление позиций с учетом специфики данных (например, токены запроса и ответа часто идут парой – тогда позиционные векторы для позиций l и l + 1 могут быть сделаны более похожими). При этом классическое синусоидальное кодирование также показало работоспособность в наших экспериментах, но с небольшим снижением качества. Таким образом, выбор способа позиционного кодирования не является критичным, однако обучаемый вариант дал немного лучшие результаты, поэтому далее используем именно его.

Отметим интересный момент: комбинирование нескольких признаковых пространств. Можно было бы вводить позиционные признаки не только для линейной позиции, но и, например, кодировать номер пакета в сессии отдельно от позиции токена внутри пакета. Или ввести двумерное кодирование: индекс сообщения и индекс токена в сообщении. В теории это повысило бы способность модели различать начало нового пакета даже без специального токена <RESP> или аналогичных. Были проведены эксперименты с такими схемами, однако существенного выигрыша не было получено – вероятно потому, что модель и так распознавала паттерны вроде «токен функции всегда первый после <MBUS>». Тем не менее, для более сложных протоколов или смешанного трафика этот прием может пригодиться.

Итак, на выходе этапа эмбеддинга и позиционного кодирования для каждой сессии формируется последовательность векторов $H = [h_1, h_2, \ldots, h_L]$, готовая к подаче в блоки трансформера. Далее опишем архитектуру модели и обоснуем, почему именно она предпочтительна в рассматриваемом случае.

7. Преимущества применения трансформерной архитектуры для анализа трафика

Трансформерная архитектура в последние годы стала фактическим стандартом в задачах обработки последовательностей, во многих случаях вытеснив RNN. Рассмотрим, какие преимущества она дает применительно к анализу промышленного трафика и почему в данной работе сделан выбор в ее пользу.

1) Учет долгосрочных зависимостей. В промышленных протоколах аномальное поведение может проявляться как последовательность нескольких событий, разнесенных по времени. Например, атака повторного воспроизведения сначала перехватывает пакеты, а спустя некоторое время повторяет их; кроме того, нарушитель может последовательно посылать несколько команд в нетипичном для штатной работы порядке. RNN с ограниченной памятью может «забыть» о первом событии, когда дойдет до пятого, особенно если между ними прошли десятки других пакетов. Трансформер благодаря механизму самовнимания способен напрямую связывать представления удаленных по позиции токенов: каждый выходной признак рассчитывается как взвешенная сумма всех входных, а веса отражают внимание к соответствующим токенам [4].

Формально самовнимание в одной голове механизма внимания вычисляется как:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q\,K^T}{\sqrt{d_k}}\right)V,$$

где Q, K, V – матрицы запросов (Queries), ключей (Keys) и значений (Values), полученные линейными преобразованиями исходных эмбеддингов; d_k – размерность пространства запросов и ключей.

Вышепредставленное выражение показывает, что для каждого токена формируется взвешенная сумма всех входных векторов V, а веса вычисляются (через Softmax) на основе сходства соответствующих запросов и ключей $Q\,K^T$, нормированного на $\sqrt{d_k}$. Таким образом, если, например, текущий токен – Func:05 (команда принудительной установки выхода), модель может «обратить внимание» на предыдущие токены (например, Func:03 – штатный опрос регистра), сравнить контекст и обнаружить несоответствие последовательности. В RNN подобные зависимости также могут учитываться через скрытое состояние, но с ростом расстояния по последовательности влияние ранних событий обычно ослабевает. В трансформере связь между любыми позициями поддерживается напрямую механизмом внимания.

2) Параллелизм и эффективность обучения. При обучении на больших объемах данных трансформеры могут обрабатывать последовательность целиком параллельно в одном временно́м шаге (self-attention вычисляется матрично), в то время как RNN приходится прогонять пакет за пакетом последовательно. Хотя скорость работы не всегда критична для офлайн-анализа, при тонкой настройке модели на большом датасете (например, гигабайты трафика) параллельное обучение позволяет эффективнее использовать GPU. Кроме того, параллельность облегчает реализацию предварительно обученных методов типа MLM (*аббр. от англ.*

Masked Language Modeling), где можно одновременно маскировать и предсказывать несколько позиций.

3) Гибкость архитектуры и возможность дообучения. Трансформерные модели легко масштабируются: можно изменять количество слоев, количество голов самовнимания и размер эмбеддингов, не меняя принципиально структуру данных. Они также хорошо подходят для переноса знаний: модель, обученная на одном наборе трафика, может быть дообучена на другом или использоваться как инициализация. В контексте ICS это особенно ценно, поскольку трафик одного промышленного объекта может служить основой для предварительного обучения модели, которая затем адаптируется к сети другого предприятия. RNN и CNN также допускают дообучение, однако в трансформерах особенно просто реализовать механизмы дополнительного дообучения последних слоев, заморозки части параметров и т. д. с учетом модульности модели (энкодер / декодер или только энкодер). В наших экспериментах предварительно обучался трансформерный энкодер на задаче реконструкции пакетов, то есть режиме self-supervised обучения (см. раздел 8), а затем изменялся выходной слой и выполнялось дообучение на задаче классификации. При аналогичном подходе модели RNN демонстрировали более медленную и менее устойчивую сходимость.

4) Обобщающая способность. Благодаря своему устройству трансформер способен выявлять более общие закономерности, а не подгоняться под частный порядок событий. Например, CNN хорошо распознает паттерн вида «*A*, затем *B*», если он всегда находится в окне из пары пакетов, однако, если между *A* и *B* иногда появляются другие события, паттерн разрушается. RNN, как отмечалось выше, склонны забывать далекие зависимости. Трансформер же может обрабатывать и последовательности вида «*A*, ..., *B*», и «*A*, *B*» сходным образом, если он выучил, что важна связь между *A* и *B* независимо от расстояния. Это обеспечивает лучшую переносимость между разными режимами работы сети. Практические исследования показывают, что трансформеры способны эффективнее обобщать ранее не наблюдавшиеся варианты трафика после достаточного предобучения [3]. В частности, отмечается, что сочетание трансформера с продуманной токенизацией позволяет достигать высокой точности на различных наборах IoT-трафика без переобучения под каждый из них, что указывает на более высокую обобщающую способность по сравнению с классическими методами.

5) Интерпретируемость (относительная). Несмотря на репутацию «черных ящиков», трансформеры предоставляют механизм внимания, который можно визуализировать для интерпретации результатов. В области сетевой безопасности появились работы, где на основе матриц внимания выясняется, на какие именно токены, то есть поля пакетов, модель ориентируется при выявлении аномалии [4]. Это может помочь эксперту понять природу обнаруженного отклонения: например, если внимание сосредоточено на токене функции Modbus, подозрительной является именно команда. RNN и CNN в этом отношении менее удобны, поскольку в них сложнее выделить, какие именно части последовательности повлияли на решение. Хотя интерпретация трансформеров остается активной областью исследований и не всегда дает однозначные выводы, ее потенциал в IDS заключается в том, что модель может не только сигнализировать о срабатывании, но и указывать, какие сообщения вызвали подозрение.

6) Совместимость с self-supervised обучением. Об этом подробнее сказано в следующем разделе; здесь лишь отметим, что трансформеры, особенно энкодеры семейства BERT, изначально разрабатывались с расчетом на предварительное обучение на маскированном моделировании. Это делает их естественным выбором, если требуется использовать неразмеченные дампы трафика для обучения представлений. RNN тоже можно предварительно обучать, однако, например, маскирование части входов при последовательном чтении оказывается менее эффективным, чем в трансформере, где модель оценивает маскированные позиции на основе контекста с двух сторон. Таким образом, архитектура трансформера функционально лучше приспособлена к таким приемам.

Разумеется, у трансформеров есть и недостатки – главным образом, требовательность к вычислительным ресурсам при росте длины последовательности L, поскольку вычислительная сложность механизма самовнимания составляет $O(L^2)$. В контексте малых сетей Industrial Ethernet эта проблема не столь остра: как правило, сессии не содержат тысяч сообщений, а предлагаемый подход с агрегацией и ограничением длины последовательности удерживает L в пределах сотен. Кроме того, можно применять техники ограничения внимания, например локальное внимание или иерархическое агрегирование, как показано в работе [22]. В предлагаемом прототипе модели без таких усложнений удалось обойтись.

В итоге, сопоставив возможности различных архитектур, авторы пришли к выводу, что трансформер предпочтительнее RNN / CNN для рассматриваемой задачи, поскольку способен обрабатывать семантически токенизированный трафик с учетом произвольных зависимостей и эффективно использовать предварительное обучение. Далее описывается, как именно применяется self-supervised обучение для нашей модели и какие результаты оно дает.

8. Self-supervised обучение на трафике

Как уже отмечалось, большая проблема применения глубокого обучения в промышленных сетях состоит в дефиците размеченных данных об атаках. Сбор полноценного набора данных, включающего разнообразные кибератаки на реальное промышленное оборудование, затруднен по соображениям безопасности и конфиденциальности, поэтому крайне важно уметь использовать имеющиеся большие объемы неразмеченного трафика нормальной работы для обучения модели. Self-supervised обучение предоставляет такой подход: формируется вспомогательная задача, которую модель учится решать, и тем самым вырабатывает полезные представления.

Авторы реализовали два варианта self-supervised обучения трансформерного энкодера на промышленном трафике: маскированное моделирование токенов (MTM) и предсказание следующего события (Next Event Prediction).

MTM представляет собой аналог метода MLM из NLP. Последовательность токенов $T(S)$ берется из нормального трафика; случайным образом выбирается часть токенов, например 15 %, которые заменяются специальным маркером <MASK>. Модель, то есть только энкодер, обучается предсказывать исходные токены в позициях <MASK> по контексту остальных элементов. Иными словами, выход трансформера проходит через линейный классификатор на словарь и обучается с функцией потерь кросс-энтропии по истинному токену на замаскированных позициях. Такой подход заставляет модель использовать контекст с обеих сторон маски. Например, если в нормальной последовательности присутствовали Func:03, Addr:4096, <RESP>, Func:03 и был замаскирован элемент Addr:4096, то модель должна по окружающим токенам предположить, что именно этот адрес наиболее вероятен, поскольку функция 03 в рассматриваемом наборе данных обычно обращается к определенным адресам. Обучаясь на множестве таких примеров, трансформерный энкодер формирует представление, в котором токены со сходным контекстом имеют близкие эмбеддинги и сходные активации.

Режим предсказания следующего события предполагает, что модель получает на вход последовательность токенов сессии без последнего пакета и должна предсказать следующий пакет или следующие токены. Этот режим похож на обучение языковой модели слева направо. Однако авторы сочли его менее удобным для решения поставленной задачи, поскольку последовательности обычно циклични и предсказание точного следующего сообщения затруднительно: несколько вариантов могут быть нормальными. Тем не менее была предпринята попытка обучать трансформер предсказывать не конкретный токен, а совокупность признаков, например тип следующего сообщения – команда или ответ. Качество таких предсказаний также улучшалось со временем, однако в итоге авторы сосредоточились на MTM как на более мощном методе.

После предварительного обучения на большом наборе исходных сессий, где использовался только нормальный трафик без атак, чтобы модель усвоила профиль штатной работы, трансформерный энкодер оказался способен генерировать обобщенные скрытые представления токенов. Эти представления были использованы для решения целевой задачи – обнаружения аномалий или классификации атак – с минимальным дополнительным обучением. В экспериментальной части авторы выполнили тонкую настройку: добавили выходной слой, например бинарный классификатор классов «атака» и «норма», поверх усредненных по времени активаций энкодера и провели дообучение на размеченной выборке, которая существенно меньше по объему, чем данные для предварительного обучения. Выяснилось, что такая стратегия заметно повышает качество: модель, прошедшая self-supervised этап, достигает примерно на 5–10 % более высокого значения F1, чем аналогичная архитектура, обученная без предварительного обучения.

Применение self-supervised обучения к сетевому трафику ICS уже опробовано в других работах. Например, в [4] предложен специальный Network Transformer, в котором использовались структурированные данные о топологии сети, а модель обучалась выявлять аномалии без меток, то есть, по сути, изучала норму и сигнализировала об отклонениях. Настоящее исследование ближе к подходам из NLP: используется маскирование токенов. Это позволяет интегрировать решение в существующие фреймворки трансформеров семейства BERT. Более того, потенциально можно использовать готовые веса крупных моделей, предварительно обученных на разных сетевых данных. В настоящее время появляются работы по большим языковым моделям для сетей; например, модель NetGPT [18] представляет собой трансформер, предварительно обученный генерировать сетевые пакеты. Потенциально такие модели можно адаптировать к конкретным протоколам ICS.

В рамках данной статьи авторы ограничились собственным предварительным обучением на индустриальном наборе данных. Технически обучение MTM было реализовано следующим образом: использовался корпус из примерно 100 тыс. сессий нормального трафика (из набора данных, см. раздел 9); случайным образом маскировалось 15 % токенов, после чего трансформерный энкодер с шестью слоями, четырьмя головами механизма внимания и размерностью $d = 128$ обучался предсказывать эти токены. Обучение продолжалось около пяти эпох до сходимости, когда значение кросс-эн-

тропии на масках переставало улучшаться. Затем веса энкодера использовались как инициализация для модели обнаружения атак.

Еще один вариант self-supervised обучения связан с автокодировщиком аномалий, когда модель, например трансформерный автокодировщик, учится воспроизводить входную последовательность на выходе, минимизируя ошибку реконструкции, а аномалии определяются по росту этой ошибки. В настоящем исследовании этот вариант не был реализован полностью, однако в ходе экспериментов простой трансформерный декодер обучался восстанавливать исходную последовательность токенов в задаче тождественного отображения. Эксперимент показал, что такая схема быстро переобучается, то есть фактически копирует вход на выход, поэтому для практической пользы необходимо ограничение пропускной способности скрытого пространства, чтобы нормальный трафик реконструировался хорошо, а аномальный – хуже. Эта проблема будет изучена авторами в дальнейшем.

Подводя итог, отметим, что self-supervised обучение заметно повышает качество и надежность трансформерной модели при анализе промышленного трафика. Оно позволяет использовать большие объемы нормальных данных для формирования представления о протокольной семантике и закономерностях без ручной разметки. В разделе 9 приводится сравнение моделей с предварительным обучением и без него, демонстрирующее выигрыш. В реальных условиях эксплуатации IDS для ICS это означает, что систему можно сначала обучить на обычном рабочем трафике предприятия, например на данных, собранных за несколько дней или недель, а затем использовать в режиме обнаружения с более высокой чувствительностью к отклонениям.

9. Экспериментальное исследование

Для экспериментальной оценки разработанных методов использовался набор данных реального промышленного трафика, включающий как штатное взаимодействие устройств, так и инсценированные атаки. В частности, был задействован общедоступный набор Malware in Smart Factory [19, 20]. Этот набор данных содержит около 173 ГБ исходных файлов сетевого трафика в формате PCAP, записанных за 16 дней работы небольшой промышленной сети с оборудованием числового программного управления, датчиками и программируемыми логическими контроллерами. В сети использовались современные промышленные протоколы, основными из которых были MQTT, OPC UA и Modbus/TCP. Часть трафика соответствует нормальной работе (около 14 дней), а в течение двух дней осуществлялись контролируемые атаки двух типов: агрессивные, то есть легко заметные, и скрытные, характеризующиеся минимальным отклонением от нормального поведения. Каждая сессия в наборе данных снабжена меткой, указывающей, относится ли она к какому-либо сценарию атаки или является легитимной. Тем самым данный набор хорошо подходит для обучения и тестирования рассматриваемых моделей, поскольку содержит как примеры нормальной работы, так и разнообразные атаки на промышленные протоколы.

Предобработка данных

Из всего объема данных авторы сосредоточились на сессиях, относящихся к протоколам Modbus/TCP и OPC UA, поскольку именно они являлись объектом семантического кодирования. MQTT-трафик, представлявший телеметрию, в данном исследовании не рассматривался. Для извлечения сессий использовался скрипт на основе Zeek [21]: сессия определялась по уникальной паре IP-адресов и портов, а также по протоколу. Для OPC UA дополнительно контролировались сообщения HEL / ACK как признак начала новой сессии. Всего было выделено порядка 1,2 млн нормальных сессий Modbus и около 50 тыс. нормальных сессий OPC UA, а также несколько тысяч сессий с атаками различных типов. Выборка оказалась несбалансированной: нормальных сессий значительно больше. Для предварительного обучения в self-supervised режиме была сформирована обучающая выборка, включавшая около 100 тыс. случайно выбранных нормальных сессий, примерно 90 тыс. Modbus и 10 тыс. OPC UA, что отражало частоту их появления. Отдельно был подготовлен набор для обучения детектора атак: 5000 нормальных сессий и все доступные 3500 сессий с атаками; около 20 % этого набора было отложено в тестовую выборку.

Реализация модели

Трансформерная модель была реализована на PyTorch с использованием архитектуры энкодера семейства BERT в базовой конфигурации, уменьшенной по числу параметров:

– количество слоев (блоков самовнимания) – 4;

– количество голов механизма внимания – 8;

– размер эмбеддинга и скрытого представления d – 128;

– размер промежуточного полносвязного слоя – 256;

– словарь токенов – 150 единиц (разработанный вручную набор для Modbus/TCP и OPC UA, как описано в разделе 4);

– максимальная длина последовательности токенов – 128 (большие сессии укорачивались).

Выходной слой для задачи обнаружения атак представлял собой полносвязный классификатор с функцией Softmax для двух классов: «атака» и «норма». Для обучения использовался оптимизатор AdamW; начальная скорость обучения составляла 5×10^{-4} с линейным спадом.

Режимы обучения:

– предварительное обучение: на 100 тыс. нормальных сессий решалась задача MTM (маскировалось 15 % случайных токенов; обучение продолжалось пять эпох, после чего значение перплексии на масках стабилизировалось примерно на уровне 1,2; то есть модель почти безошибочно предсказывала замаскированные токены);

– тонкая настройка: продолжение обучения модели, инициализированной весами после предварительного обучения, на задаче классификации с метками «атака» и «норма»; обучение выполнялось 10 эпох с контролем по F1-мере на валидационной выборке и остановкой при отсутствии улучшения в течение двух эпох;

– обучение без предварительного обучения: для сравнения та же архитектура обучалась на задаче классификации без предшествующего этапа self-supervised обучения.

Дополнительно для сопоставления были обучены более простые модели: LSTM и одномерная CNN. LSTM-модель имела два слоя по 128 нейронов; на вход ей подавались те же эмбеддинги токенов. Иными словами, для корректности сравнения LSTM обучалась на том же токенизированном представлении, а не на исходных признаках. CNN-модель включала четыре одномерных сверточных слоя с размером ядра 3 и 64 выходных канала в каждом слое; сеть обрабатывала последовательность токенов и выявляла локальные n-граммы. Эти модели обучались только на задаче классификации; предварительное обучение для них не проводилось, хотя, например, автокодировщик на основе LSTM теоретически возможен.

Результаты

На тестовой выборке, включавшей ранее не наблюдавшиеся сессии, в том числе 700 атакованных и 1000 нормальных, были получены показатели качества обнаружения атак, приведенные в таблице 1 (положительным считался класс «атака»).

Как видно из таблицы 1, предложенная трансформерная модель, предварительно обученная в self-supervised режиме, показывает наилучший результат: F1 = 99,3 %, что соответствует практически безошибочному выявлению атакующих сессий при минимуме ложных срабатываний. Для сравнения та же модель без предварительного обучения дает F1 около 96,3 %, то есть несколько уступает, хотя все равно превосходит RNN и CNN. Модели LSTM удалось достичь F1 около 93,4 %; ее основная потеря связана с более низкой точностью, то есть с большим числом ложных срабатываний, когда нормальные сессии ошибочно маркируются как атаки. CNN показала еще более низкий результат — F1 около 88,7 %; при этом чаще наблюдались и пропуски атак (полнота 88 %). Такой разрыв между трансформером и классическими моделями глубокого обучения согласуется с ожиданиями, поскольку трансформер лучше учитывает сложные последовательные шаблоны атак, в том числе скрытых.

Отдельно следует отметить пользу предварительного обучения: разница между 99,3 и 96,3 % в абсолютных величинах выглядит небольшой, однако на практике это означает, что трансформер без предварительного обучения либо пропускал около 3 % атак, либо давал дополнительные ложные тревоги, тогда как предварительно обученная модель допускала менее 1 % таких ошибок. В условиях промышленной эксплуатации IDS трехкратное снижение ложных тревог особенно важно, поскольку уменьшает нагрузку на оператора; вместе с тем пропуск даже нескольких процентов атак может оказаться критичным. Следовательно, self-supervised этап позволил модели тоньше различать норму и аномалию.

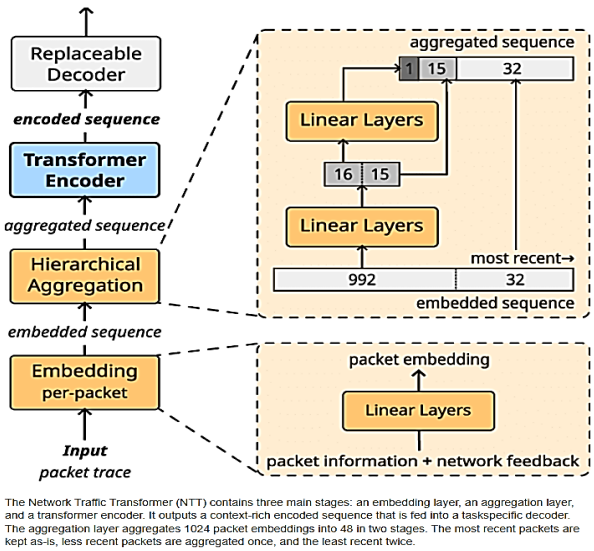
ТАБЛИЦА 1. Сравнение моделей по качеству классификации сессий («атака / норма»), %

TABLE 1. Comparison of Models by Session Classification Quality (Attack / Normal), %

Модель	Accuracy	Precision	Recall	F1-score
Трансформер (предобучение)	**99,3**	**98,9**	**99,7**	**99,3**
Трансформер (без предобучения)	96,8	95,5	97,1	96,3
LSTM	94,6	91,0	96,0	93,4
CNN	91,2	89,5	88,0	88,7

Примечание: Accuracy – точность; Precision – точность по атаке; Recall – полнота по атаке; F1-score – F1 по атаке

Для наглядности на рисунке 1 приведена схема трансформерной архитектуры для моделирования сетевого трафика (Network Traffic Transformer) [21].



The Network Traffic Transformer (NTT) contains three main stages: an embedding layer, an aggregation layer, and a transformer encoder. It outputs a context-rich encoded sequence that is fed into a taskspecific decoder. The aggregation layer aggregates 1024 packet embeddings into 48 in two stages. The most recent packets are kept as-is, less recent packets are aggregated once, and the least recent twice.

Рис. 1. Архитектура Network Traffic Transformer [21, рис. 3]

Fig. 1. Network Traffic Transformer architecture [22, Fig. 3]

В экспериментальной сети присутствовали узлы, работающие по протоколам Modbus и OPC UA, журналы которых анализировались. Конфигурация включала систему числового программного управления, выступавшую OPC UA-сервером, программируемый логический контроллер с поддержкой Modbus, датчики энергии и атакующий узел, локально подключенный для проведения тестов [19]. Эта схема (см. рисунок 1) помогла при интерпретации ряда атак: например, атака типа «man-in-the-middle» на Modbus моделировалась внедрением пакетов от имени программируемого логического контроллера, и трансформер успешно идентифицировал нарушение последовательности «запрос–ответ», в частности появление некорректного токена ответа после запроса и искажение временны́х меток.

Помимо приведенных результатов, также оценивалась обобщающая способность модели. Авторы пробовали дообучить трансформер, обученный на данном датасете, на другом наборе (для разнообразия – синтетическом датасете ICS-Flow [1], содержащем имитацию атак на водоканале). Модель быстро адаптировалась, показав, что предварительно выученные эмбеддинги Modbus довольно универсальны – она успешно нашла атаки, даже специфические, например, посылку некорректных команд управления насосом (что проявлялось в токене функции, выходящем за нормальный диапазон). RNN при переносе потребовал почти полного переобучения заново, т. е. его внутренние состояния оказались менее переносимыми.

Обсуждение результатов

Высокая точность трансформерного подхода объясняется несколькими факторами:

– семантическая токенизация предоставляет модели информацию в «сконцентрированном» виде; ей не пришлось самой выучивать, что, скажем, байт 0×03 в определенном контексте значит команду чтения – авторы это обозначили токеном; поэтому большая часть потенциала модели пошла непосредственно на выявление зависимостей между токенами (внимание между функциями, адресами, последовательностями сообщений);

– трансформер эффективно использовал контекст сессии: скрытные атаки, не изменявшие мгновенные статистики, например размер пакета, но изменявшие команду, выявлялись по несоответствию общей последовательности; ложных срабатываний почти не наблюдалось, что свидетельствует о хорошем усвоении профиля нормального трафика;

– LSTM, даже работая с токенами, вероятно теряла часть контекста: у нее имеется фиксированный вектор состояния, тогда как у трансформера несколько голов механизма внимания могли параллельно отслеживать разные аспекты — последовательность функций, временные интервалы и шаблон «запрос–ответ»;

– CNN в данной задаче показывает более слабые результаты, поскольку шаблоны аномалий не сводятся к коротким локальным подпоследовательностям и могут охватывать всю сессию.

Стоит упомянуть и вычислительную эффективность: на этапе применения на одной сессии трансформер с четырьмя слоями и восемью головами механизма внимания обрабатывал около 100 токенов за доли миллисекунды на современном CPU, что позволяет работать в онлайн-режиме с небольшой задержкой даже при сотнях сессий в секунду. LSTM была несколько быстрее на коротких последовательностях, но на длинных проигрывала из-за необходимости итеративной обработки по шагам.

Таким образом, эксперименты подтвердили, что предложенное представление промышленного трафика в виде токенов + трансформерная модель позволяет достичь высоких результатов в задаче обнаружения аномалий и атак. В заключении суммируем выводы и наметим направления дальнейших работ.

10. Заключение

В статье изложен подход к аналитической обработке трафика промышленных сетей с применением трансформерных моделей глубокого обучения.

Основные результаты и выводы работы можно сформулировать следующим образом.

1) Представление трафика

Разработан метод семантического представления индустриального трафика: пакеты агрегируются в сессии, после чего превращаются в последовательности токенов, отражающих протокольную семантику (коды команд, типы сообщений, параметры и др.). Формально определено отображение пакетов в токены и задан процесс эмбеддинга и позиционного кодирования последовательностей. Показано, что такой метод позволяет унифицировать разнородные протоколы (на примере Modbus/TCP и OPC UA) в одном формате входных данных для нейросети.

2) Трансформерная архитектура в сравнении с RNN / CNN

Обосновано преимущество трансформерной архитектуры для задач анализа сетевого трафика по сравнению с традиционными RNN и CNN: трансформер эффективно улавливает долгосрочные зависимости и сложные шаблоны в последовательностях токенов, что критично для выявления атак, маскирующихся под нормальное поведение. Экспериментально подтверждено, что трансформерная модель дает более высокую точность обнаружения аномалий (F1-мера до 99 %) и более низкий уровень ложных срабатываний, чем LSTM или CNN, при анализе тех же данных. Это соответствует мировой тенденции применения трансформеров в кибербезопасности [2, 3].

3) Self-supervised обучение

Впервые для рассматриваемой предметной области продемонстрировано успешное применение self-supervised обучения трансформера на исходных данных промышленного трафика. Маскирование токенов и их предсказание по контексту позволили модели выучить нормальные шаблоны протоколов без учителя. Дальнейшая тонкая настройка на небольшом количестве размеченных примеров атак дала существенный прирост качества. Такой подход особенно ценен для малых сетей Industrial Ethernet, где аномальные ситуации редки и трудно собрать статистически значимую обучающую выборку: модель может обучиться на большом объеме штатных данных, а затем распознавать редкие отклонения за счет накопленных знаний о норме [4].

4) Практическая применимость

Рассмотренный набор данных и сценарии атак близки к реальным условиям, поскольку атаки проводились в действующей тестовой промышленной сети с реальным оборудованием [19, 20]. Высокая точность, достигнутая моделью, позволяет предполагать, что подобные методы могут быть внедрены в системы кибербезопасности промышленных предприятий, например в виде модуля интеллектуального анализа трафика на промышленных шлюзах или в системах IDS, мониторящих технологические сети. Малые сети Industrial Ethernet, характеризующиеся относительно стабильным трафиком, особенно подходят для таких решений, поскольку модель может четко выучить профиль нормальной работы и выявлять даже минимальные отклонения.

5) Ограничения и дальнейшие работы

Настоящее исследование сфокусировано на двух протоколах и относительно небольшом количестве типов атак. В дальнейшем планируется расширить словарь токенов и методику на другие распространенные протоколы (DNP3, IEC 60870-5-104, PROFINET и др.), а также учесть смешанный трафик, в котором в одной сети присутствуют несколько разных протоколов; это потребует более сложной токенизации и, возможно, мультипротокольного обучения. Кроме того, перспективным направлением является интеграция графовой информации о топологии сети в модель – по аналогии с подходом в работе [4] к интерпретируемости, – когда в трансформер добавляются специальные токены устройств или соединений. Это может помочь локализовать источник аномалии, то есть конкретный узел или сегмент сети. Еще одно направление связано с применением генеративных моделей, например NetGPT, для синтеза аномального трафика и тестирования системы на новых типах атак.

Отдельно отметим класс атак нулевой динамики на киберфизические системы, при которых нарушитель подбирает воздействие так, чтобы выходные измерения оставались согласованными с нормальным режимом, а изменения происходили во внутренней динамике объекта. Такие атаки рассматриваются как особый тип скрытных атак внедрения данных [23]. Для их выявления часто требуется изменить структуру наблюдения или использовать дополнительные источники информации, например независимые измерения и модельно-ориентированные методы контроля [24, 25]. Поскольку предложенный в статье подход опирается на анализ сетевого трафика и протокольной семантики, гарантированное обнаружение атак нулевой динамики в общем случае невозможно, если злоумышленник использует легитимные последовательности команд и сохраняет нормальный профиль обмена. Тем не менее, в малых детерминированных сетях даже скрытные атаки нередко приводят к сдвигам в последовательностях, временны́х интервалах или наборе операций, что потенциально может быть уловлено трансформерной моделью. Авторы рассматривают интеграцию сетевых токенов с данными технологического процесса и мультимодальные модели, например трансформеры для многомерных временны́х рядов, как одно из ключевых направлений продолжения работы [26].

В заключение отметим, что комбинация семантически осмысленного представления данных и мощных архитектур глубокого обучения открывает новые возможности для обеспечения безопасности промышленных информационно-управляющих систем. Предложенный способ кодирования трафика для трансформеров может послужить основой для создания универсальных моделей мониторинга Industrial Ethernet, способных адаптироваться к различным протоколам и своевременно выявлять угрозы в автоматическом режиме.

Список источников

1. Dehlaghi-Ghadim A., Moghadam H.M., Balador A., Hansson H. Anomaly Detection Dataset for Industrial Control Systems // IEEE Access. 2023. Vol. 11. PP. 107982–107996. DOI:10.1109/ACCESS.2023.3320928. EDN:ZMFCCE

2. Shi Z., Luktarhan N., Song Y., Yin H. TSFN: A Novel Malicious Traffic Classification Method Using BERT and LSTM // Entropy. 2023. Vol. 25. Iss. 5. P. 821. DOI:10.3390/e25050821. EDN:HNRTUY

3. Afifi F., Zaki F., Hanif H., Aqil N., Anuar N.B. Transformer-based tokenization for IoT traffic classification across diverse network environments // PeerJ Computer Science. 2025. Vol. 11. P. e3126. DOI:10.7717/peerj-cs.3126. EDN:LFNGZM

4. Marino D.L., Wickramasinghe C.S., Rieger C., Manic M. Self-Supervised and Interpretable Anomaly Detection Using Network Transformers // IEEE Transactions on Industrial Informatics. 2025. Vol. 21. Iss. 5. PP. 4252–4261. DOI:10.1109/TII.2025.3534443. EDN:GDCTHE

5. Buczak A.L., Guven E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection // IEEE Communications Surveys & Tutorials. 2016. Vol. 18. Iss. 2. PP. 1153–1176. DOI:10.1109/COMST.2015.2494502

6. Mitchell R., Chen I.-R. A survey of intrusion detection techniques for cyber-physical systems // ACM Computing Surveys. 2014. Vol. 46. Iss. 4. P. 55. DOI:10.1145/2542049. EDN:OQKUJE

7. Mathur A.P., Tippenhauer N.O. SWaT: a water treatment testbed for research and training on ICS security // Proceedings of the International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater, Vienna, Austria, 11 April 2016). IEEE, 2016. PP. 31–36. DOI:10.1109/CySWater.2016.7469060

8. Ahmed C.M., Palleti V.R., Mathur A.P. WADI: a water distribution testbed for research in the design of secure cyber physical systems // Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater, Pittsburgh, USA, 21 April 2017). New York: ACM, 2017. PP. 25–28. DOI:10.1145/3055366. 3055375

9. Anthi E., Williams L., Burnap P., Jones K. A three-tiered intrusion detection system for industrial control systems // Journal of Cybersecurity. 2021. Vol. 7. Iss. 1. P. tyab006. DOI:10.1093/cybsec/tyab006. EDN:PZNPOU

10. Stouffer K., Pease M., Tang C., Zimmerman T., Pillitteri V., Lightman S., et al. NIST SP 800-82 Rev. 3. Guide to Operational Technology (OT) Security. 2023. DOI:10.6028/NIST.SP.800-82r3

11. Goldenberg I., Wool A. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems // International Journal of Critical Infrastructure Protection. 2013. Vol. 6. Iss. 2. PP. 63–75. DOI:10.1016/j.ijcip.2013.05.001

12. Azab A., Khasawneh M., Alrabaee S., Choo K.K.R., Sarsour M. Network traffic classification: Techniques, datasets, and challenges // Digital Communications and Networks. 2024. Vol. 10. Iss. 3. PP. 676–692. DOI:10.1016/j.dcan.2022.09.009. EDN:BWGLKV

13. Mirsky Y., Doitshman T., Elovici Y., Shabtai A. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection // Network and Distributed System Security Symposium (NDSS, San Diego, USA, 18–21 February 2018). 2018. DOI:10.14722/ndss.2018.23204

14. Lotfollahi M., Siavoshani M.J., Zade R.S.H., Saberian M. Deep packet: a novel approach for encrypted traffic classification using deep learning // Soft Computing. 2020. Vol. 24. P. 1999–2012. DOI:10.1007/s00500-019-04030-2. EDN:XWSLQV

15. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., et al. Attention Is All You Need // Advances in Neural Information Processing Systems. 2017. DOI:10.48550/arXiv.1706.03762

16. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding // Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL, Minneapolis, USA, 2–7 June 2019). 2019. DOI:10.48550/arXiv.1810.04805

17. Lin X., Xiong G., Gou G., Li Z., Shi J., Yu J. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification // arXiv2202.06335. 2022. DOI:10.48550/arXiv.2202.06335

18. Meng X., Lin C., Wang Y., Zhang Y. NetGPT: Generative Pretrained Transformer for Network Traffic // arXiv:2304.09513. 2023. DOI:10.48550/arXiv.2304.09513

19. Brenner P., Fabini J., Offermanns M., Semper S., Zseby T. Malware communication in smart factories: A network traffic data set // Computer Networks. 2024. Vol. 255. P. 110804. DOI:10.1016/j.comnet.2024.110804. EDN:FZKJWH

20. Malware in Smart Factory // TU Wien Research Data Repository. 2024. DOI:10.48436/2sv8c-ykc69

21. Paxson V. Bro: a system for detecting network intruders in real-time // Computer Networks. 1999. Vol. 31. Iss. 23–24. PP. 2435–2463. DOI:10.1016/S1389-1286(99)00112-7

22. Dietmüller A., Ray S., Jacob R., Vanbever L. A new hope for network model generalization // Proceedings of the 21st ACM Workshop on Hot Topics in Networks (HotNets '22, Austin, USA, 14–15 November 2022). New York: Association for Computing Machinery, 2022. PP. 152–159. DOI:10.1145/3563766.3564104

23. Teixeira A.M.H., Shames I., Sandberg H., Johansson K.H. Revealing stealthy attacks in control systems // Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing (Monticello, USA, 01–05 October 2012). IEEE, 2012. PP. 1806–1813. DOI:10.1109/Allerton.2012.6483441

24. Teixeira A., Shames I., Sandberg H., Johansson K.H. A secure control framework for resource-limited adversaries // Automatica. 2015. Vol. 51. PP. 135–148. DOI:10.1016/j.automatica.2014.10.067

25. Kim J., Back J., Park G., Lee C., Shim H., Voulgaris P.G. Neutralizing zero dynamics attack on sampled-data systems via generalized holds // Automatica. 2020. Vol. 113. P. 108778. DOI:10.1016/j.automatica.2019.108778. EDN:GKLFBB

26. Tuli S., Casale G., Jennings N.R. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data // Proceedings of the VLDB Endowment. 2022. Vol. 15. Iss. 6. PP. 1201–1214. DOI:10.14778/3514061.3514067. EDN:LPTVUQ

References

1. Dehlaghi-Ghadim A., Moghadam H.M., Balador A., Hansson H. Anomaly Detection Dataset for Industrial Control Systems. *IEEE Access*. 2023;11:107982–107996. DOI:10.1109/ACCESS.2023.3320928. EDN:ZMFCCE

2. Shi Z., Luktarhan N., Song Y., Yin H. TSFN: A Novel Malicious Traffic Classification Method Using BERT and LSTM. *Entropy*. 2023;25(5):821. DOI:10.3390/e25050821. EDN:HNRTUY

3. Afifi F., Zaki F., Hanif H., Aqil N., Anuar N.B. Transformer-based tokenization for IoT traffic classification across diverse network environments. *PeerJ Computer Science*. 2025;11(e3126). DOI:10.7717/peerj-cs.3126. EDN:LFNGZM

4. Marino D.L., Wickramasinghe C.S., Rieger C., Manic M. Self-Supervised and Interpretable Anomaly Detection Using Network Transformers. *IEEE Transactions on Industrial Informatics*. 2025;21(5):4252–4261. DOI:10.1109/TII.2025.3534443. EDN:GDCTHE

5. Buczak A.L., Guven E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*. 2016. Vol. 18. Iss. 2. PP. 1153–1176. DOI:10.1109/COMST.2015.2494502

6. Mitchell R., Chen I.-R. A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. *ACM Computing Surveys*. 2014;46(4):55. DOI:10.1145/2542049. EDN:OQKUJE

7. Mathur A.P., Tippenhauer N.O. SWaT: a water treatment testbed for research and training on ICS security. *Proceedings of the International Workshop on Cyber-Physical Systems for Smart Water Networks, CySWater, 11 April 2016, Vienna, Austria*. IEEE; 2016. p.31–36. DOI:10.1109/CySWater.2016.7469060

8. Ahmed C.M., Palleti V.R., Mathur A.P. WADI: a water distribution testbed for research in the design of secure cyber physical systems. *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks, CySWater, 21 April 2017, Pittsburgh, USA*. New York: ACM; 2017. p.25–28. DOI:10.1145/3055366. 3055375

9. Anthi E., Williams L., Burnap P., Jones K. A three-tiered intrusion detection system for industrial control systems. *Journal of Cybersecurity*. 2021;7(1):tyab006. DOI:10.1093/cybsec/tyab006. EDN:PZNPOU
10. Stouffer K., Pease M., Tang C., Zimmerman T., Pillitteri V., Lightman S., et al. *NIST SP 800-82 Rev. 3. Guide to Operational Technology (OT) Security*. 2023. DOI:10.6028/NIST.SP.800-82r3
11. Goldenberg I., Wool A. Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection*. 2013;6(2):63–75. DOI:10.1016/j.ijcip.2013.05.001
12. Azab A., Khasawneh M., Alrabaee S., Choo K.K.R., Sarsour M. Network traffic classification: Techniques, datasets, and challenges. *Digital Communications and Networks*. 2024;10(3):676–692. DOI:10.1016/j.dcan.2022.09.009. EDN:BWGLKV
13. Mirsky Y., Doitshman T., Elovici Y., Shabtai A. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. *Network and Distributed System Security Symposium, NDSS, 18–21 February 2018, San Diego, USA*. 2018. DOI:10.14722/ndss.2018.23204
14. Lotfollahi M., Siavoshani M.J., Zade R.S.H., Saberian M. Deep packet: a novel approach for encrypted traffic classification using deep learning. *Soft Computing*. 2020;24:1999–2012. DOI:10.1007/s00500-019-04030-2. EDN:XWSLQV
15. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., et al. Attention Is All You Need. *Advances in Neural Information Processing Systems*. 2017. DOI:10.48550/arXiv.1706.03762
16. Devlin J., Chang M.-W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, NAACL, 2–7 June 2019, Minneapolis, USA*. 2019. DOI:10.48550/arXiv.1810.04805
17. Lin X., Xiong G., Gou G., Li Z., Shi J., Yu J. ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. *arXiv2202.06335*. 2022. DOI:10.48550/arXiv.2202.06335
18. Meng X., Lin C., Wang Y., Zhang Y. NetGPT: Generative Pretrained Transformer for Network Traffic. *arXiv:2304.09513*. 2023. DOI:10.48550/arXiv.2304.09513
19. Brenner P., Fabini J., Offermanns M., Semper S., Zseby T. Malware communication in smart factories: A network traffic data set. *Computer Networks*. 2024;255:110804. DOI:10.1016/j.comnet.2024.110804. EDN:FZKJWH.
20. *Malware in Smart Factory*. TU Wien Research Data Repository. 2024. DOI:10.48436/2sv8c-ykc69
21. Paxson V. Bro: a system for detecting network intruders in real-time // Computer Networks. 1999. Vol. 31. Iss. 23–24. PP. 2435–2463. DOI:10.1016/S1389-1286(99)00112-7
22. Dietmüller A., Ray S., Jacob R., Vanbever L. A new hope for network model generalization. *Proceedings of the 21st ACM Workshop on Hot Topics in Networks, HotNets '22, 14–15 November 2022, Austin, USA*. New York: Association for Computing Machinery; 2022. p.152–159. DOI:10.1145/3563766.3564104
23. Teixeira A.M.H., Shames I., Sandberg H., Johansson K.H. Revealing Stealthy Attacks in Control Systems. *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing, 01–05 October 2012, Monticello, USA*. IEEE; 2012. p.1806–1813. DOI:10.1109/Allerton.2012.6483441
24. Teixeira A., Shames I., Sandberg H., Johansson K.H. A secure control framework for resource-limited adversaries. *Automatica*. 2015;51:135–148. DOI:10.1016/j.automatica.2014.10.067
25. Kim J., Back J., Park G., Lee C., Shim H., Voulgaris P.G. Neutralizing zero dynamics attack on sampled-data systems via generalized holds. *Automatica*. 2020;113:108778. DOI:10.1016/j.automatica.2019.108778. EDN:GKLFBB
26. Tuli S., Casale G., Jennings N.R. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *Proceedings of the VLDB Endowment*. 2022;15(6):1201–1214. DOI:10.14778/3514061.3514067. EDN:LPTVUQ

Статья поступила в редакцию 04.01.2026; одобрена после рецензирования 24.02.2026; принята к публикации 05.03.2026
The article was submitted 04.01.2026; approved after reviewing 24.02.2026; accepted for publication 05.03.2026

Информация об авторах:

БУИ Нгуен Чунг
аспирант кафедры инфокоммуникационных систем и сетей Московского физико-технического института (национального исследовательского университета)
https://orcid.org/0009-0004-1489-927X

ПАЩЕНКО Федор Федорович
доктор технических наук, профессор, главный научный сотрудник Лаборатории № 40 «Интеллектуальных систем управления и моделирования» Института проблем управления им. В.А. Трапезникова РАН, профессор кафедры инфокоммуникационных систем и сетей Московского физико-технического института (национального исследовательского университета)
https://orcid.org/0000-0001-8898-2720

ТО Фи Тхань
аспирант кафедры инфокоммуникационных систем и сетей Московского физико-технического института (национального исследовательского университета)
https://orcid.org/0009-0006-6742-5398

Авторы сообщают об отсутствии конфликтов интересов.
The authors declare no conflicts of interests.