Абрамкин Роман Викторович, адъюнкт, avg62rus@rambler.ru, Россия, Санкт-Петербург, Военная академия связи им. Маршала Советского Союза С. М. Буденного

THE MODEL OF THE FUNCTIONING OF THE COMMUNICATION NODE IN CASE OF FAILURES OF ELEMENTS OF THE POWER SUPPLY SYSTEM

R.V. Abramkin

A model is considered that allows in the first approximation to assess the reliability of the functioning of a field communication node in the conditions of failures of elements of its power supply system. The model is based on an approach that allows to decompose the reliability of the functioning of the node into structural and functional components, as well as probabilistic and temporal characteristics of the process of restoring the connectivity of the node structure.

Key words: field communication node, reliability of functioning, structural reliability, functional reliability, probability of connectivity, nodal ways of passing messages.

Abramkin Roman Victorovich, postgraduate, avg62rus@rambler.ru, Russia, St. Petersburg, Military Academy of Communications named after Marshal Soviet Union S.M. Budyonny

УДК 004.056.57
DOI: 10.24412/2071-6168-2021-12-182-191

МЕТОДИКА ОБНАРУЖЕНИЯ КОЛЛИЗИЙ СЕТЕВОГО ТРАФИКА

А.М. Крибель

Исследуются возможности использования искусственных нейронных сетей сложной архитектуры (в частности, на основе Long Short-Term Memory (LSTM)) для обнаружения коллизий сетевого трафика, вызванных кибератаками, использующих уязвимости межсетевых экранов веб-приложений (Web Application Firewall, WAF), которые выявить другими средствами весьма затруднительно. Экспериментальные результаты, основанные на сгенерированном наборе данных, подтвердили высокую эффективность разработки: методика позволяет обнаруживать кибератаки в реальном масштабе времени.

Ключевые слова: компьютерная сеть, обнаружение коллизий, кибератака, межсетевой экран веб-приложений, нейронная сеть, LSTM, автоэнкодер.

Современный этап развития общества характеризуется повышением роли информационной сферы, представляющей собой совокупность информации, информационной инфраструктуры и субъектов, осуществляющих сбор, формирование, распространение и использование информации.

К числу наиболее заметно нарастающих угроз информационной сферы относятся кибератаки. Статистика показывает, что большинство кибератак направлены на компьютерные сети. По сравнению с 2012 г. в 2020 г. количество кибератак на элементы компьютерных сетей увеличилось более чем в два раза [1].

Современные кибератаки представляют собой сложное комплексное воздействие на сеть, в результате которого осуществляется компрометация киберресурсов и нарушается управление процессами в киберпространстве. Зачастую этому предшествует долгая и кропотливая работа: разведка, поиск характерных уязвимостей и захват информационных активов.

Меняющийся ландшафт угроз, частота их появления, сложность и целевой характер атак – все это требует эволюции существующей парадигмы в сфере информационной безопасности.

Для минимизации потенциального ущерба важен переход к сочетанию технологий предотвращения, обнаружения и реагирования на кибератаки. Однако сегодня анализ инцидента происходит в основном «по факту», путем «латания дыр» в уязвимостях систем обеспечения информационной безопасности. Большинство государственных и частных организаций имеют средства для обнаружения известных атак, хотя, как показывает практика, данные решения не всегда спасают от вредоносных сетевых вторжений.

Самое сложное в деле защиты конфиденциальных информационных ресурсов – это остановить *неизвестные* атаки, созданные специально с целью обхода имеющейся защиты и использующие изменения сигнатур и шаблонов поведения.

В 2020–2021 гг. более 75% кибератак были направлены на уязвимости веб-приложений и сайтов [2]. Такие атаки, как правило, незаметны для инфраструктуры и служб информационной безопасности. Уязвимости веб-приложений несут в себе, в свою очередь, риски компрометации учетных записей и персональных данных пользователей, паролей, номеров кредитных карт. Кроме того, уязвимости в веб-сайте служат для злоумышленников точкой входа в корпоративную сеть.

Одним из наиболее эффективных способов защиты являются межсетевые экраны веб-приложений (Web Application Firewall, WAF) – защитные экраны, предназначенные для блокировки атак на веб-приложения [3, 4]. К числу таких атак относятся SQL-инъекции, межсайтовое написание сценариев (скриптов) на интерпретируемых языках программирования, удаленное выполнение кода, подбор паролей и обход авторизации, а также атаки, использующие неизвестные (zero-day) уязвимости.

WAF обеспечивает защиту, выполняя мониторинг содержимого веб-страниц независимо от того, на чем они написаны, будь то POST-forms, HTML или Java Script. При этом происходит фильтрация потенциально вредоносных запросов по протоколам HTTP/HTTPS.

Таким образом, обход WAF является очень опасным и критичным способом кибератак на компьютерную сеть. Общая идея, лежащая в основе реализации обхода WAF, заключается в том, что запрос приводится к виду, в котором он остается понятным для человека, атакующего веб-приложения, но при этом становится непонятным для WAF. Различные специальные символы могут нарушать логику работы WAF. Однако при этом они понятны самому серверу. Существуют различные варианты использования специальных символов. Так, их можно закодировать в URL-строках или представить в иных кодировках. Кроме того, можно вставлять спецсимволы в запрос без какой-либо кодировки, т.е. в raw-формате. Этот способ может оказаться довольно неожиданным для WAF.

Как правило, в WAF присутствует модуль, отвечающий за обнаружение вредоносных запросов с помощью обученного классификатора. Однако главной проблемой классификаторов является их способность выявлять лишь те атаки, на которых их обучили первоначально. Для этого необходим достаточно большой набор данных (data set), включающий в себя все известные атаки. В противном случае может возникнуть (и случается это довольно часто) необходимость дообучения или переобучения модели в режиме реального времени, что негативно сказывается на способности WAF неизвестные ранее атаки.

В статье предлагается новый метод конструирования межсетевых экранов веб-приложений и обнаружения коллизий в трафике, вызванных попытками злоумышленников реализовывать кибератаки, нацеленные на обход WAF. Под термином «коллизии» в работе мы понимаем именно аномально-пиковые отклонения трафика, вызванные целевым информационно-техническим воздействием со стороны злоумышленника, с умыслом получения доступа к ресурсам сети. Этот метод основан на одновременном использовании двух разновидностей нейронных сетей: нейронной сети с долгой краткосрочной памятью (Long Short-Term Memory, LSTM) и автоэнкодера.

Так как нейронная LSTM-сеть относится к классу рекуррентных, то она обладает возможностью хранить информацию, что дает дополнительные преимущества при обнаружении коллизий в различных наборах данных. Автоэнкодеры, как правило, используются для снижения размерности входных данных. Поэтому совместное использование нейронных LSTM-сетей и автоэнкодеров представляется достаточно перспективным направлением в области защиты от кибератак, нацеленных на обход WAF.

Описание LONG SHORT-TERM MEMORY. Предлагаемая архитектура нейронной сети представлена на рис. 1. Автоэнкодер состоит из трех наборов слоев: (1) *Input Layer*, который принимает входной сигнал $X=\{x_1,x_2,\ldots,x_t,\ldots,x_w\}$; (2) несколько *Hidden Layers* {LSTM-1, …, LSTM-n}, каждый из которых образован ячейками LSTM нейронной сети (количество скрытых слоев равно n); (3) *Output Layer*, выход которого $\tilde{X}=\{\tilde{x}_1,\tilde{x}_2,\ldots,\tilde{x}_t,\ldots,\tilde{x}_w\}$ является выходом всей предлагаемой архитектуры [5].

Автоэнкодер, являясь нейронной сетью прямого распространения, восстанавливает входной сигнал на выходе. Внутри автоэнкодера имеются один или несколько скрытых слоев, с помощью которых задается код, описывающий некоторую модель. Поэтому автоэнкодер может быть представлен как нейронная сеть, состоящая из двух частей: кодировщика и декодировщика. С помощью кодировщика происходит сжатие данных, с помощью декодировщика –-восстановление. После обучения автоэнкодер может воспроизводить практически без ошибок или с

небольшой ошибкой нормальные наблюдения, т.е. данные, в которых отсутствуют коллизии. Входной сигнал восстанавливается с ошибками из-за потерь при кодировании, и, чтобы их минимизировать, для обучения автоэнкодера отбираются наиболее важные признаки.

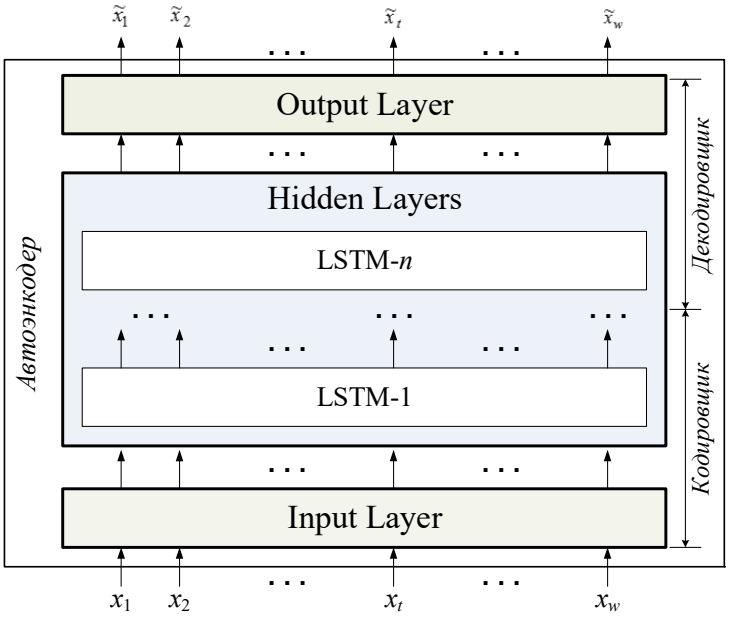

Рис. 1. Общая архитектура нейронной сети

Принцип использования автоэнкодера состоит в том, чтобы научить его максимально точно восстанавливать данные, на которых происходило обучение. Тогда, если на вход автоэнкодера ввести аномальный образец (набор данных, не принадлежащий к обучающей выборке), автоэнкодер с высокой долей вероятности не сможет отнести аномальные данные к какому-либо предварительно обученному классу. Иными словами, автоэнкодер будет пытаться предсказать/реконструировать выявленные коллизии в данных. Однако для данных с отклонениями ошибка восстановления данных будет намного выше, чем для нормальных данных.

В качестве скрытых слоев автоэнкодера могут использоваться нейронные сети различных типов. В предлагаемой архитектуре применяются LSTM-ячейки нейронной сети, которая является разновидностью рекуррентных нейронных сетей. Особенностью LSTM-нейросетей, как и всех рекуррентных сетей в целом, является их способность сохранять во времени свое состояние, которое выражается некоторым числовым значением. Это повышает способность нейронных сетей с LSTM предсказывать поведение анализируемой системы, а также обнаруживать отклонения в больших наборах данных в случаях, когда для процесса обучения отводилось недостаточно времени. Иначе говоря, предполагается, что LSTM-нейросети позволяют построить систему обнаружения коллизий, ориентированную на работу с большими данными, а также на реальный или близкий к реальному режим времени.

Структура скрытого LSTM-слоя в предлагаемой архитектуре представлена на рис. 2, где показаны связанные между собой LSTM-ячейки $(t-1)$, t и $(t+1)$.

На вход t LSTM-ячейки поступают следующие сигналы:

сигнал x_t со входного уровня;

сигнал h_{t-1} с предыдущей ячейки своего скрытого слоя;

сигнал состояния S_{t-1} с предыдущей ячейки.

На выходе t LSTM-ячейки формируются следующие сигналы:

сигнал h_t, который поступает на последующую ячейку своего скрытого слоя и на t LSTM-ячейку следующего скрытого слоя;

сигнал состояния S_t, поступающий на последующую ячейку.

В LSTM-ячейке можно выделить три внутренних слоя (*gates)*: (1) *Forget Gate*; (2) *Input Gate*; (3) *Output Gate*.

Forget Gate предназначен для принятия решения о хранении либо удалении информации о предыдущем состоянии S_{t-1}. С использованием σ-функции активации, веса W_f и смещения b_f в этом слое формируется вспомогательная функция f_t:

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right), \tag{1}$$

где $[h_{t-1}, x_t]$ – операция конкатенации (объединения) сигналов h_{t-1} и x_t.

Input Gate предназначен для того, чтобы обновлять значения состояния C_{t-1} и формировать новое потенциальное значение состояния $\tilde{C}_t$. В этом слое с использованием σ-функции активации, tanh-функции активации (tanh – гиперболический тангенс), весов W_i и W_C, а также смещений b_i и b_C формируются вспомогательные функции i_t и $\tilde{C}_t$, а также новое значение C_t:

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right); \tag{2}$$

$$\tilde{C}_t = \tanh\left(W_C\left[h_{t-1}, x_t\right] + b_C\right); \tag{3}$$

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t\,. \tag{4}$$

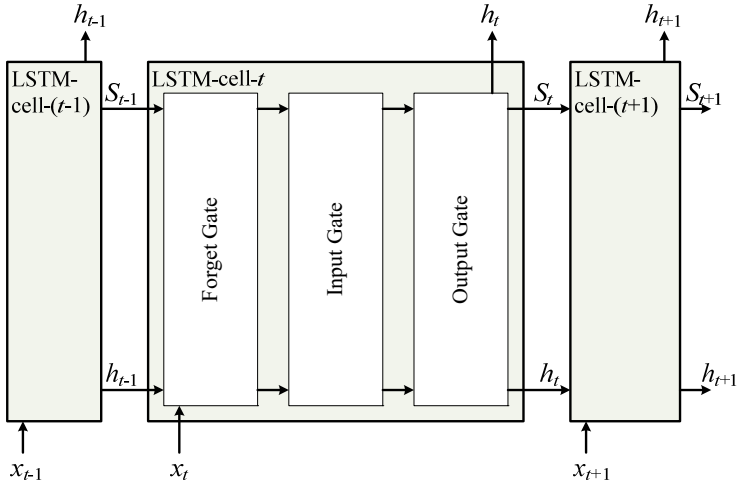

Рис. 2. Структура LSTM-слоя

Output Gate служит для того, чтобы реализовать возможность пропускать выходное значение состояния через некоторый дополнительный фильтр. С использованием σ-функции активации, tanh-функции активации, веса W_o и смещения b_o в этом слое формируются вспомогательная функция o_t и выходной сигнал LSTM-ячейки h_t:

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right); \tag{5}$$

$$h_t = o_t \tanh\left(C_t\right). \tag{6}$$

Таким образом, свойство рекуррентности позволяет нейронной сети с LSTM использовать полученные ранее результаты и проводить анализ предсказаний. В итоге будущее решение ставится в зависимость не только от первичного глубокого обучения нейронной сети с LSTM, но и от результатов ее дальнейшей работы в потоке данных. Это свойство делает нейронную сеть с LSTM особенно подходящими для анализа нестационарных временных рядов, которые изменяются с течением времени, поэтому их успешно используют в таких задачах, как распознавание речи и перевод текста. В настоящей работе исследуется возможность применения нейронной сети с LSTM для обнаружения коллизий сетевого трафика.

Проверка эффективности предложенного метода. Для формирования набора данных, который годился бы для обучения предложенной архитектуры нейронной сети и оценки ее возможности обнаруживать аномалии, необходимо сначала смоделировать кибератаки, нацеленные на обход WAF. Общий смысл нахождения способов обхода WAF заключается в приведении запроса к виду, в котором он остается понятным для атакуемого веб-приложения, но при этом не понятен или кажется безобидным для WAF. Для этой цели могут использоваться различные спецсимволы, которые могут нарушать логику работы WAF и при этом быть понятными серверу.

На рис. 3 представлен пример обычного запроса и ответа на него сервера. Запрос представляет собой попытку передать методом POST параметр “select”. Как видно из рисунка, параметр “select” был заблокирован WAF.

Рис. 3. Попытка отправить запрос без преобразования параметра

На рис. 4 представлен другой запрос, в котором параметр “select” закодирован спецсимволами, взятыми из кодировочной таблицы Unicode. Как видно из рисунка, если передать параметр “select” в кодировке Unicode, то WAF его не обнаруживает.

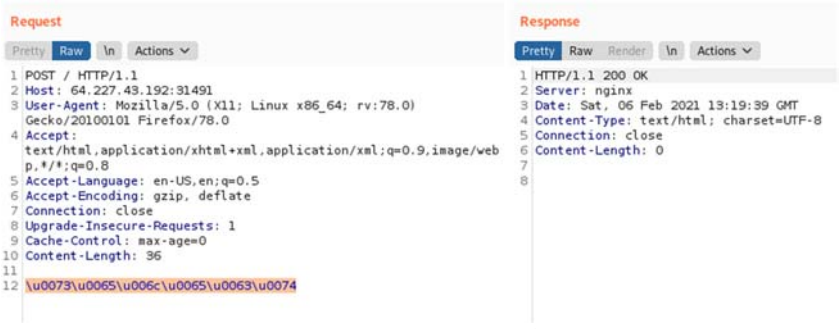

Рис. 4. Попытка отправить запрос с кодированием параметра

Передадим теперь на сервер таким же способом (т.е. с использованием кодировки Unicode) параметр “union select sleep(10)#” (рис. 5). Запрос также успешно обошел WAF и поступил на обработку на сервер.

Рис. 5. Пример обхода WAF зашифрованным запросом

Запрос, который был направлен на обработку на сервер, выглядит следующим образом:

SELECT note FROM notes WHERE assignee = “union select sleep(10)#”.

Сервер дал ответ на этот запрос с задержкой в 10 с. Тем самым была реализована эффективная кибератака вида “time base sql injection”. Пример реализации этой кибератаки, осуществляющей обход WAF, представлен на рис. 6.

Для тестирования предлагаемого подхода установим систему управления содержимым сайта Wordpress и развернем Nemesida WAF Free. При попытке реализации кибератаки с передачей полезной нагрузки в виде URL encode получаем ошибку запрета доступа к странице (рис. 7).

Для обучения автоэнкодера был сформирован набор данных, состоящий из пользовательских запросов (рис. 8). Данные в наборе данных максимально полно передают пользовательское поведение и исключают коллизии. Количество записей в наборе данных – 28 000.

Рис. 6. Пример кибератаки обхода WAF

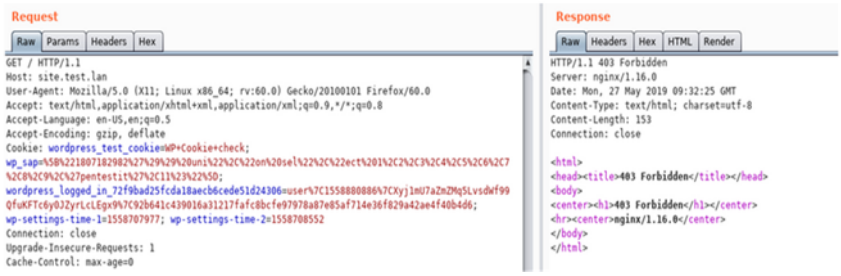

Рис. 7. Пример отправки зашифрованной полезной нагрузки

Рис. 8. Набор данных, состоящий из нормальных данных

Автоэнкодер обучался только отрицательным (т.е. нормальным) наблюдениям. Для того чтобы сформировать обучающий набор данных, из исходного набора данных были удалены все аномальные (положительные) случаи (рис. 9).

```
['<START>GET /0 or 1=1<STOP>',
 '<START>GET /limit<STOP>',
 "<START>GET /);waitfor delay '0:0:__TIME__'--<STOP>",
 '<START>GET /||UTL_HTTP.REQUEST<STOP>',
 '<START>GET /post/new/distinct<STOP>',
 "<START>POST /post/new/title=||'6&text=Это+мой+первый+пост!<STOP>",
 "<START>GET /post/1/x' AND 1=(SELECT COUNT(*) FROM tabname); --<STOP>",
 "<START>GET /') or ('a'='a<STOP>"]
```

Рис. 9. Пример записи аномальных данных в наборе данных

Тестовый набор данных, сформированный путем добавления в нормальный набор данных записей коллизий, использовался для экспериментальной оценки предложенного подхода.

Поскольку протокол HTTP является текстовым, для классификации текста использовались современные текстовые классификаторы. Было решено опробовать модель с LSTM-архитектурой.

Так, в LSTM-архитектуре на естественном языке применяется векторное представление слов. Однако было неясно, какие слова встречаются в искусственном языке, например в HTTP.

Так как готовые представления не годятся для выполнения поставленной задачи, были использованы простые отображения символов в числовые коды с несколькими внутренними маркерами. Это отображение осуществлялось следующим образом. Сначала символы, встречающиеся в наборе данных с отрицательными наблюдениями, заменялись на числовой эквивалент, который не имеет самостоятельного смысла для внешнего или внутреннего применения (данную процедуру можно называть токенизацией). Затем в соответствии с токенизацией слова переводились в последовательности (рис. 10). Важным обстоятельством здесь является то, что все последовательности должны быть одной длины. Если запрос меньше длины последовательности, оставшиеся символы заполняются нулями.

```
data_train = pad_sequences(X_train_clear_sequences, maxlen=max_len_str)
data_test = pad_sequences(X_test_clear_sequences, maxlen=max_len_str)

data_train

array([[ 0,  0,  0, ...,  4, 15, 29],
       [ 0,  0,  0, ...,  4, 15, 29],
       [ 0,  0,  0, ...,  4, 15, 29],
       ...,
       [ 0,  0,  0, ...,  4, 15, 29],
       [ 0,  0,  0, ...,  4, 15, 29],
       [ 0,  0,  0, ...,  4, 15, 29]], dtype=int32)
```

Рис. 10. Пример представления последовательностей

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 1, 999)]          0
_________________________________________________________________
lstm (LSTM)                  (None, 1, 499)            2992004
_________________________________________________________________
bottleneck (LSTM)            (None, 249)               746004
_________________________________________________________________
repeat_vector (RepeatVector) (None, 1, 249)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 1, 249)            497004
_________________________________________________________________
lstm_2 (LSTM)                (None, 1, 499)            1495004
_________________________________________________________________
time_distributed (TimeDistri (None, 1, 999)            499500
=================================================================
Total params: 6,229,516
Trainable params: 6,229,516
Non-trainable params: 0
_________________________________________________________________
```

Рис. 11. Автоэнкодер с LSTM-ячейками

Окончательная модель автоэнкодера с LSTM-ячейками, которая исследовалась на обнаружение аномалий, вызванных кибератаками, нацеленными на обход WAF (рис. 11), имеет пять скрытых слоев, среди которых первые два и последние два слоя представлены LSTM-ячейками. Средний скрытый слой – обычный. Первые два скрытых LSTM-слоя образуют кодировщик, последние три скрытых слоя – декодер. Количество ячеек в каждом слое и количество настраиваемых при обучении параметров показаны на рис. 11. Общее количество настраиваемых параметров равно 6 229 516.

```
Trial 70 Complete [00h 03m 10s]
val_accuracy: 0.0800832062959671

Best val_accuracy So Far: 0.26791471242904663
Total elapsed time: 01h 58m 36s

Search: Running Trial #71

Hyperparameter    |Value             |Best Value So Far
dropout           |0.1               |0.3
activation        |sigmoid           |relu
learning_rate     |0.1001            |0.0051
tuner/epochs      |10                |10
tuner/initial_e...|4                 |4
tuner/bracket     |2                 |3
tuner/round       |1                 |2
tuner/trial_id    |ab4c611fc614e2e...|3e9e8e824784b6c...
```

Рис. 12. Перебор гиперпараметров

Результаты полного перебора гиперпараметров (рис. 12) говорят о том, что наилучшими гиперпараметрами являются: функция активации relu, learning rate = 0.0051, dropout = 0.3. Поэтому они использовались далее при обучении модели.

Результат обучения модели на 50 эпохах представлен на рис. 13. После завершения каждой эпохи лучшая модель сохранялась в качестве контрольной точки, которую затем можно было загрузить.

```
Epoch 42/50
66/66 [==============================] - 2s 23ms/step - loss: 0.5432 - accuracy: 0.8492 - val_loss: 0.8284 - val_accu
racy: 0.8179
Epoch 43/50
66/66 [==============================] - 1s 22ms/step - loss: 0.5251 - accuracy: 0.8462 - val_loss: 0.8189 - val_accu
racy: 0.8549
Epoch 44/50
66/66 [==============================] - 1s 22ms/step - loss: 0.5998 - accuracy: 0.8576 - val_loss: 0.7851 - val_accu
racy: 0.8791
Epoch 45/50
66/66 [==============================] - 1s 22ms/step - loss: 0.4763 - accuracy: 0.8704 - val_loss: 0.7558 - val_accu
racy: 0.8563
Epoch 46/50
66/66 [==============================] - 1s 22ms/step - loss: 0.4883 - accuracy: 0.8597 - val_loss: 0.7162 - val_accu
racy: 0.8634
Epoch 47/50
66/66 [==============================] - 1s 22ms/step - loss: 0.4538 - accuracy: 0.8733 - val_loss: 0.6976 - val_accu
racy: 0.8819
Epoch 48/50
66/66 [==============================] - 1s 22ms/step - loss: 0.4141 - accuracy: 0.8742 - val_loss: 0.7180 - val_accu
racy: 0.8634
Epoch 49/50
66/66 [==============================] - 1s 22ms/step - loss: 0.3743 - accuracy: 0.8558 - val_loss: 0.7525 - val_accu
racy: 0.8677
Epoch 50/50
66/66 [==============================] - 1s 22ms/step - loss: 0.4019 - accuracy: 0.8689 - val_loss: 0.6941 - val_accu
racy: 0.8819
```

Рис. 13. Результаты обучения модели на 50 эпохах

На рис. 14 показаны результаты обнаружения коллизий в наборе данных. Аномальные запросы выделены красным цветом, а нормальные – синим. Горизонтальная черная прямая показывает максимальную ошибку, которая возникает при обучении автоэнкодера.

а

б

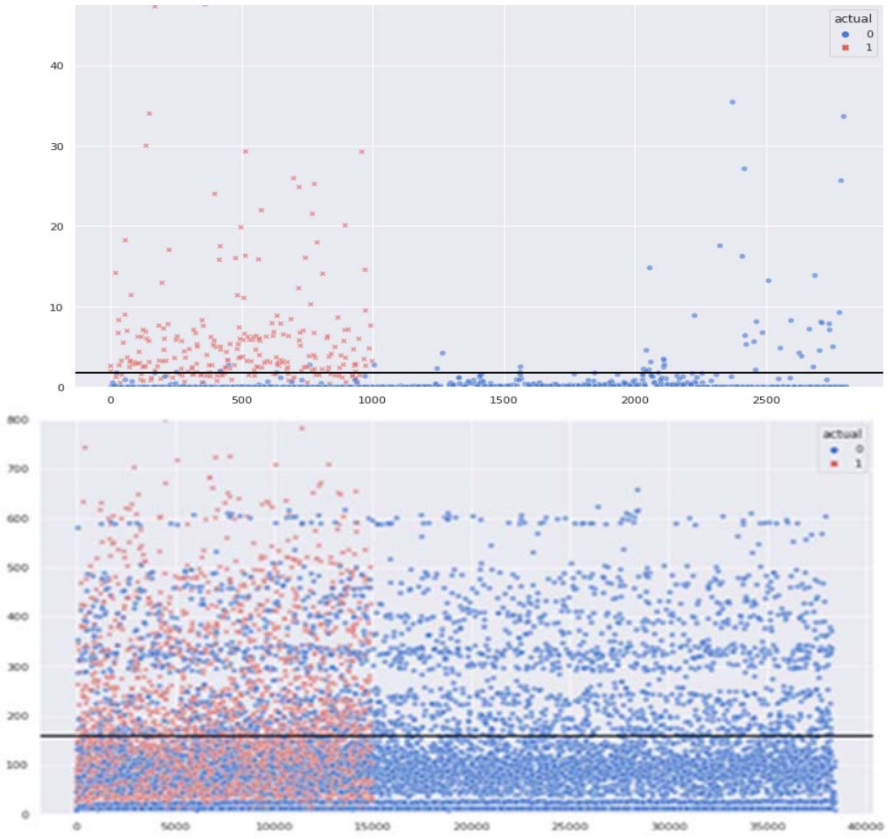

Рис. 14. Результаты обнаружения коллизий сетевого трафика: а – разделение запросов на слова; б – разделение запросов на символы

На рис. 14, *а* показаны результаты разделения запросов на слова. В этом случае автоэнкодер ведет себя наиболее агрессивно, выявляя максимальное количество аномалий. Несмотря на то что черная прямая находится на уровне 2, автоэнкодер в то же время блокирует максимальное количество запросов.

На рис. 14, *б* приведены результаты разделения запросов на символы. Здесь автоэнкодер достаточно лояльно выявляет отклонения. Черная прямая находится на уровне 170. Блокируются только те запросы, которые лежат выше указанной линии.

На рис. 15 представлены зависимости точности (доли аномалий, принадлежащих к данному классу, относительно всех аномалий, которые система отнесла к этому классу) и полноты обнаружения коллизий (доли найденных аномалий, принадлежащих к классу, относительно всех запросов этого класса в тестовой выборке). Пересечение этих двух кривых позволяет найти оптимальное значение для порога.

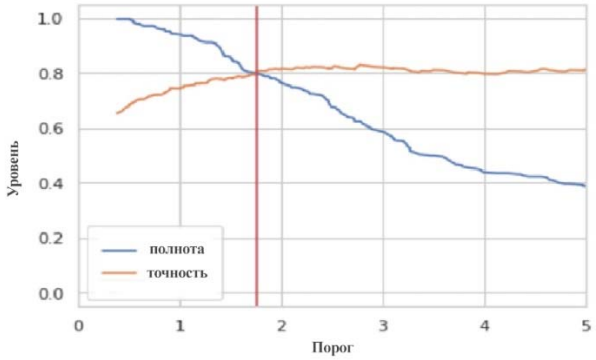

Рис. 15. Зависимости точности и полноты обнаружения атак от порога, при котором легитимные запросы не блокируются

ЗАКЛЮЧЕНИЕ

Анализ экспериментальных результатов показал, что способность нейронных сетей с LSTM не только обучаться, но и обрабатывать коллизии, связанные с аномальным поведением сетевого трафика, позволяет заблаговременно предупреждать о проникновениях в компьютерные сети извне.

В предложенном автором подходе обязательным этапом является процесс обучения на контрольной выборке. По скорости обнаружения атак он не уступает сигнатурному методу, однако имеет более высокую точность обнаружения известных атак и хорошую точность обнаружения неизвестных атак. При этом доля ложных срабатываний крайне низкая.

Список литературы

1. Cyber Security Trends You Can’t Ignore in 2021. [Элекктронный адрес]. -URL: https://purplesec.us/cyber-security-trends-2021(last_accessed:2021.04,24

2. Cybersecurity Statistics and Trends for 2021. –[Электронный адрес]. -URL: https://www.varonis.com/blog/cybersecurity-statistics(last_accessed:2021.04.24).

3. Clincy, V. Web Application Firewall: Network Security Models and Configuration / V. Clincy, H. Shahriar // Proceedings of the 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). 2018. P. 835-836.

4. Baddar Sh.Al-H. Anomaly Detection in Computer Networks: A State-of-the-Art Review / Sh.Al-H. Baddar, A. Merlo, M. Migliardi // J. Wirel. Mob. Networks Ubiquitous Comput. Dependable. Appl. 5 (2014). P. 29-64.

5. Gers F. Learning precise timing with LSTM recurrent networks / F. Gers, N. Schraudolph, J. Schmidhuber // Journal of Machine Learning Research. 2002. Vol. 3. P. 115-143.

Крибель Александр Михайлович, адъюнкт, nemo4ka74@gmail.com, Россия, Санкт-Петербург, Военная академия связи им. С.М.Буденного

METHODOLOGY FOR DETECTING NETWORK TRAFFIC COLLISIONS

A.M. Kribel

The possibilities of using artificial neural networks of complex architecture (in particular, based on Long Short-Term Memory (LSTM)) are investigated to detect network traffic collisions caused by cyber attacks using vulnerabilities of web application firewalls (Web Application Firewall, WAF),

which are very difficult to detect by other means. Experimental results based on the generated data set confirmed the high efficiency of the development: the technique allows detecting cyberattacks in real time.

Key words: computer network, collision detection, cyberattack, web application firewall, neural network, LSTM, autoencoder.

Kribel Alexander Mikhailovich, postgraduate, nemo4ka74@gmail.com, Russia, St. Petersburg, Military Academy of Communications named after S.M.Budyonny

УДК 004.94:69
DOI: 10.24412/2071-6168-2021-12-191-194

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ ВОЗДУШНЫХ МАСС В ГРУППЕ ГОРОДСКИХ ЗДАНИЙ

Т.А. Алексеева

Приводится алгоритм процесса моделирования движения воздушных масс в районе, в котором построена группа зданий переменной этажности. Оцениваются скорости движения воздуха в рассматриваемом районе в сечениях, расположенных на разной высоте относительно уровня земли.

Ключевые слова: компьютерное моделирование, город, строительство, анализ данных, исследование, математическое моделирование, аэродинамика, строительство.

Ученым, инженерам, конструкторам и другим лицам предоставляются широкие возможности для исследования различных явлений и процессов, во многом благодаря техническому прогрессу и появлению, в связи с этим, специальных программных средств, на базе которых возможно проведение компьютерных и математических моделирований [1,2]. Последующий анализ результатов моделирований позволяет выявить не только закономерности протекания явлений, но и улучшить (оптимизировать), в частности, процессы или конструкции. Большая часть программ, позволяющих трехмерно моделировать основаны на способе математического анализа под названием метод конечных элементов [3,4], который позволяет замоделировать практически любую физическую и мультифизическую задачу, например, движение воздушных масс в районе города, с целью определения скорости ветра на разных высотах, его траектории. Для этого было выполнено моделирование в программе Ansys [5-7], которое проводилось в следующей последовательности:

1. Строилась трехмерная модель района со зданиями, на которые будет воздействовать ветер. Модель создавалась в система автоматизированного проектирования, хотя возможно ее построение во встроенных подпрограммах Ansys.

2. Проводилась инверсия полученной трехмерной модели для создания области, в которой дует ветер.

3. Полученная инверсионная модель разбивалась на части (конечные элементы) с заданным размером одного элемента. В данном случае для уменьшения времени расчета была построена сетка с размером элемента равным не более 0,5 м по одной из сторон. С учетом масштабов модели можно утверждать, что такое допущение не скажется значительным образом на качестве и точности расчета.

4. На область, разбитую на части, задавались начальные условия: температура воздуха, физико-химические свойства воздуха, направление и скорость ветра и пр.

5. Запускался расчет согласно заданным начальным значениям, который продолжался около 6,5 часов.