Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Кафедра «Информационная безопасность компьютерных систем»

О т ч е т

о н а у ч н о - и с с л е д о в а т е л ь с к о й р а б о т е

Обнаружение сетевых атак с помощью LSTM сетей

Выполнил
студент гр.43609/1 Д.А. Куликов

Руководитель
доцент, к.т.н. В.В. Платонов

«___» _____ 2019 г.

Санкт-Петербург

2019

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Нарушители со временем используют все более изощренные и сложные методы атак, поэтому отличить их от нормального трафика становится все труднее и труднее. Так как сигнатурные методы не всегда могут выявить новые атаки, необходимо применять методы выявления аномального трафика. Благодаря своим способностям к обучению, с этой целью могут справиться нейронные сети и другие инструменты машинного обучения. Также широко применяется data mining, для выявления значимых признаков, по которым можно классифицировать трафик.

В данной работе реализуется выявление аномального трафика с помощью LSTM сетей, или сетей с долгой краткосрочной памятью. Также проводится их описание и анализ результатов.

Для обучения и оценки работоспособности сети использовались данные UNSW-NB15 [1], опубликованные в 2018 году. Также, для проверки и сравнения использовались KDD Cup данные 2000 года [2].

Целью данной работы является исследование возможности применения LSTM сетей для обнаружения сетевых атак. Реализуется как бинарный классификатор, так и классификатор атак по типам.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- изучить механизм работы LSTM сетей;

- предварительно обработать данные, выбрать архитектуру и параметры сети;
- выполнить программную реализацию бинарного классификатора с помощью LSTM, обучить сеть и проанализировать результаты.
- выполнить программную реализацию классификатора с разделением атак на типы с помощью LSTM, обучить сеть и проанализировать результаты.

1. RNN И LSTM СЕТИ

Сетевые протоколы работают как функция от времени, поэтому используются рекуррентные нейронные сети (RNN), которые способны учитывать зависимость между данными на разных временных промежутках.

Связи в RNN проходят не только между нейронами от входного слоя к выходному, но и от нейрона к «самому себе», а именно, к его предыдущему значению или предыдущим значениям других нейронов того же слоя.

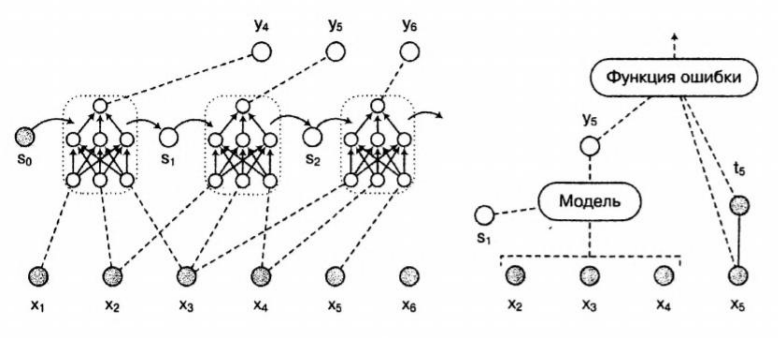


Рисунок 1 – Архитектура рекуррентной нейронной сети [3]

На рисунке 1 s_t отображает скрытое состояние нейрона в период времени t, которое зависит от того что происходило ранее.

Выделяют пять типов задач по отношению входов к выходам:

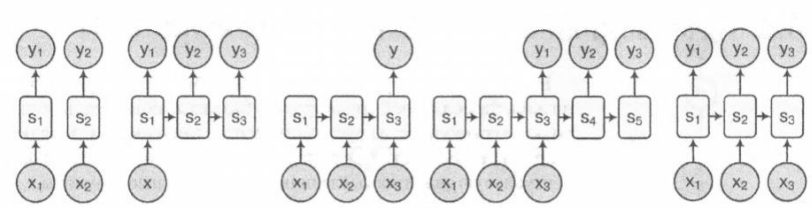


Рисунок 2 – Задачи, решаемые рекуррентными сетями [3]

На рисунке 2 графически представлены следующие схемы:

- Один вход, один выход; обычная схема, здесь внутренние состояния никуда не передаются, следовательно, ничем не отличается от обычной сети.
- Один вход, много выходов; используется, например, когда генерируется описание на основе картинки.
- Один выход, много входов; такие задачи применяются для классификации последовательностей.
- Много входов, а потом много выходов; по этой схеме работают интеллектуальные переводчики.
- Много входов, много выходов, синхронизированные.

1.1. RNN

Рассмотрим архитектуру обычной рекуррентной сети:

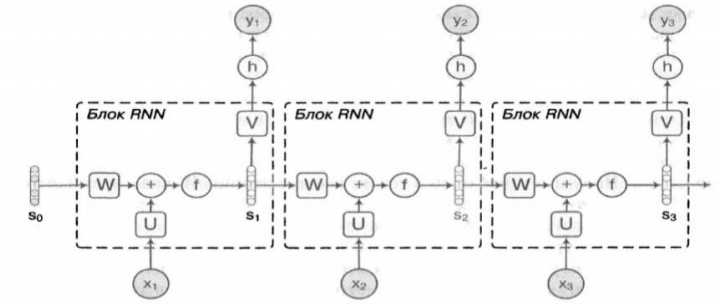


Рисунок 3 – Архитектура обычной рекуррентной нейронной сети [3]

Пусть W – матрица весов для входа состояния, U – матрица входных весов последовательности, V – веса выходов, h –функция выхода (например, softmax), f –нелинейная функция.

Тогда в момент времени t:

$$a_t = b + W_{s_{t-1}} + Ux_t,$$

$$s_t = f(a_t),$$

$$o_t = c + V_{s_t},$$

$$y_t = h(o_t).$$

Если прошло T шагов, $t = 1 \dots T$, и задана функция ошибки $Loss(o_1, \dots, o_T)$. Тогда можно посчитать частные производные, разворачивая состояния s_t $t = T \dots 1$. Таким образом, каждый вес из матрицы W будет встречаться $T - 1$ раз. Таким образом, можно представить рекуррентную сеть, как

многоуровневую обычную сеть, в которой одни и те же веса используются на каждом уровне.

Недостатки:

- Взрывные градиенты. Если матрица весов такова, что значительно увеличивает норму градиента при проходе через один развернутый слой, то при проходе через T слоев эта норма возрастет экспоненциально от T.
- Также влияние текущего входа затухает экспоненциально по мере удаления, что не позволяет распознать далекие зависимости в данных.

1.2. LSTM

В LSTM узлы называются гейтами: входной, забывающий и выходной, а также рекуррентная ячейка со скрытым состоянием.

Обозначив x_t входной вектор во время t, h_t – вектор скрытого состояния во время t, W_i – матрицы весов ко входу, W_h - матрицы весов в рекуррентных соединениях, b – векторы свободных членов, получим формальный вид работы LSTM:

$$c'_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c'}),$$
$$i_i = \sigma(W_{x^i}\, x_t + W_{hi}h_{t-1} + b_i),$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}\, h_{t-1} + b_f),$$
$$o_t = \sigma(W_{x_0}\, x_t + W_{h_o}\, h_{t-1} + b_o),$$
$$c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t,$$
$$h_t = o_t \cdot \tanh(c_t),$$

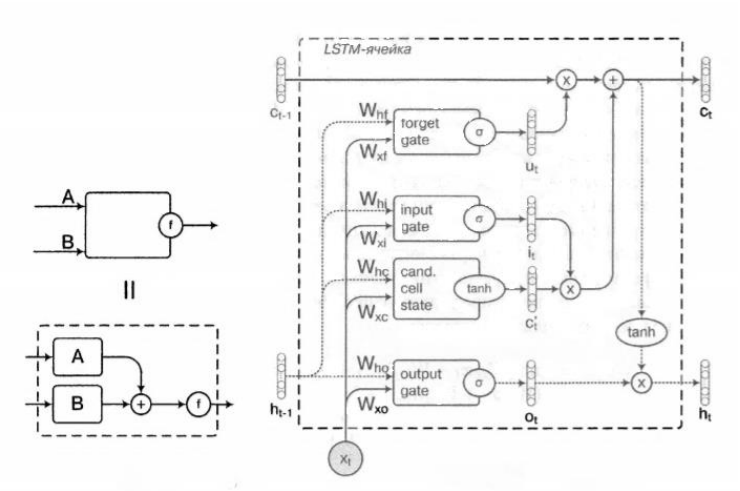


Рисунок 4 – Структура LSTM ячейки [3]

Также, как и в RNN, на вход подается вектор входных данных x_t и вектор скрытого состояния h_{t-1}. Функцию памяти выполняет вектор c_t.

Из формулы $c_t = f_t \cdot c_{t-1} + i_t \cdot c'_t$ можно увидеть, что новое значение ячейки памяти получается, как линейная комбинация из старого, умноженного на выход забывающего гейта f_t и нового кандидата c'_t с коэффициентами из входного гейта i_t.

Таким образом, если значения f_t малы, то значение ячейки памяти «забудется», а если велики, то к ним добавятся новые данные. Благодаря своим свойствам, на каждом шаге может быть изменена только часть памяти ячейки, другая часть памяти может сохраниться, это определяется матрицами весов и делает структуру ячейки гибкой.

LSTM позволяет решить проблему исчезающих градиентов. Для простоты положим $f_t = 1$ для всех t. Тогда:

$$c_t = c_{t-1} + i_t \cdot c'_t,$$

$$\frac{\partial c_t}{\partial c_{t-1}} = 1.$$

Данный эффект называется «каруселью константной ошибки», это означает, что скрытое состояние может сохраняться неограниченно долго, если не будут перезаписаны. Таким образом, LSTM может запоминать контекст на гораздо более продолжительное время чем RNN, при необходимости. От взрывающихся же градиентов LSTM также, как и RNN не защищены.

Существует множество модификаций LSTM, вот некоторые из них [3]:

1. LSTM без входного гейта;
2. LSTM без забывающего гейта;
3. LSTM без выходного гейта o_t;
4. LSTM без функции активации σ на входном гейте;
5. LSTM без функции активации σ на выходном гейте;
6. LSTM без замочных скважин;
7. LSTM со связанными входным и забывающим гейтом;
8. LSTM с дополнительными рекуррентными связями на каждом гейте;

2. ОПИСАНИЕ ДАННЫХ И ИХ ОБРАБОТКА

Сетевые пакеты UNSW-NB15 были получены авторами [1] инструментом IXIA PerfectStorm в лаборатории Cyber Range Австралийского центра кибербезопасности (ACCS) с целью получить нормальный и зловредный профиль поведения современной сети.

Используемые данные делятся на нормальный трафик и трафик с аномалиями. Аномальный трафик подразделяется на 9 типов:

- Эксплоиты – фрагмент программного кода или последовательность команд, которые используются уязвимости в программном обеспечении и применяются для осуществления атаки на вычислительную систему.
- Разведка – получение и обработка данных об информационной системе, ресурсах информационной системы, используемых устройствах и программном обеспечении и их уязвимостях, средствах защиты, а также о границе проникновения в информационную систему.
- Атака отказа в обслуживании – атака на вычислительную систему с целью довести её до отказа, то есть создание таких условий, при которых добросовестные пользователи системы не могут получить доступ к предоставляемым системным ресурсам (серверам), либо этот доступ затруднён.
- Стандартные атаки – атаки носящие массовый характер.

- Шелл-код – попытка загрузки исполняемого кода, который передаёт управление командному процессору.
- Использование фаззинга – метода тестирования программного обеспечения, заключающегося в передаче приложению на вход неправильных, неожиданных или случайных данных, с целью найти в нем ошибки и уязвимости.
- Черви – атака сетевыми червями, которые используют сетевые протоколы и устройства для распространения.
- Потайные ходы – атака, при которой на компьютере пользователя открывается канал передачи данных, доступный нарушителю.
- Анализ сети – разведка с целью построения карты сети информационной системы.

Всего, данные содержат 49 признаков, 2 из которых относятся к маркировке трафика. Среди прочих – сервисы (около 150 шт.), IP-адреса, количественные характеристики потока трафика, флаги (в зависимости от протокола), порты, размеры пакетов, временные характеристики трафика и количественные характеристики соединений.

После выделения строковых признаков, они кодируются вещественными числами, далее проводится min-max масштабирование, для приведения значений данных в диапазон $[0, 1]$.

Из обработанных, пакетов с маркировкой нормального трафика – 1867614. Пакетов с маркировкой атаки – 232389. Из этого можно сделать вывод, что данные сильно несбалансированные, откуда следует необходимость увеличения весов функции ошибки для аномального трафика.

Аномальный трафик распределяется следующим образом:

Таблица 1

Количество пакетов с разными типами атак

Атака	Количество пакетов
Эксплоиты	33086
Разведка	10457
Отказ в обслуживании	11446
Стандартные	153603
Шелл-код	1140
Фаззинг	18856
Черви	131
Потайные ходы	1663
Анализ сети	2007

Пропорционально этим данным, для классификации по типам атак были увеличены веса функции ошибки:

Таблица 2

Веса для функции ошибки

Атака	Множитель
Эксплоиты	55
Разведка	172
Отказ в обслуживании	157
Стандартные	12
Шелл-код	1580
Фаззинг	95
Черви	13000
Потайные ходы	1080
Анализ сети	895

В данных KDD 2000 [2] 71460 пакетов с атаками и 77049 пакетов нормального трафика. Каждый пакет имеет 42 признака, среди которых тип протокола, сервис, флаги, количество неверных фрагментов, состояние сессии клиента, проценты ошибок передачи, количественные характеристики трафика. Данные сбалансированные, поэтому дополнительные веса применять не требуется.

3. РЕАЛИЗАЦИЯ И АНАЛИЗ LSTM СЕТИ

Реализация выполнялась на языке Python, с помощью открытой нейросетевой библиотеки Keras [4] с TensorFlow. Keras предоставляет реализацию LSTM слоя и выбор следующих параметров:

1. Units – количество LSTM ячеек в слое (соответствует размерности выхода).
2. Return_sequence – позволяет либо получать на выходе только последнее значение из выходной последовательности, либо все значения (в данном случае используются все).
3. Stateful – позволяет либо запоминать скрытое состояние ячеек при каждом новом входном значении, либо обнулять его.

Этот набор параметров позволяет использовать сеть для различных задач, описанных ранее.

Также, для ускорения вычислений, стоит использовать Keras на GPU, что позволяет значительно снизить время ожидания, например, при обучении.

3.1. Бинарный классификатор на основе LSTM

Для данных из KDD 2000 [2], была реализована сеть, которую можно представить в следующем виде:

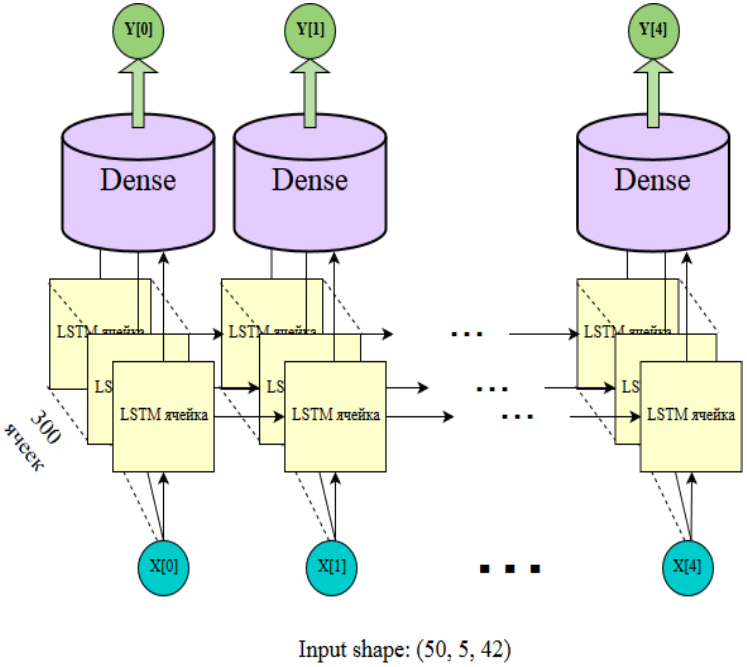


Рисунок 5 – Представление работы сети с одним LSTM слоем

На рисунке 5 изображена работа сети с одним LSTM слоем, 300 ячейками, 5 временными интервалами и 42 признаками входных данных (размерность векторов $X[i]$), а также размером батча, равным 50.

Далее, к ней были добавлены еще 2 таких же LSTM слоя, друг за другом.

Результаты работы сети, для объединенного множества тестовых и тренировочных данных из KDD 2000 [2], с перекрестной проверкой по 5 блокам с 4 эпохами обучения и размером батча, равным 50:

Таблица 3

Матрица неточностей

		Результаты классификации	
		Отрицательный	Положительный
Настоящие значения	Отрицательный	75003	1500
	Положительный	2300	68697

Далее, можем посчитать следующие характеристики:

Таблица 4

Характеристики проверки

Метрика	Значение
True Positive Rate	0.4657
True Negative Rate	0.5084
False Positive Rate	0.0102
False Negative Rate	0.0156
Accuracy	0.9742

Precision	0.9786
Recall	0.9676
F1	0.9745

Для данных [1] такая сеть же сеть показала результаты немногим хуже: на первом этапе перекрестной проверки была получена следующая матрица неточности:

Таблица 5

Матрица неточностей

		Результаты классификации	
		Отрицательный	Положительный
Настоящие значения	Отрицательный	673983	3802
	Положительный	3900	18315

В UNSW-NB15[1] данных гораздо больше, это приводит к переобучению, поэтому используется метод dropout. Суть заключается в том, что можно обучить ансамбль сетей, вместо одной, а затем усреднить полученные результаты. Для этого проводится исключение из сети нейронов с некоторой вероятностью, так что он возвращает 0 для любых входных данных. Таким образом, нейроны не полагаются на другие, для исправления ошибок, что предотвращает переобучение. Ниже приведена иллюстрация, как работает dropout во время обучения:

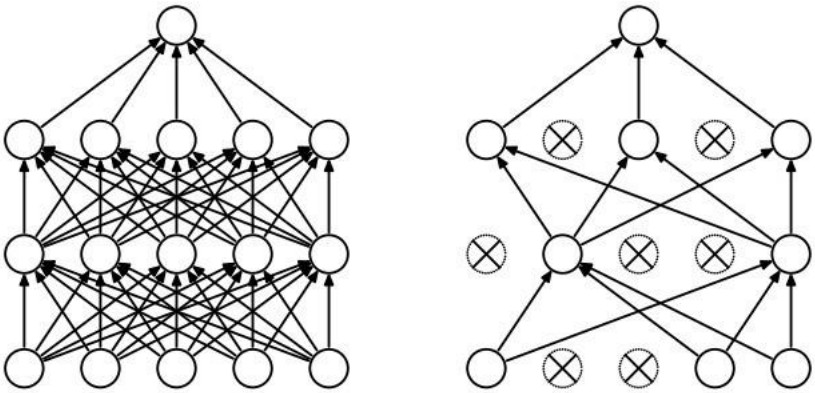


Рисунок 6 – Работа сети до и после применения dropout

Применив dropout между всеми слоями и направив выход каждого слоя на вход каждому последующему, был получен следующий результат для данных из работы [1]:

Таблица 6

Матрица неточностей

		Результаты классификации	
		Отрицательный	Положительный
Настоящие значения	Отрицательный	1855928	11682
	Положительный	15657	216732

Далее, можем посчитать следующие характеристики:

Таблица 7

Показатели проверки

Метрика	Значение
True Positive Rate	0.1032
True Negative Rate	0.8838
False Positive Rate	0.0056
False Negative Rate	0.0074
Accuracy	0.9869
Precision	0.9489
Recall	0.9326
F1	0.9407

Таким образом, LSTM сеть успешно доказала возможность применения для обнаружения атак, однако некоторые все же остались необнаруженными (7% от общего числа атак).

3.2. Классификатор типов атак на основе LSTM

Для построения классификатора атак по типам использовалась та же архитектура, описанная в разделе 3.1, за тем исключением, что Dense слой имеет размер выходного вектора на единицу больший, чем количество классифицируемых типов атак. Тогда классы маркируются целочисленным вектором единичной длины.

Проведя перекрестную проверку по 5 блокам, для нормального трафика получена следующая матрица неточности:

Таблица 8

Матрица неточностей нормального трафика

		Результаты классификации	
		Отрицательный	Положительный
Настоящие значения	Отрицательный	231338	201
	Положительный	31636	1826825

Для классов атак, расположим матрицы таким же образом но в одну таблицу:

Таблица 9

Матрицы неточностей атак

Эксплоиты	
2042255	14951
17313	15481
Стандартные	
1936438	9
18092	135461
Черви	
2089871	0
1652	0

Разведка	
2079631	19
10291	59
Шелл-код	
2088869	0
1131	0
ПХ	
2088348	0
1652	0

DoS	
2062191	16432
4149	7228
Фаззинг	
2071450	2
18455	93
Анализ	
2087995	0
2005	0

Рассчитаем характеристики для этих значений:

Таблица 10

Классификация атак по типу

Название	Precision	Accuracy	Recall	F1
Нормальный	0.9999	0.9848	0.9829	0.9839
Стандартные	0.9999	0.9913	0.8822	0.9336
DoS	0.3054	0.9902	0.6353	0.7740
Эксплоиты	0.5087	0.9846	0.4721	0.6382
Разведка	0.7564	0.9951	0.0057	0.0113
Фаззинг	0.9789	0.9912	0.0050	0.0099
Анализ	0	0.9990	0	0
ПХ	0	0.9992	0	0

Шелл-код	0	0.9918	0	0
Черви	0	0.9961	0	0

Достаточно хорошо классифицируются general атаки, атаки DoS, и эксплоиты. Остальные не определяются вовсе, поэтому было решено добавить к функции потерь веса, указанные в разделе 2. После этого было получено:

Таблица 11

Матрицы неточностей атак

Эксплоиты	
2053020	4186
26577	6217
Стандартные	
1936168	279
18031	135522
Черви	
2081755	8116
49	80

Разведка	
2074483	5167
6347	4003
Шелл-код	
2072163	16706
381	750
ПХ	
2062625	25723
332	1320

DoS	
2059514	19109
3201	8176
Фаззинг	
2060389	11063
11989	6559
Анализ	
2065715	22280
622	1383

Таблица 12

Классификация атак по типу

Название	Precision	Accuracy	Recall	F1
Нормальный	1	0.9480	0.9415	0.9699
Стандартные	0.9979	0.9912	0.8826	0.9367

DoS	0.2997	0.9893	0.7186	0.4229
Разведка	0.4365	0.9944	0.3868	0.4101
Фаззинг	0.3722	0.9889	0.3536	0.3626
Эксплоиты	0.5976	0.9852	0.1896	0.2878
Анализ	0.0584	0.9890	0.6898	0.1076
ПХ	0.0488	0.9875	0.7990	0.0919
Шелл-код	0.0429	0.9918	0.6631	0.0805
Черви	0.0098	0.9961	0.6202	0.0192

Наихудшими показателями обладают атаки с малым количеством примеров в выборке, несмотря на то, что у них завышенные веса, это приводит к значительному росту FP для этих классов. Повышение весов улучшило показатели определения атаки разведки, но ухудшило обнаружение DoS, что может говорить о некой схожести в поведении данных атак (например, прослушивание портов может быть похоже на DoS). Также улучшились показатели для атак фаззингом, но ухудшились показатели для атак эксплоитами, возможно, это также говорит о их зависимости.

4. АНАЛИЗ НЕОБХОДИМОСТИ ПАМЯТИ СЕТИ

Stateful модель, используемая в разделе 3 данной работы не подходит для изучения того, насколько далеко сеть должна запоминать данные, так как мы не можем знать, что она хранит в состоянии ячейки на каждой итерации. Ниже приведена иллюстрация того, как необходимо сформировать данные для того, чтобы быть уверенным, что stateless модель может запомнить все что нужного из явно ограниченного количества предыдущих пакетов. После каждой партии данных, содержащей несколько, в данном случае 5, пакетов, состояние ячеек сети будет обнуляться, и они будут забывать о предыдущих пакетах.

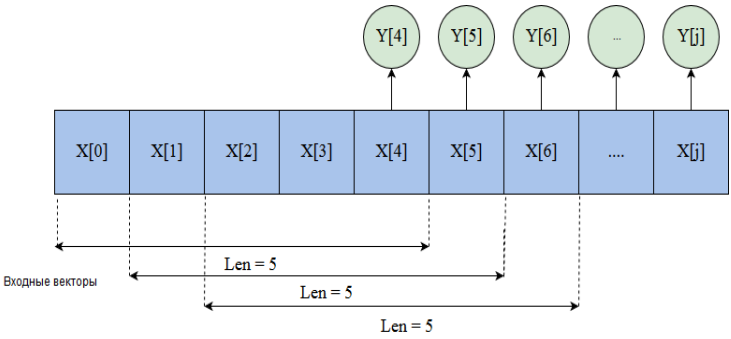
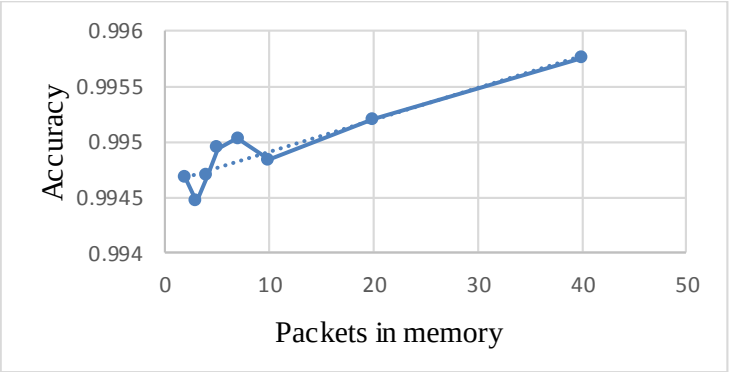
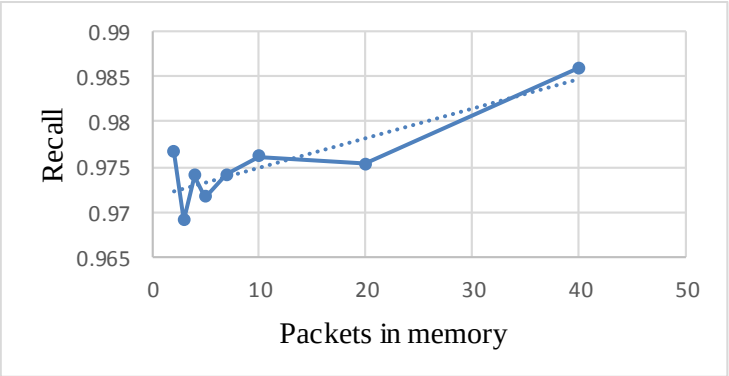


Рисунок 7 – Формирование данных для подачи stateless модели

Проведя несколько таких манипуляций, для разных величин запоминаемых пакетов, были получены следующие графики:





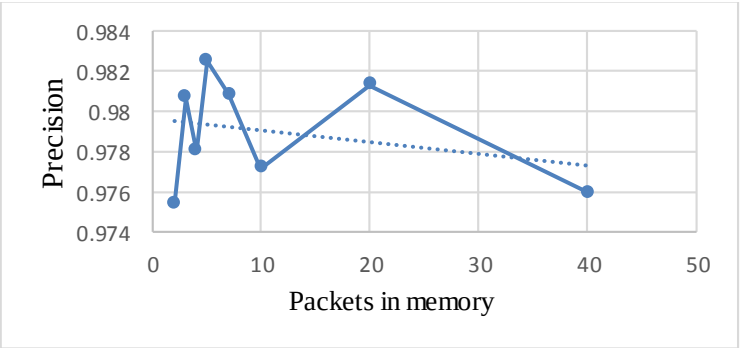


Рисунок 8 – Графики зависимости характеристик от количества запоминаемых пакетов

Данные модели получили лучшие результаты, чем stateful (вероятно, это связано с искусственно увеличенной выборкой, путем дублирования векторов столько раз, сколько пакетов запоминается (см. рисунок 7). Производительность у них также хуже во столько раз, во сколько больше пакетов запоминается (в stateless, с данными с рисунка 7, сначала будут поданы $X[0] - X[4]$, далее $X[1] - X[5]$, и т.д., в stateful же, ранее рассмотренной, будут подаваться $X[0] - X[4]$, далее $X[5] - X[9]$ и т.д.).

Зависимость от количества запоминаемых пакетов кажется небольшой, но если учитывать что выборка большая, но это помогает обнаруживать до 1000 новых атак, нежели, если не иметь истории прошлых пакетов, так, запомним 10 пакетов, сеть выявила на 500 штук атак больше, чем зная о 4 пакетах.

Ухудшение показателей после 20 можно связать либо с переобучением сети, либо слишком большим количеством схожей и лишней информации. Если запоминать 40 пакетов, то в выборке каждый пакет будет встречаться 40 раз, что кажется излишним и может приводить к переобучению. К тому же, это сильно увеличивает размер (400 Мб превращаются в 16 Гб, которые уже сложно обработать).

Таким образом, оптимальные значения достигаются при запоминании 20 пакетов.

ЗАКЛЮЧЕНИЕ

В данной работе были построены бинарный классификатор, а также классификатор атак по типам на основе LSTM, они позволяют обнаруживать сетевые атаки с высокой точностью. Система доказала возможность обнаружения аномалий в трафике, что важно, учитывая растущее количество новых атак.

Была реализована система, позволяющая проводить различное конфигурирование сети, для анализа полученных с ее помощью результатов.

Также была выявлена необходимость в запоминании предыдущих пакетов, которая с учетом увеличивающейся сложности атак будет только нарастать, поэтому и системы обнаружения должны тоже становиться сложнее.

Метод, реализованный в данной работе необходимо использовать в совокупности с сигнатурными методами, реализуемыми такими системами, как Snort.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 The UNSW-NB15 Dataset [Электронный ресурс]. — Режим доступа: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.— (Дата обращения: 27.06.2019).

2 The KDD Cup [Электронный ресурс]. — Режим доступа: https://semanticommunity.info/Data_Science/KDD_Cup.— (Дата обращения: 27.06.2019).

3 Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия «Библиотека программиста»).

4 Keras: The Python Deep Learning library [Электронный ресурс]. — Режим доступа: https://keras.io/. — (Дата обращения: 27.06.2019).

5 LSTM – сети долгой краткосрочной памяти [Электронный ресурс]. — Режим доступа: https://habr.com /ru/company/wunderfund/blog/331310/. — (Дата обращения: 27.06.2019).