Применение глубокого обучения для выявления и классификации DGA доменов

Е.В. Дюльдин, К.С. Зайцев

Аннотация. Целью настоящей работы является исследование методов выявления вредоносных доменов, сгенерированных с помощью алгоритмов DGA. Для решения данной задачи предлагается создание архитектуры глубокого обучения на основе фреймворка Tensorflow, и реализации слоёв на основе библиотеки Keras. Для обучения, тестирования и валидации полученной архитектуры были использованы данные, сгенерированные на основе 25 известных алгоритмов по генерации DGA, и легитимные данные, полученные из Top Alexa. Используя полученные данные, были проведены сравнения предложенной архитектуры нейронной сети с известными реализациями архитектур машинного обучения по классификации DGA доменов. Целевой метрикой, по которой сравнивали качество классификации, была выбрана f-мера с параметром - вес точности в метрике (β) равным 0.4, что позволило выбирать модель с наибольшей точностью предсказания. Полученные результаты подтвердили эффективность предложенного решения. Итогом работы стало создание эффективной архитектуры машинного обучения, применяемой для классификации вредоносных DGA доменов.

Ключевые слова **– генерация доменов, DGA, классификация, машинное обучение, глубокое обучение, Keras**

I. ВВЕДЕНИЕ

В настоящее время каждая крупная IT-компания имеет в наличии огромные вычислительные мощности для решения прикладных задач, серверные мощности включают кластеры, на которых развёрнута инфраструктура компании. При заражении рабочей машины, злоумышленник хочет не только остаться в рабочей сети, но также передавать и получать данные из внешних источников. Для отправки программного кода на заражённую машину генерируется множество нелегитимных доменов (доменных имен), которые отправляют запросы на DNS сервер компании с целью передачи информации на заражённую машину об адресах, к которым она должна подключаться для получения дальнейших команд для запуска вредоносного кода в контурах компании [1].

Статья получена 20 мая 2022.

Дюльдин Евгений Владимирович, Национальный Исследовательский Ядерный Университет МИФИ, магистрант, Zhecos1@yandex.ru

Зайцев Константин Сергеевич, Национальный Исследовательский Ядерный Университет МИФИ, профессор, KSZajtsev@mephi.ru

Для решения данной проблемы раньше использовались идеи, называемые «белый лист» и «чёрный лист». Белый лист основывался на списке легитимных доменов, которым можно было подключаться к внутренним контурам компании. Эта идея показывала себя хорошо, если в компании точно знали какие домены будут запрашивать доступ к их системе. Если же специфика компании предполагала, что любой пользователь или другие компании могут отправлять запросы на доступ к DNS серверу, то ручная проверка доменов занимала слишком много времени [2]. Параллельно обычно использовались идеи «чёрного списка», которые наоборот запрещали группам доменов пройти DNS сервер компании. При множественных генерациях доменных имён этот тип алгоритма тоже не справляется с задачей выявления и бинарной классификации DGA доменов, так как не способен быстро распознавать разнообразные комбинации нелегитимных доменов и вернуть их на сторону DNS сервера [3]. В дальнейшем для классификации доменных имён использовались классические статистические алгоритмы, основанные на подсчёте численных признаков в каждом доменном имени, так же в группу классических алгоритмов можно отнести полиномиальные методы, считавшиеся эффективными до использования методов машинного обучения. Следующим шагом улучшающим точность моделей классификации стало использование методов классического машинного обучения совместно с алгоритмами DNS защиты. Сама методика DNS защиты была основана на использовании индикаторов, которые блокировали вредоносный код и уже после заражения выясняли откуда пришёл вредоносный домен. Эта была база знаний и группа методов по обнаружению нелегитимных DGA доменов в сети [4]. Данная технология поставлялась с набором мощных серверов, так как накладывались определённые затраты на хранение и обработку больших данных. Совместно использовались подходы, основанные на регрессионных моделях и RandomForest, которые зависимы от обучающих данных обучения и признаках, используемых для обучения модели. Поэтому имея базу алгоритмов генерации DGA доменов, можно было создать модель машинного обучения, которая выявляла зависимости в данных, но при появлении новых алгоритмов, приходилось генерировать данные уже на их основе и переобучать ранее подобранную модель. Это накладывало трудности на получение исходного кода новых вредоносных алгоритмов, а

также на поиск новых более эффективных моделей. Так, например, для RandomForest приходилось заново подбирать гиперпараметры из-за энтропийной особенности распределения данного алгоритма. Перечисленные трудности приводили к сложному цикл поддержки моделей, основанных на классическом машинном обучении. Хотя подходы, основанные на классическом машинном обучении, активно применяются и в настоящее время, так как имеют хорошие показатели по скорости и времени предсказания. Но, из-за увеличивающегося темпа и уровня киберугроз эти методы не всегда могут справиться с потоком угроз, что приводит к идее использования при классификации доменных имён подходов, основанных на глубоком обучении.

В научных публикациях приведены подходы, демонстрирующие, как определённые слои нейронных сетей могут улавливать закономерности в генерации DGA доменов. Но в описанных реализациях применяются одиночные слои или реализации, с очень сложными последовательными архитектурами, использующими генерацию признаков на основе Embedding слоя, не рассматривая при этом поиска других зависимостей в данных для генерации еще более сложных признаков [5].

В настоящей статье предлагается для решения задачи классификации доменных имён использовать параллельную архитектуру, основанную на совмещении обобщающих способностей слоёв глубокого обучения. Так же будет использована генерация признаков, основанных только на имени домена с выделением энтропийных признаков и слоёв обработки данных Embedding.

II. АЛГОРИТМЫ ВЫЯВЛЕНИЯ DGA

Рассмотрим основные подходы для бинарной классификации доменов на легитимные и нелегитимные DGA.

а) *LSTM-like*. Основной проблемой классического машинного обучения и алгоритмов, основанных на подсчёте количественных признаков в доменном имени, было то, что при получении данных, эти алгоритмы не выявляли закономерности между соседними символами и символами, находящимися более, чем на один левее или правее от выбранного. Для решения этой задачи применяли слои RNN, предшествующие LSTM слоям, которые могли выявлять зависимости между соседними символами, но обладали слишком высокой скоростью роста градиента, что приводило к некорректной классификации на доменах, состоящих более, чем из пяти символов [6].

Слой LSTM позволил использовать возможность выделения долговременных зависимостей.

LSTM архитектура имеет вид множества одинаковых ячеек, связанных множественными преобразованиями, каждая ячейка включает в себя входные данные и информацию о ранее полученных результатах, которые передаются на второй вход ячейки. Далее данные собираются в матрицу, которая проходит через несколько этапов, чтобы, а конечном итоге получить два выхода, первый из которых выводит полученный из ячейки результат в иные слои архитектуры, а второй позволяет передать накопленные данные дальше по слою.

Ключевым компонентом данного слоя является, горизонтальный проход, который участвует в линейных преобразованиях не меняя градиент, что помогает избежать проблем с быстрым ростом или затуханием градиента на более поздних шагах.

Во время прохождения ячейки, информация проходит через несколько врат, которые позволяют добавить новые зависимости и удалить неактуальные зависимости из наших данных.

Для определённости опишем формально каждый из шагов обработки данных в слое LSTM:

1. На первом шаге предстоит выбрать информацию, которая будет забыта. Это реализуется с помощью сигмоидального слоя, возвращающего промежуток от 0 до 1 и называемый “слоем забвения”:

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

2. На втором шаге надо решить какая информация останется в ячейке. В данном случае будет произведено две операции, сначала выполнится операция с использованием сигмоидального слоя для выборки кандидатов (2), затем будет применён слой гиперболического тангенса и построен вектор новых значений (3):

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i), \quad (2)$$

$$C_t = tanh(W_c * [h_{t-1}, x_t] + b_c), \quad (3)$$

3. На последнем шаге мы объединяем полученные ранее слои для получения выходных вердиктов из ячейки:

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o), \quad (4)$$

$$h_t = O_t * \tanh(C_t), \quad (5)$$

При реализации этого слоя стоит учитывать, что можно составлять различные вариации входных ворот, заменяя и добавляя новые и добиться лучшей скорости работы, также как при использовании GRU подобных архитектур [7].

Проходя между всеми вратами, информация фильтруется и часть новой добавляется для прохода в новые ячейки, часть удаляется, в основе этого, как было описано лежит сигмовидная функция и гиперболический тангенс.

Такой подход описывает только один слой, т.к. архитектура подобных сетей строится на одиночном входном слое Embedding, позволяющим выделять из текстовых данных зависимости и преобразовывать из в векторное представление заданной размерности. Embedding в данных архитектурах используется, как двухслойная нейронная сеть без функции активации, что помогает изменить размерность данных не теряя информации. Так же слой Embedding позволяет в отличии от SVD

реализаций выполнять масштабирование для выявления из текстовых данных признаков, которые основаны только на алгоритмических преобразованиях собственных векторов [8].

После слоя Embedding следует слой LSTM, принимающий векторные выходы и формирующий ячейки для обработки последовательности. Далее выходы слоя LSTM поступают на полносвязные Dense слои, которые, используя полученную активацию, пропускают через себя данные и затем на выходе взвешивают их с помощью функции softmax.

Результатом работы становится вероятностная классификация. В этом алгоритме подразумевается использование категориальной энтропии.

Такой подход зарекомендовал себя, как достаточно надёжный при обучении на алгоритмах, генерирующих домены на основе случайного ядра, но алгоритмы, основанные на текстовом массиве, могут обойти это архитектурное решение.

Для лучшего понимания, архитектура описанного алгоритма приведена ниже (рис. 1).

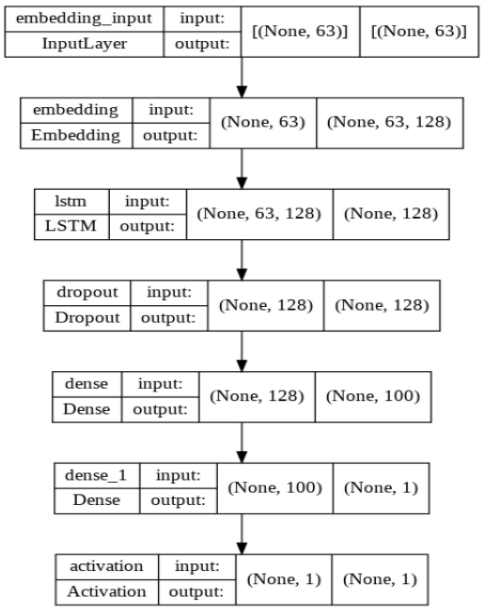

Рис.1. LSTM-like архитектура общего использования.

б) *Conv-like*. Для решения задачи классификации доменов, основанных на текстовых признаках, вспомним архитектуры, использующие свёрточные слои.

Этот подход пришёл из машинного зрения, и позволяет на основе многочисленных свёрток с определёнными фильтрами выявлять зависимости из текстовых данных, преобразованных в численные векторы. При их использовании создаётся одномерное ядро, проходящееся по всему массиву данных и преобразующее его в меньшую размерность с помощью обученных фильтров.

На каждом шаге такого слоя мы выбираем ядро, с которым будем проходить по оставшемуся вектору.

Далее производим свёртку данного ряда с помощью обученных фильтров.

Последним шагом полученный вектор передаём в слои, идущие за сверточным слоем ConvD. В базовом случае используются два полносвязных Dense слоя, первый из которых имеет меньшую размерность входного вектора, а второй состоит из двух нейронов, при использовании категориальной энтропии в качестве ошибки.

Для доменов, основанных на текстовых данных, такой слой - это эффективное решение. Но алгоритмы генерации доменов усложняются, что приводит к ухудшению точности этого алгоритма, также, как алгоритмов, основанных на LSTM слоях.

Для лучшего понимания описанной архитектуры рассмотрим рисунок ниже (рис. 2).

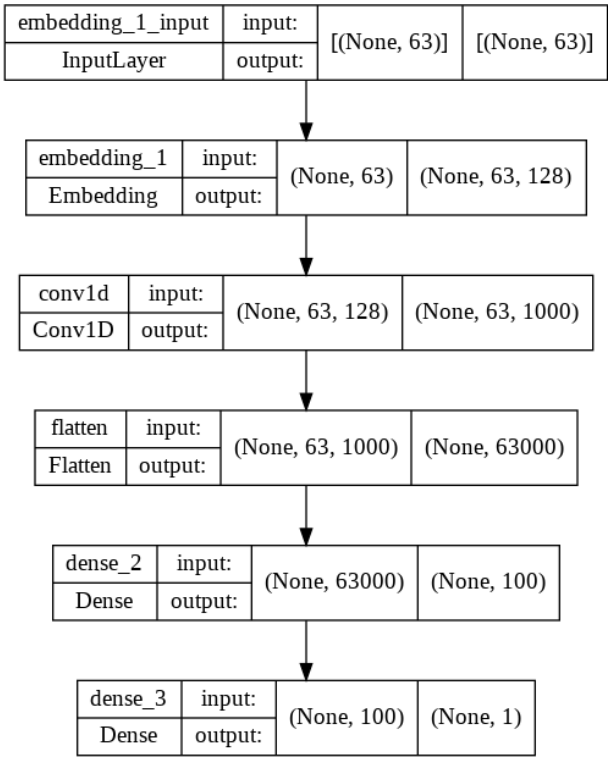

Рис.2. Conv-like архитектура общего использования.

в) Последовательное совмещение. Для сложных алгоритмов генерации доменов используется параллельный алгоритм совмещения LSTM-like и Conv-like слоёв, что даёт прибавку точности, но не даёт эффективно распараллелить работу архитектурного решения.

Такой подход не отличается точностью на алгоритмах, совмещающих подход генерации на основе даты и времени, получаемых в реальном времени, с подходом генерации на основе текстовых наборов данных со случайным ядром сдвига на каждой итерации.

Для повышения точности и решению задачи классификации сгенерированных доменов разным образом рассмотрим создание параллельной сети с модификацией архитектуры и используемых признаков для обучения модели.

III. ПОДГОТОВКА ДАННЫХ

Для тестирования и сравнения результатов работы построенных архитектур был проанализирован список доменов из алгоритмов, находящихся в открытом доступе, и доменов из Top Alexa, который, как известно, является агрегатором посещений доменов в сети Internet. Эти наборы данных изначально были объединены в одну большую обучающую выборку, где легитимные домены составляли 1 000 000 доменов, а выборка из нелегитимных сгенерированных доменов включала 1 200 000 доменов. В дальнейшем эта выборка будет обогащена доменами из открытых источников и будет включать по 2 000 000 доменов в каждой из групп легитимных и нелегитимных доменов.

Готовый массив данных был разделен на три меньших массива в отношении 70%, 20%, 10%, где первый массив использовался для обучения модели, второй - для валидации и выбора наилучших архитектур после обучения, и третий – для валидации на реальных данных. Так же стоит отметить, что для воспроизводимости разбиения данных, были использованы значения равные 38 и 32 для случайного состояния, в данном случае использовано два значения, чтобы на первом шаге разбить одну большую выборку на две части и далее меньшую подвыборку разделить на две части в соотношении, которое было описано выше. В настоящей работе было выбрано классическое разбиение массива данных при формировании трёх выборок, что часто встречается в документации к готовым программным реализациям машинного обучения.

Следующим этапом подготовки данных явилась генерация признаков, идеи которой на основе подсчёта значений признаков подробно описаны во многих статьях [9]. Для нашей реализации было принято решение использовать несколько признаков. Первый - признак длины, подсчитываемый, как число всех элементов в доменном имени. Другим признаком было выбрано значение классической энтропии:

$$H_x = -\sum_{i=1}^{n} p_i * log_2 p_i \,, \qquad (6)$$

Такой подход позволяет сгенерировать энтропийные признаки, которые показывают частоту появления каждого символа, в доменном имени, знак минус перед оператором суммы, позволяет избавиться от отрицательного значения энтропии, так как при подсчёте вероятности из-под логарифма никогда не выйдет положительное значение.

Следующим является также энтропийный признак, зависящий не от вероятности встречаемости символов в доменном имени, а рассчитываемый на основе словаря вероятностей, получаемых при анализе всего известного набора данных для обучения модели. Вид формулы (6) не изменяется просто при обучении мы заранее вычисляем общее число символов во всех доменных именах и далее делим на него каждое значение массива, в который входит количество символов каждого типа. Полученная вероятность и подставляется в формулу (6). При отсутствии символа в доменном имени на тестовых данных сначала предполагалось использовать небольшое отрицательное значение, но потом оно было заменено на использование нулевого значения. Это не меняло значения формулы, так как мы не подставляем нулевое значение под логарифм, а делаем его как выход из формулы (6).

Так же были сгенерированы признаки, основанные на формулах подсчёта, в которые входили, как hash признаки, основанные на числе корзин, так и признаки, основанные на TF-IDF методах, которые применялись в ранее используемых моделях и показывающие неплохие результаты для моделей, обученных на них [9].

После генерации признаков, следующей задачей было создание векторного представления для данных, которые будут использоваться для обучения, тестирования и валидации. Ранее было описано использование слоя Embedding, который принимает вектор определённого размера и преобразует его в вектор иной размерности задаваемой, как гиперпараметр архитектуры. Перед передачей данных было подсчитано, что при разделении данных в доменах используется примерно 201 символ с интервалом в 2 символа в большую или меньшую сторону. Поэтому слой Embedding будет сконструирован для приёма такого формата данных.

Домены были преобразованы в вид векторов размерности 30 символов. Это число было найдено при проведении статистического анализа данных по длине, где размер в 30 символов, соответствовал квантилю 0.99.

При преобразовании были рассмотрены методы Hash корзин, позволяющие задавать hash функцию, которая принимала буквенное значение и преобразовывала его в численное значение. Другие методы, такие, как порядковое кодирование не использовались в этой работе, так как делали бы избыточным число признаков и делали вход модели глубокого обучения слишком широким.

IV. ПРЕДЛАГАЕМАЯ АРХИТЕКТУРА

После генерации признаков на сформированных массивах данных были обучены ранее известные архитектуры, описанные выше.

Для предлагаемой архитектуры (рис. 3) были реализованы классы пакетной передачи данных и классы callback для оценки параметров и метрик обучения в режиме реального времени.

Полученная архитектура принимает на вход данные, которые будут отправлены на три входных слоя:

1. Слой Embedding (201, 251),
2. Слой Conv1D (128),
3. Слой LSTM (128).

Первый слой Embedding принимает данные, преобразованные из символов доменного имени в численные векторы с помощью hash функции.

Второй слой Conv принимает набор данных, как и слой Embedding.

Третий слой LSTM принимает доменную часть нулевого уровня, который при отсутствии информации такового принимает массив нулей, имеющих длину, основанную на среднем значении доменов нулевого уровня, как длину в три символа.

Далее после обработки на каждом из описанных выше трёх слоёв данные передаются дальше на второй уровень, соответственно по номерам путей:

1. BiLSTM (GRU(128))
2. Conv1D (64)
3. Ожидает

На втором уровне архитектуры, данные проходят через новые преобразования, чтобы веса в слоях получили более полное понимание о имеющихся зависимостях в данных. Стоит отметить, что слой BiLSTM, имеет внутри себя два уровня слоёв GRU, что было выбрано после изучения статьи [10].

Слой Conv1D работает с 64 фильтрами, что, как мы считаем, должно помочь выявить закономерности полнее, чем в слоях с кратковременной памятью, так как фильтры имеют более сложные конфигурации, чем переход от ячейки к ячейке.

Последний слой, соответствующий LSTM (128) ожидает предобработки данных и градиентного прохода по двум слоям, описанным ранее.

На третьем уровне полученные выходы из каждого слоя вытягиваются и образуют новый массив данных, зависящий так же от первоначального входного пакета доменных имён.

После процесса вытягивания при помощи средств Keras, подобно встроенному слою Flatten, мы решили не использовать сложные слои Keras. Так как дополнительные слои глубокой сети приводит только к ухудшению обобщающих качеств архитектуры, за счет, как недостаточного объёма обучающих данных, так и потерь информации при проходах по слоям LSTM. И с тем, и с другим недостатками можно бороться в будущем, увеличивая объем обучающих выборок и экспериментируя со структурами слоев, однако эти недостатки не сильно влияют на качество решений.

Для устранения потери данных было принято решение не использовать слои, имеющие в основе идею кратковременной памяти. Так же на выходы из слоя BiLSTM после объединения двух выборок применяется алгоритм Attention, позволяющий выделять наиболее важные признаки. В нашем случае – это веса, которые будут использоваться дальше, и которые в простейшем случае являются обычным массивом, содержащим взвешенные значения. В нашем случае он является встроенным слоем в Tensorflow, имеющим автоматический подбор весов при обучении архитектуры [10].

Для последнего этапа были выбраны классические полносвязные слои Dense, идущие последовательно и имеющие соответственно (256, 128, 2) нейронных входов в каждом.

Стоит отметить, что функциями активации мы оставили базовую *relu,* так как использование рекомендуемой в некоторых работах функции *elu,* имеющей обучающийся параметр, не дало существенных изменений качества классификации. Но эта функция, имея обучаемый параметр, при расчёте обратной ошибки потребует дополнительных вычислительных ресурсов для расчёта собственного значения.

В последнем слое используется активация на основе softmax и два нейрона, показывающих к какому классу отнести доменное имя. Такая реализация необходима, так как используемой функцией ошибки, по которой мы и оптимизируем веса, является категориальная кросс энтропия, ожидающая получения на выходе вероятностных представлений. Эта функция для бинарной классификации имеет вид бинарной кросс энтропии или *logloss*. Но ранее мы готовили данные для экспериментов с использованием другой функции оптимизации. Поэтому с учетом выполнения бинарной классификации, мы можем продолжать ориентироваться на ранее выбранную функцию оптимизации и передавать только одно число целевого класса.

Layer (type)	Output Shape	Param #	Connected to
input_22 (InputLayer)	[(None, 30, 1)]	0	[]
input_21 (InputLayer)	[(None, 30)]	0	[]
conv1d_12 (Conv1D)	(None, 29, 128)	384	['input_22[0][0]']
embedding_14 (Embedding)	(None, 30, 251)	50451	['input_21[0][0]']
input_23 (InputLayer)	[(None, 4, 1)]	0	[]
conv1d_13 (Conv1D)	(None, 28, 128)	32896	['conv1d_12[0][0]']
bidirectional_2 (Bidirectional)	(None, 256)	292608	['embedding_14[0][0]']
lstm_8 (LSTM)	(None, 128)	66560	['input_23[0][0]']
flatten_7 (Flatten)	(None, 3584)	0	['conv1d_13[0][0]']
flatten_8 (Flatten)	(None, 256)	0	['bidirectional_2[0][0]
flatten_9 (Flatten)	(None, 128)	0	['lstm_8[0][0]']
concatenate_1 (Concatenate)	(None, 3968)	0	['flatten_7[0][0]', 'flatten_8[0][0]', 'flatten_9[0][0]']
dropout_2 (Dropout)	(None, 3968)	0	['concatenate_1[0][0]']
dense_11 (Dense)	(None, 256)	1016064	['dropout_2[0][0]']
dense_12 (Dense)	(None, 128)	32896	['dense_11[0][0]']
dense_13 (Dense)	(None, 2)	258	['dense_12[0][0]']

Total params: 1,492,117
Trainable params: 1,492,117
Non-trainable params: 0

Рис.3. Предложенная архитектура в виде результирующей таблицы Keras.

Далее перейдём к рассмотрению архитектур по полученным метрикам.

V. РЕЗУЛЬТАТЫ СРАВНЕНИЯ МОДЕЛЕЙ

После создания собственной архитектуры и компиляции ранее известных и используемых архитектур для классификации DGA доменов, модели прошли через цикл тренировок,

включающий 20 эпох и после тренировки были протестированы на ранее размеченных данных.

На тренировочных данных предложенная в данной статье модель показала наилучшие результаты по качеству классификации на целевой метрике, но она имеет просадки по скорости работы на больших объёмах данных, что объясняется большим числом матричных операций, которые образуются при создании глубоких сетей.

Полученные результаты экспериментов с классическим машинным обучением можно увидеть в таблице 1.

Таблица 1. Результаты экспериментов

Метрики / Архитектуры	$f_{0.4}$	f1	accuracy	precision
Предл. арх	0.908	0.874	0.83	0.923
LSTM	0.827	0.828	0.83	0.827
Conv	0.881	0.857	0.827	0.891
Посл.арх	0.87	0.865	0.86	0.894

Видно, что предлагаемая архитектура превосходит остальные ранее используемые архитектуры по целевой метрике, но данные результаты получены на тестовых данных. Рассмотрим результаты работы на основе вариационных данных (Табл. 2).

Таблица 2. Результаты экспериментов

Метрики / Архитект.	$f_{0.4}$	f1	accuracy	precision
Пред.арх	0.893	0.891	0.889	0.894
LSTM	0.76	0.74	0.714	0.768
Conv	0.755	0.741	0.723	0.761
Посл.арх	0.809	0.779	0.742	0.821

На валидационных данных значения метрик для всех архитектур ухудшились. Это связано с тем, что в валидационные данные были добавлены алгоритмы генерации доменов, не используемые для тренировки, с целью имитации ситуации получения данных с реального промышленного блока.

Предложенную архитектуру глубокой нейронной сети можно сравнить и с архитектурами, используемыми ранее для решения задачи бинарной классификации доменов. Но полагаясь на технический прогресс это не целесообразно, так как они уступают в своей обобщающей способности глубоким архитектурам.

VI. ДИСКУССИЯ ПО ТЕМЕ ИССЛЕДОВАНИЙ

В настоящее время де-факто стандартом для классификации DGA доменов являются модели на основе глубокого обучения, которые используются для защиты контуров, отвечающих за безопасность данных компании. Для повышения эффективности работы подобных алгоритмов ранее использовалась идеи создания наиболее глубоких сетей [10]. Однако, такой подход не выдержал конкуренции с более простыми, но имеющими лучшую обобщающую способность алгоритмами, использующими распределённые параллельные сети для каждой из частей доменного имени.

Большая часть работ, в которых рассматриваются архитектуры глубокого обучения для классификации DGA доменов имеют в своей основе идею использования численных преобразований признаков. И, последующей передачи полученного векторного представления в слой Embedding для дальнейшего преобразования его в новый более сжатый в классическом представлении формат векторных данных [8].

Но такой подход не улавливает более глубоких скрытых зависимостей, которые можно получить, используя параллельно несколько различных слоёв, предназначенных для обобщения зависимостей после преобразования доменного имени в численный вектор.

Авторы статей, основанных на классических подходах машинного обучения, выделяют скорость работы их реализаций, что верно, так как при матричном умножении есть прямая зависимость между количеством параметров в каждом слое нейронной сети и временем обработки [7]. Трудно не согласиться, что скорость работы важна, но в направлении улучшения целевых метрик точности классификации (не только f-beta меры) классические подходы за последние несколько лет не сдвинулись с места. А при учете скорости развития мощности вычислительных кластеров, вопрос преимущества в скорости становится не таким очевидным.

Применение иных генеративных подходов для решения рассматриваемой задачи, так же возможно, но они не учитывают, того, что по скорости работы они будут уступать классическим алгоритмам, а по точности - даже не приблизятся к архитектурам, основанным на глубоком обучении [9].

Значительный интерес сегодня представляют исследования по работе с глубоким обучением и возможностью использования более скоростных алгоритмов для решения узко направленных задач, таких как машинное зрение, или для решения более общих задач, таких как классификация легитимных и нелегитимных доменов.

VII. ЗАКЛЮЧЕНИЕ

В настоящей работе рассматривались подходы для решения задач выявления доменов, сгенерированных с помощью DGA алгоритмов, основанные на архитектурах глубокого обучения, пришедших на смену более простым алгоритмам классического машинного обучения.

Для выявления DGA доменов была предложена параллельная архитектура, имеющая в своей основе слои глубокого обучения, позволяющие более точно выделить зависимости из различных частей доменного имени.

В работе был проведён первичный статистический анализ для получения оптимального значения длины домена, после преобразования в численный вектор.

Так же были рассмотрены новые наборы признаков, которые можно выделить из доменного имени на основе энтропии и численных преобразований.

Для проведения обучения, тестирования и валидации был собран и сгенерирован массив данных на основе существующих DGA алгоритмов в открытом доступе и легитимных имен доменов из базы знаний Alexa.

Для создания архитектуры машинного обучения были выбраны такие средства, как Tensorflow и Keras, которые подробно задокументированы, и с помощью которых созданы многочисленные проекты, использующие архитектуры глубокого обучения.

Во время создания собственной глубокой архитектуры были подобраны слои и их гиперпараметры для получения наилучших результатов на основе целевой метрики «$f_{0.4}$ мера».

После анализа результатов экспериментов, можно утверждать, что при классификации доменов с использованием ранее известных архитектур можно добиться точности более 0.8 на реальных данных. Но при появлении новых уникальных DGA алгоритмов модель приходится переобучать, так как по отдельности слои LSTM или Conv не способны выявить основные зависимости в генерации нелегитимных данных.

Подводя итог, можно сказать, что для решения задачи классификации DGA доменов предложенная архитектура глубокой нейронной сети подходит наилучшим образом по критерию «оценка качества» при ориентации на разные метрики качества. Предложенная модель превзошла своих конкурентов за счет более сложной структуры, учитывающей разные части доменного имени и генерации более сложных признаков.

БЛАГОДАРНОСТИ

Авторы выражают благодарность Высшей инжиниринговой школе НИЯУ МИФИ за помощь в возможности опубликовать результаты выполненной работы.

БИБЛИОГРАФИЯ

[1] Sengupta, T.K., Lestandi, L., Haider, S.I. et al. Correction to: Reduced order model of flows by time-scaling interpolation of DNS data. Adv. Model. and Simul. in Eng. Sci. 5, 27 (2018). https://doi.org/10.1186/s40323-018-0120-9

[2] Yang, L, Liu G, Zhai J, Dai Y, Yan Z, Zou Y, Huang W (2018) A novel detection method for word-based dga In: International Conference on Cloud Computing and Security, 472–483.. Springer, Haikou.

[3] Wang, W, Shirley K (2015) Breaking bad: Detecting malicious domains using word segmentation. arXiv e-prints:arXiv:1506.04111. https://ui.adsabs.harvard.edu/abs/2015arXiv150604111W.

[4] Wang, Q., Feng, C., Xu, Y. et al. A novel privacy-preserving speech recognition framework using bidirectional LSTM. J Cloud Comp 9, 36 (2020). https://doi.org/10.1186/s13677-020-00186-7.

[5] ArunKumar KE, Kalaga DV, Kumar ChMS, Kawaji M, Brenza TM. Forecasting of COVID-19 using deep layer recurrent neural networks (RNNs) with gated recurrent units (GRUs) and long short-term memory (LSTM) cells. Chaos Solitons Fractals. 2021;146:110861.

[6] Leevy, J.L., Khoshgoftaar, T.M. & Villanustre, F. Survey on RNN and CRF models for de-identification of medical free text. J Big Data 7, 73 (2020). https://doi.org/10.1186/s40537-020-00351-4.

[7] Touzani, Y., Douzi, K. An LSTM and GRU based trading strategy adapted to the Moroccan market. J Big Data 8, 126 (2021). https://doi.org/10.1186/s40537-021-00512-z

[8] Burgess, J, Carlin D, O'Kane P, Sezer S (2020) REdiREKT: Extracting Malicious Redirections from Exploit Kit Traffic In: 2020 IEEE Conference on Communications and Network Security (CNS).. IEEE.

[9] Duncan, B (2020) Malware Traffic Analysis. https://www.malware-traffic-analysis.net/. Accessed 7 May 2021.

[10] Lu, Q., Sun, S., Duan, H. et al. Analysis and forecasting of crude oil price based on the variable selection-LSTM integrated model. Energy Inform 4, 47 (2021). https://doi.org/10.1186/s42162-021-00166-4

Applying Deep Learning to Identify and Classify DGA Domains

E. Diuldin, K.S. Zaytsev

Abstract - The purpose of this work is to study methods for detecting malicious domains generated using DGA algorithms. To solve this problem, proposed to create a deep learning architecture based on the Tensorflow framework, and implement layers based on the Keras library. For training, testing and validation of the resulting architecture, data generated based on 25 known DGA generation algorithms and legitimate data obtained from Top Alexa were used. Using the data obtained, the proposed neural network architecture was compared with the known implementations of machine learning architectures according to the classification of DGA domains. The target metric by which the quality of the classification was compared was the f-measure with the parameter - the weight of accuracy in the metric (β) equal to 0.4, which made it possible to choose the model with the highest prediction accuracy. The results obtained confirmed the effectiveness of the proposed solution. The result of the work was the creation of an effective machine learning architecture used to classify malicious DGA domains.

Keywords **– domain generation, DGA, classification, machine learning, deep learning, Keras**

REFERENCES

[1] Sengupta, T.K., Lestandi, L., Haider, S.I. et al. Correction to: Reduced order model of flows by time-scaling interpolation of DNS data. Adv. Model. and Simul. in Eng. Sci. 5, 27 (2018). https://doi.org/10.1186/s40323-018-0120-9

[2] Yang, L, Liu G, Zhai J, Dai Y, Yan Z, Zou Y, Huang W (2018) A novel detection method for word-based dga In: International Conference on Cloud Computing and Security, 472–483.. Springer, Haikou.

[3] Wang, W, Shirley K (2015) Breaking bad: Detecting malicious domains using word segmentation. arXiv e-prints:arXiv:1506.04111. https://ui.adsabs.harvard.edu/abs/2015arXiv150604111W.

[4] Wang, Q., Feng, C., Xu, Y. et al. A novel privacy-preserving speech recognition framework using bidirectional LSTM. J Cloud Comp 9, 36 (2020). https://doi.org/10.1186/s13677-020-00186-7.

[5] ArunKumar KE, Kalaga DV, Kumar ChMS, Kawaji M, Brenza TM. Forecasting of COVID-19 using deep layer recurrent neural networks (RNNs) with gated recurrent units (GRUs) and long short-term memory (LSTM) cells. Chaos Solitons Fractals. 2021;146:110861.

[6] Leevy, J.L., Khoshgoftaar, T.M. & Villanustre, F. Survey on RNN and CRF models for de-identification of medical free text. J Big Data 7, 73 (2020). https://doi.org/10.1186/s40537-020-00351-4.

[7] Touzani, Y., Douzi, K. An LSTM and GRU based trading strategy adapted to the Moroccan market. J Big Data 8, 126 (2021). https://doi.org/10.1186/s40537-021-00512-z

[8] Burgess, J, Carlin D, O'Kane P, Sezer S (2020) REdiREKT: Extracting Malicious Redirections from Exploit Kit Traffic In: 2020 IEEE Conference on Communications and Network Security (CNS).. IEEE.

[9] Duncan, B (2020) Malware Traffic Analysis. https://www.malware-traffic-analysis.net/. Accessed 7 May 2021.

[10] Lu, Q., Sun, S., Duan, H. et al. Analysis and forecasting of crude oil price based on the variable selection-LSTM integrated model. Energy Inform 4, 47 (2021). https://doi.org/10.1186/s42162-021-00166-4