СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1 АЛГОРИТМЫ ГЕНЕРАЦИИ ДОМЕННЫХ ИМЕН 6

1.1 Общие принципы работы 6

1.2 Рассмотренные алгоритмы 7

2 СУЩЕСТВУЮЩИЕ ПОДХОДЫ К КЛАССИФИКАЦИИ 13

3 НЕЙРОННЫЕ СЕТИ 14

3.1 Long short-term memory 17

4 ЭКСПЕРИМЕНТ 20

4.1 Существующие подходы 20

4.2 Long short-term memory 26

4.3 Сравнительный анализ 30

ЗАКЛЮЧЕНИЕ 31

БИБЛИОГРАФИЧЕСКИЙ СПИСОК 32

ВВЕДЕНИЕ

В настоящее время наибольшую угрозу информационной безопасности несут вредоносные программы. Некоторые разновидности вредоносных программ используют алгоритмы генерирования доменных имен для определения адресов управляющих серверов. Подобные алгоритмы позволяют защитить управляющие вредоносные сервера от однократного отключения или добавления адресов в черные списки. Чаще всего данные алгоритмы используются в крупных ботнетах.

Ботнет - некоторая сеть, в том числе компьютерная, состоящая из устройств (ботов), со специально запущенным вредоносным программных обеспечением. Чаще всего боты инфицируются посредством вредоносного программного обеспечения, полученного из сети Интернет. Однако путём инфицирования может служить также локальная сеть или устройства ввода, например флэш накопители. Ботнеты являются наиболее распространенным средством кибер атак. Они управляются его создателем при помощи специальных управляющих командных серверов (Command and Control Servers). Большинство из них используются для монетизации различными способами, такими как: распределенные атаки отказ в обслуживании (DDoS атаки), продажа Drive By Download, атаки на клиентов дистанционного банковского обслуживания, для спама и проведения фишинговых атак. Эти и другие угрозы ботсетей выделены в статье [10].

Для удержания контроля над ботами и их управления ботнеты используют множество способов. Это могут быть одноранговые сети, почтовые протоколы, социальные сети или анонимные сети, такие как TOR или i2p. Однако самым распространенным методом на данный момент являются алгоритмы генерации доменных имен (Domain Generation Algotithms).

Они позволяют удерживать контроль над управляющими серверами. В основном подобные алгоритмы используются в крупных ботсетях. Например, одним из первых случаев был компьютерный червь Conficker в 2008 году. На

сегодняшний день подобных вредоносных программ насчитываются десятки, каждая из которых представляет серьезную угрозу. Помимо этого, алгоритмы совершенствуются, их обнаружение становится сложнее. Например, осенью 2014 года была обнаружена новая версия ботнета Matsnu, в которой для генерации доменов используются существительные и глаголы из встроенного списка. Подробнее историю развития алгоритмов генерации доменных имен рассмотрена в статье [6].

Целью данной работы является разработка модели на основе методов машинного обучения для распознавания и классификации вредоносных доменных имен, полученных при помощи анализа алгоритмов генерации доменных имен, а также получение количественной оценки качества классификации на сформированной выборке доменных имен.

Для достижения поставленной цели были поставлены три задачи. Во-первых, подготовка эксперементальных данных. При этом необходимо было составить выборки не только для легетимных и вредоносных имен, но и выборки по отдельным классам вредоносных программ. Под вредоносным доменном именем понимается доменное имя, искуственно полученное в результате работы алгоритма генерации доменных имен. Во-вторых, поиск и реализация существующих подходов, их сравнительный анализ на подготовленных данных. В-третьих, разработка и апробация собственной модели на основе реккурентных нейронных сетей.

Новизна данной работы заключается в решении третьей задачи. Так, для классификации предлагается использование модели, построенной на основе рекуррентной нейронной сети, что является новым направлением исследования для решения проблемы обнаружения и распознования вредоносных доменных имен. Данный подход позволяет улучшить качество классификации по сравнению с существующими подходами решения данного класса задач. Поэтому рассмотрение нейросетей для предотвращения современных угроз, связанных с ботнетами и алгоритмами генерации доменных имен является актуальной проблемой информационной безопасности не только в настоящее время, но и в будущем.

В главе 1 рассматриваются принципы работы алгоритмов генерации доменных имен. Приводится простейший пример реализации, а также особенности каждого из рассмотренных таких алгоритмов.

В главе 2 производится обзор существующей литературы, затрагивающей данную проблематику. Выделяются наиболее эффективные модели и подходы к решению задачи распознавания вредоносных доменных имен.

В главе **??** рассмотрены теоретические основы и архитектура разработанной модели. Затрагиваются вопросы, связанные с нейронными сетями, реккурентными нейронными сетями, моделью Long short-term memory.

Глава 4 посвящена практической реализации существующих подходов 4.1 и разработанной модели 4.2, так в разделе 4.1 обсуждаются вопросы выбора параметров для классификаторов, в разделе 4.3 представлен сравнительный анализ реализованных подходов.

1 АЛГОРИТМЫ ГЕНЕРАЦИИ ДОМЕННЫХ ИМЕН

Алгоритмы Генерации Доменных Имен (DGA) представляют собой алгоритмы, используемые вредоносным программным обеспечением (malware) для генерации большого количества псевдослучайных доменных имен, которые позволят им установить соединение с управляющим командным центром. Тем самым они обеспечивают мощный слой защиты инфраструктуры для вредоносных программ. С первого взгляда концепция создания большого количества доменных имен для установки связи кажется не сложной, но методы, используемые для создания произвольных строк часто скрываются за разными слоями обфускации. Это делается для усложнения процесса обратной разработки и получения модели функционирования того или иного семейства алгоритмов. В разделе 1.1 рассмотрены общие принципы алгоритмов генерации доменных имен, а раздел 1.2 раскрывает особенности реализаций некоторых из рассмотренных алгоритмов.

1.1 Общие принципы работы

Общий принцип работы представлен на рис 1.1. В общем случае вредоносному файлу необходим какой-либо параметр для инициализации генератора псевдослучайных чисел (ГПСЧ). В качестве этого параметра может выступать любой параметр, который будет известен вредоносному файлу и владельцу ботнета. В нашем случае - это значение текущей даты и времени. Вредоносный файл, используя протокол HTTP посылает запрос на сайт cnn.com. В ответ на этот запрос cnn.com возвращает в заголовках HTTP ответа текущие время и дату в формате GMT. Владелец ботнета таким же способом получает текущее время и дату в формате GMT. Далее, это значение, попадает в сам алгоритма генерации доменных имен, инициализируя ГПСЧ, который может иметь вид например Линейного конгруэнтного генератора или Регистра сдвига с обратной связью. Поэтому, используя одинаковые вектора инициализации, вредоносный файл и владелец ботнета получают идентичные таблицы доменных имен. После

этого владельцу ботнета достаточно зарегистрировать лишь один домен, для того, чтобы вредоносный файл, рекурсивно посылая запросы к DNS серверу получил IP адрес управляющего сервера для дальнейшей установки с ним соединения и получения, выполнения команд.

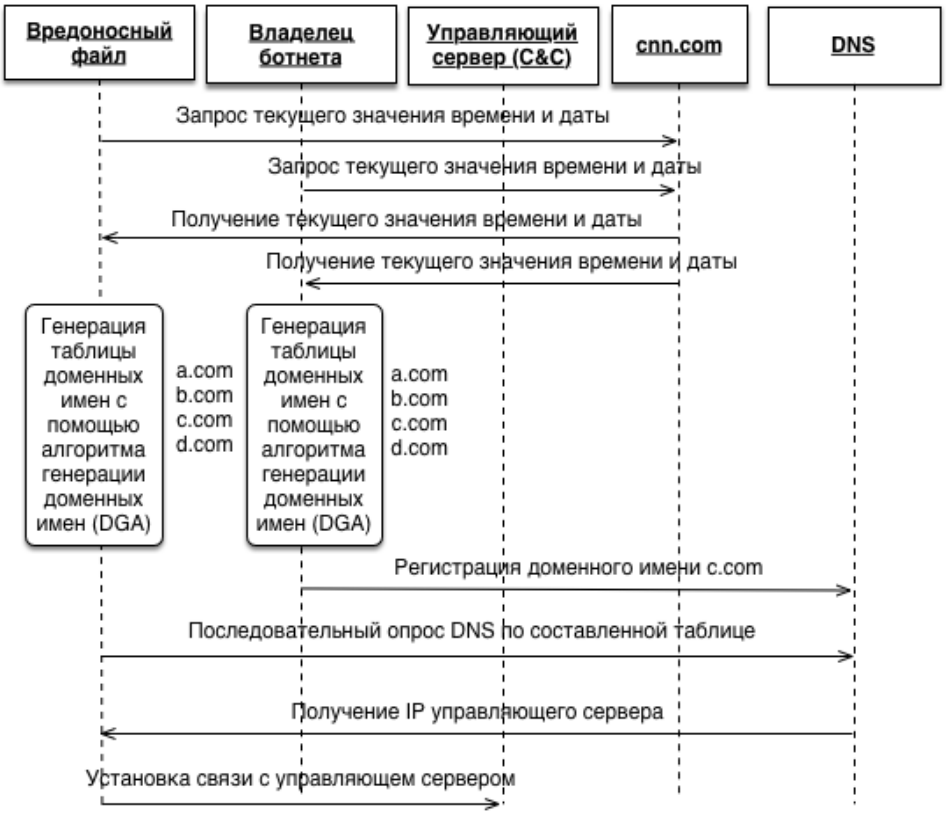

Рис. 1.1 – Общий принцип работы

1.2 Рассмотренные алгоритмы

В ходе выполнения данной работы были проанализированы 8 разновидностей алгоритмов генерации доменных имен, а именно: Conficker, Cryptolocker, Ramdo, PushDo, Zeus, Tinba, Rovnix, Matsnu. Для каждого из них, путём обратной разработки были составлены модели работы алгоритмов генерации доменных имен и реализованы на языках программирования высокого уровня. Далее в работе представлены описание и особенности работы каждого из этих алгоритмов.

Conficker - вирус, впервые появившийся в 2008 году, использующий для заражения машин популярную уязвимость MS08-067. Одним из первых применил технику DGA. Процесс генерации доменного имени можно описать пятью шагами, как показано на рис 1.2 и его архитектура идентична принципу, описанному в части 1.1.

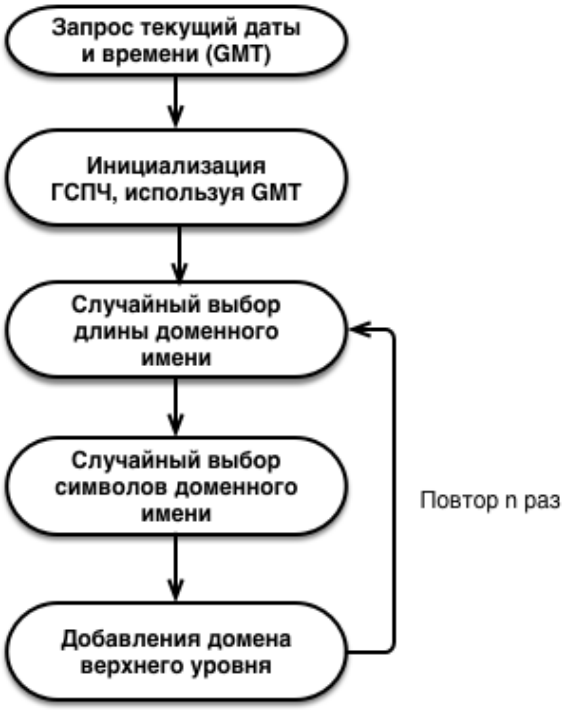

Рис. 1.2 – Принцип работы conficker DGA

Cryptolocker - имя вредоносных программ вида троян-вымогатель (ransomware). Целью данного вируса являются системы Microsoft Windows. Cryptolocker полностью шифрует содержимое файловой системы жертвы и требует заплатить выкуп для получения ключа дешифрования. DGA, который использует cryptolocker относительно прост, однако использует множество приемов, которые осложняют процесс его обратной разработки. Его алгоритм использует для инициализации 4 значения - ключ, день, месяц, год. Ключ может быть константой или рассчитываться по формуле

$$key = ((key * 0x10624DD3) >> 6) * 0xFFFFFC18) + key \qquad (1.1)$$

В данной работе за значение ключа взята константа 0x41. Каждый символ рассчитывается по формуле

$$chr(ord(a) + (year\ xor\ month\ xor\ date)\ \%\ 25) \tag{1.2}$$

А длина составляет доменного имени составляет

$$date >> 3\ xor\ year >> 8\ xor\ year >> 11\ and\ 3 + 12 \tag{1.3}$$

В финальной стадии генерации доменного имени добавляется домен верхнего уровня, который последовательно выбирается из массива зашитых значений.

PushDo - второй по величине ботнет, впервые появившийся в 2007 году. Для установки связи с командным сервером использует два механизма. Первый - зашитые доменные имена, и второй - DGA. Второй механизм имеет четыре составляющие, а именно - системное время, вектор инициализации, функцию хеширования MD5 и сам генератор доменных имен. Каждый день, при генерации первого доменного имени DGA вектору инициализации присваивается значение системного времени, которое получается путем вызова WINAPI функции GetLocalTime, операций битового сдвига и XOR - функция шифрования. После этого вектор инициализации попадает на вход функции хеширования MD5, вывод которой в свою очередь подает в генератор доменного имени и вектор инициализации генерации следующего доменного имени. Однако вектор инициализации следующего доменного имени берет не весь результат функции MD5, а лишь первые её 4 байта. Схема работы этого алгоритма генерации доменных имен представлена на рис 1.3.

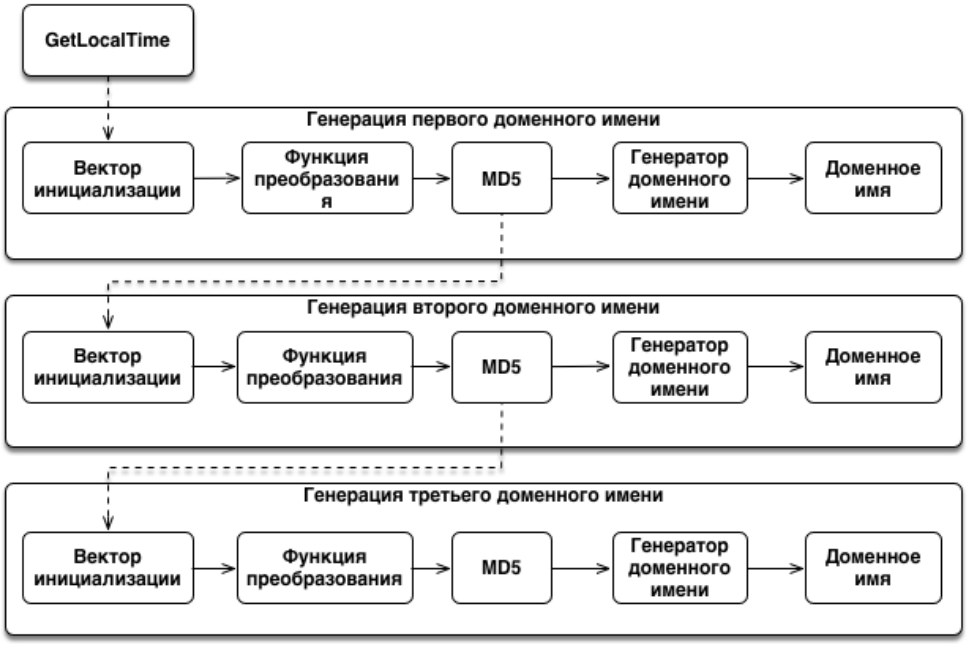

Рис. 1.3 – Принцип работы PushDo DGA

Отдельно стоит рассмотреть генератор доменного имени, получающий на вход результат функции MD5. Каждый экземпляр такого генератора имеет четыре уникальных строки. Это позволяет обновлять генератор с каждой новой версий ботнета или иметь множество непересекающихся ботнетов одного и того же типа. Сначала генератор вычисляет длину доменного имени. Она рассчитывается делением первых четырех байт на 4 и лежит в пределах от 9 до 12. Далее специальный цикл, используя остаток от деления переводит вывод функции MD5 в читаемое доменное имя и добавляет к нему домен верхнего уровня. В итоге, всего, каждый день PushDo DGA генерирует 30 доменных домен, зависящих не только от системного времени, но и от зашитых значений - специальных строк генератора.

Zeus - семейство вредоносных программ, ворующих аутентифицирующие данные, которое впервые появилось в 2007 году. Первые два варианта этого ботнета базировались на зашитых централизованных адресах управляющих серверов. Эти сервера постоянно отслеживались и отключались при помощи антивирусных компаний и центров реагирования на инциденты информационной безопасности. Именно поэтому, в 2011 году было обнаружено появление

новой версии этого ботнета, который использует одноранговые сети и DGA для защиты своей инфраструктуры. В первую очередь защита построена на одноранговых сетях, однако если все зараженные машины не отвечают, то используется DGA. Схема его реализация схожа с алгоритмом, используемом в ботнете PushDo. Алгоритм также использует системное время, алгоритм хеширования MD5 и генератор доменного имени. Однако в отличии от PushDo системное время является вектором инициализации для всех 1000 доменных имен генерируемых каждую неделю, а номер доменного имени является параметром salt (соль) в функции хеширования MD5. Генератор доменного имени можно представить в виде следующего псевдокода.

```
hash = MD5(time+salt)
name = ””
for (j = 0; j < len(hash); j++) {
    c1 = (hash[j] & 0x1F) + ’’a;
    c2 = (hash[j] / 8) + ’’a;
    if(c1 != c2 && c1 <= ’’z) name += c1;
    if(c1 != c2 && c2 <= ’’z) name += c2;
}
```

Семейство Ramdo, впервые обнаруженное в декабре 2013 года кардинально отличается от других ботнетов. Ramdo используют множество приемов антиотладки, а также использует технологию double flux для установки соединения с управляющими серверами. Несмотря на сложность обратной разработки данный ботнет имеет достаточно простой и небольшой DGA, который основывается на зашитой константе и итераторе, основанного на регистре сдвига и булевой операции исключающее или. Для формализации модели DGA данного семейства в работе использован исходный код из работы [15]. Кроме этого идентичную модель DGA использует и Tinba, более известный как Tiny Banker. Основное отличие лишь в том, что для генерации следующего доменного имени Tinba использует также зашитый первоначальный адрес управляющего сервера.

Бурное развитие методов идентификации и фильтрации вредоносных доменных имен привело к развитию новых видов DGA. Такими примерами

являются алгоритмы генерации доменных имен ботнетов Rovnix и Matsnu. Их алгоритмы проектировались специально для обхода систем идентификации, построенных на количественной оценки энтропии доменного имени или лингвистических моделях, например n-gramm. Rovnix первым из ботнетов предложил использовать специально заготовленный текст для генерации доменного имени. В качестве такого текста Rovnix использует Декларацию о независимости США. На основе системного времени Rovnix выбирает слова из данного текста и конкатенирует их, для достижения определенной длины. Ботнет Matsnu имеет схожий подход, вредоносная программа использует два зашитых словаря, один из которых содержит существительные, а второй глаголы. Тем самым, основываясь на системном времени алгоритм генерации доменного имени выбирает существительное, а затем выбирает глаголы, для достижения необходимой длины доменного имени. Стоит отметить, что подходы этих двух ботнетов действительно с легкостью обходят системы идентификации, основанные лишь на энтропийной оценке, поэтому для идентификации этих классов вредоносных доменных имен требуются более сложные модели, описанные в главах 4.1 и 4.2.

2 СУЩЕСТВУЮЩИЕ ПОДХОДЫ К КЛАССИФИКАЦИИ

В данной главе рассматриваются теоретические основы существующих подходов классификации вредоносных доменных имен. Их реализация и результаты их работы представлены в разделе 4.1.

В настоящее время существует множество работ, связанных анализом алгоритмов генерации доменных имен. Проблемы автоматического анализа алгоритмов DGA и пути их решения можно найти в статье [1]. Идея использования методов машинного обучения освещена в работе [7]. Так, ряд известных компаний, занимающихся информационной безопасностью (Damballa, OpenDns, Click Security и др.), применяют подобные решения для анализа и фильтрации сетевой активности вредоносных программ. Например, Click Security в своей работе [3] предлагают использовать решающие деревья для бинарной классификации на принадлежность доменов к вредоносным. Для этого ими предложен способ выделения признаков из домена. Стоит отметить работу [4], которая рассматривает возможность классификации, используя метод опорных векторов (Support Vector Machine) и выделения из доменов признаков n-gram - подстрока, состоящая из последовательных n символов исходной строки. Схожий подход описывается и в работе [5]. Однако, в отличие от работы [4], имеет большую практическую направленность и предлагает использование алгоритма C4.5 (алгоритм для построения деревьев решений). Подход, основанный на анализе морфем в статье [8], является неактуальным, так как последние исследования [6] показывают, что алгоритмы генерации доменных имен совершенствуются с целью обхода существующих способов обнаружения. В результате анализа существующих работ выделены 5 наиболее перспективных существующих подходов к классификации, а именно Naive Bayes, Logistic Regression, Random Forest, Extra Tree Forest, Voting Classification.

3 НЕЙРОННЫЕ СЕТИ

Искусственные нейронные сети первоначально были разработаны как математические модели, отображающие деятельность мозга. Базовая структура нейронной сети представляет собой сеть узлов, соединенных взвешенными связями. Узлы представляют собой нейроны, а взвешенные связи - силу этой связи. Нейронная сеть активируются путем подачи информации в нейроны, и эта активация распространяется по всей сети вдоль взвешенных связей. Впервые такую сеть описали У. Маккалок и У. Питтс в 1943 году. В настоящее время существует множество разновидностей таких сетей с разнообразными свойствами. Например, важное отличие между ними является наличие цикла, так например сети, не имеющие циклов называются сети прямого распространения (feedforward neural networks), а сети, имеющие цикл - рекурсивными или рекуррентными нейронными сетями. Наиболее распространенной сетью прямого распространения является многослойный перцептрон Румельхарта. Данная архитектура нейронной сети имеет входной слой, скрытые слоя и выходной слой. Процесс прохода этих слоев называется прямым (forward pass). Обучая такую модель мы изменяем веса связей, и именно они определяют функцию зависимости выходного вектора от входного. Хорник в 1989 году доказал, что многослойный перцептрон с одним скрытым слоем, содержащий достаточной количество нелинейных связей с определенной точностью способен аппроксимировать любую непрерывную функцию. Исходя из этого, многослойный перцепртрон часто используется для апроксимации какого-либо закона природы.

Рассмотрим многослойный перцептрон с I входными нейронами, которые активируются входным вектором x. Каждый нейрон первого скрытого слоя вычисляет взвешенную сумму входных нейронов. Для скрытого нейрона h мы обозначим это как a_h. Далее a_h подаётся в функцию активации θ_h. Обозначив

веса от нейрона i к нейрону j как ω_{ij} имеем

$$a_h = \sum_{i=1}^{I} w_{ij} x_i \tag{3.1}$$

$$b_h = \theta_h(a_h) \tag{3.2}$$

Таким образом каждый слой такой нейронной сети состоит из сумматора и функции активации. В качестве функции активации могут выступать различные функции, но наиболее часто используемые - сигмоид и гиперболический тангенс.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3.4}$$

Одно из свойств сигмоида позволяет усиливать слабые сигналы, не завися от входных слишком сильных сигналов. В отличии от гиперболического тангенса область значений сигмоида лежит между 0 и 1, в то время как гиперболический тангенс принимает значения от -1 до 1. Важное свойство этих двух функций активации - это их нелинейность. Это позволяет нейронной сети аппроксимировать любые нелинейные функции. Кроме этого, это свойство позволяет строить нейронные сети с несколькими скрытыми слоями, в отличии от нейронных сетей с линейной функцией активацией, где любая такая нейронная сеть с несколькими скрытыми слоями эквивалентна нейронной сети с одним линейным скрытым слоем. Это позволяет создавать более эффективные модели. Другое важное свойство этих функции заключается в их дифференцируемости. Это позволяет обучать нейронную сеть при помощи алгоритма градиентного спуска. А то, что функции могут быть легко выражены через самих себя существенно сокращает вычислительную сложность метода обратного распространения ошибки.

$$\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x)) \tag{3.5}$$

$$\frac{\partial tanh(x)}{\partial x} = 1 - tanh(x)^2 \tag{3.6}$$

После расчета блока в первом скрытом слое, значения передаются в следующий слой, где опять проходят процесс суммирования и подает в функции активации

до тех пор, пока не попадут в выходной слой. Этот вектор y также попадает в нейрон выходного слоя, происходит суммирование

$$a_k = \sum_{h \in H_i} w_{hk} x_h \tag{3.7}$$

Количество выходных нейронов и выходной функции активации зависит от поставленной задачи. Для задач бинарной классификации обычно используется сигмоид. Выходное значение y этой функции можно трактовать как вероятность принадлежности к первому классу, а значение $1-y$ как вероятность принадлежности ко второму классу. Для задачи многоклассовой классификации можно использовать $softmax$ функцию

$$p(C_k|x) = \frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}} \tag{3.8}$$

В результате на выходе мы имеем K выходных значений, которые мы можем интерпретировать как вероятности принадлежности к конкретному классу. Поэтому многослойный перцептррон так часто используется для задач классификации. Поскольку все операции описанные выше дифференцируемы, многослойный перцептрон может быть обучен при помощи метода градиентного спуска. Основная идея этого метода найти производную функцию потерь относительно каждого из весов сети, и изменить их в направлении наиболее быстрого убывания функции потерь, для того, чтобы увеличить вероятность выбора правильного класса при задаче классификации. Пусть z - правильно выбранный класс, тогда функцию потерь можно представить в виде

$$\mathcal{L}(x, z) = (z-1)ln(1-y) - zlny \tag{3.9}$$

Для эффективного расчета градиента в этой работе используется алгоритм обратного распространения ошибки, описанный в работе [17]. Однако нейронные сети без циклов имеют существенный недостаток - выход такой сети зависит лишь от входных векторов в конкретный момент времени и не зависит от предыдущих или будущих входных векторов, а лишь подстраивает веса целевой функции в процессе обучения. Для решения этой проблемы применяются рекуррентные нейронные сети. Рекуррентные нейронные сети главным

образом отличаются наличием цикла. Это позволяет не только сопоставить входному векторы выходной, но и иметь зависимость выходного вектора от всех предыдущих входных векторов.На рис 3.1 изображена развертка такой сети. На каждой итерации в нейронную сеть подается на вход вектор x_t, а на выходе вектор скрытых состояний h_t.

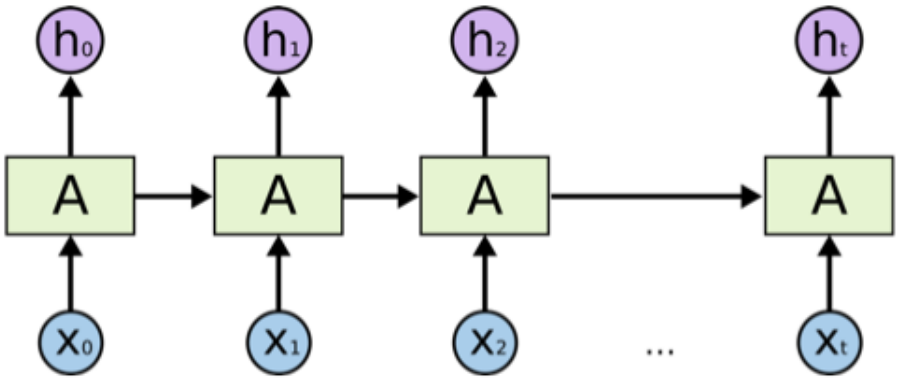

Рис. 3.1 – Рекуррентная нейронная сеть

Обучение такой нейронной сети схоже с обучением многослойного перцептрона и в нашем случае основывается на алгоритме обратного распространения ошибки. Исходя из вышесказанного, мы можем утверждать, что такая архитектура нейронной сети может анализировать информацию, поданную ранее для анализа информации в настоящий момент времени. Однако, на практике оказывается, что если разрыв между прошлой информацией и настоящей достаточно велик, то эта связь теряется и такая сеть неспособна её обрабатывать. Решение этой проблемы было найдено в 1997 году учеными Hochreiter Schmidhuber. В своей работе они предложили новую модель рекуррентной нейронной сети, а именно Long short-therm memory.

3.1 Long short-term memory

Long short-therm memory (LSTM) - модель рекуррентной нейронной сети, способная к обучению долгосрочных зависимостей. В настоящее время данная модель широко используется для всевозможных классов задач, таких

как: распознавание речи, обработка естественных языков и др. LSTM состоит из ряда постоянно связанных подсетей, известных как блоки памяти. Вместо одного слоя нейронной сети, в данной модели используется 4 слоя, взаимодействующих особым образом. Главное в модели LSTM – это ячейка памяти. Модель имеет возможность удалить или добавить информацию в память, и это процесс управляется структурами, которые называют gates. Всего имеется 3 таких структуры: input gate, forget gate, output gate.

На первом шаге LSTM мы решаем какую информацию мы хотим выбросить из ячейки памяти C_t. Для этого вектор h_{t-1} попадает в forget gate

$$f_t = \sigma(W_f[h_{t-1,x_t}] + b_f) \tag{3.10}$$

проходя через сигмоиду на выходе мы получаем значение от 0 до 1. 1 означает “забыть все”, в то время как 0 означает “оставить все”. На следующем шаге мы определяем какую информацию мы хотим добавить в ячейку памяти C_t. Для этого выполняется 2 операции. Во-первых, input gate решает какую информацию мы обновляем 3.11. Во-вторых, слой с функцией активации гиперболический тангенс создает вектор значений $\tilde{C}_t$, который будет добавлен в C_t 3.12.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3.11}$$

$$\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{3.12}$$

Теперь, использую информацию, полученную на предыдущих шагах 3.10 3.11 3.12, мы можем обновить ячейку памяти C_{t-1}, получив C_t. Для этого мы умножим f_t на C_{t-1}, чтобы “забыть” старую информацию и добавим $i_t * \tilde{C}_t$, чтобы обновить C_{t-1}. В итоге получим

$$_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.13}$$

В завершении, вектор h_{t-1} попадает в output gate, проходя через функцию активации сигмоид 3.14, а новый вектор h_t получается путем перемножения выхода output gate со значением $tanh(C_t)$ 3.15.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.14}$$

$$h_t = o_t * tanh(C_t) \tag{3.15}$$

Таким образом, автором описана стандартная структура модели LSTM. Она имеет множество вариаций и дополнений. В данной работе, в разделе 4.2 использована одна из распространенных таких вариаций, а именно Gated Recurrent Unit. Данная модель отличается тем, что forget gate и input gate объединены в один update gate. Данная модель описывается следующими формулами 3.16, 3.17, 3.18, 3.19.

$$z_t = \sigma(W_z[h_{t-1}, x_t]) \tag{3.16}$$

$$r_t = \sigma(W_r[h_{t-1}, x_t]) \tag{3.17}$$

$$\tilde{h_t} = tanh(W[r_t * h_{t-1,x_t}]) \tag{3.18}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h_t} \tag{3.19}$$

4 ЭКСПЕРИМЕНТ

Для тестирования моделей была составлена обучающая выборка. Выборка состоит из 2 классов. Первый - Legit, был взят из списка Alexa Top Million. Второй - DGA, был составлен путем обратной разработки алгоритмов генерации вредоносных доменных имен, взятых из экземпляров вредоносных программ, существующих в сети Интернет. Данный процесс описан в разделе 1.2. В результате был получен список, состоящий из 1000000 Legit доменов и 100000 экземпляров каждого из типов DGA.

4.1 Существующие подходы

Не секрет, что зачастую самым важным при решении задачи является умение правильно отобрать и даже создать признаки. В англоязычной литературе это называется Feature Selection и Feature Engineering. Автор предлагает использовать следующий список параметров: length, entropy, alexa gram, word gram, diff. Первым параметром выступает длина доменного имени. Второй параметр - энтропия рассчитывалась по формуле 4.1.

$$H = \sum_{i=1}^{n} p(i) * log_2\, p(i) \tag{4.1}$$

Далее, была рассмотрена модель $N - gram$. Каждый $n - gram$ (от 3 до 5) был представлен как вектор в n-мерном пространстве, и расстояние между ними было рассчитано с помощью скалярного произведения этих векторов. Более подробно этот метод изложен в работе [11]. Для выделение данных параметров нами была использована библиотека Scikit Learn для языка программирования Python.

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
alexa_vc = CountVectorizer(analyzer='char', ngram_range=(3, 5),
   min_df=1e-4, max_df=1.0)
counts_matrix = alexa_vc.fit_transform(dataframe_dict['alexa']['
   domain'])
```

```
alexa_counts = np.log10(counts_matrix.sum(axis=0).getA1())

dict_vc = CountVectorizer(analyzer='char', ngram_range=(3, 5),
   min_df=1e-5, max_df=1.0)
counts_matrix = dict_vc.fit_transform(word_dataframe['word'])
dict_counts = np.log10(counts_matrix.sum(axis=0).getA1())

all_domains['alexa_grams'] = alexa_counts * alexa_vc.transform(
   all_domains['domain']).T
all_domains['word_grams'] = dict_counts * dict_vc.transform(
   all_domains['domain']).T
```

В результате мы получили 2 параметра : Alexa gram - косинусное расстояние до словаря, состоящего из доменов Alexa Top Million. Word gram - косинусное расстояние до специально составленного словаря, состоящего из наиболее употребительных слов и фраз. Проанализировав существующие параметры, авторами было решено добавить в модель параметр diff:

$$diff = alexa\ gram - word\ gram \tag{4.2}$$

Для анализа каждого из предложенных параметров были построены наглядные графики.

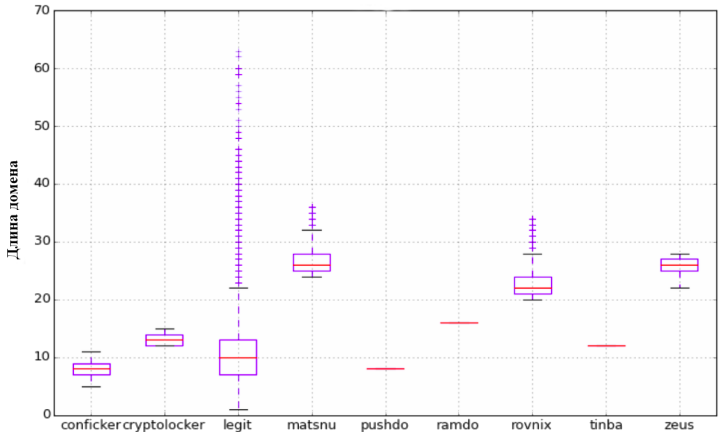

Рис. 4.1 – Длина доменных имен

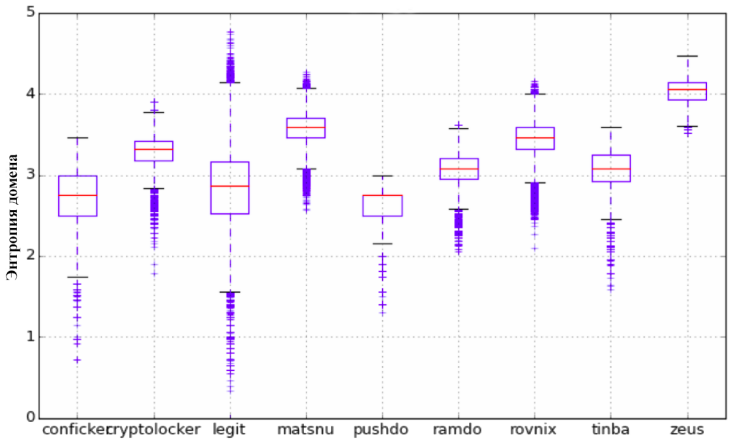

Рис. 4.2 – Энтропия доменных имен

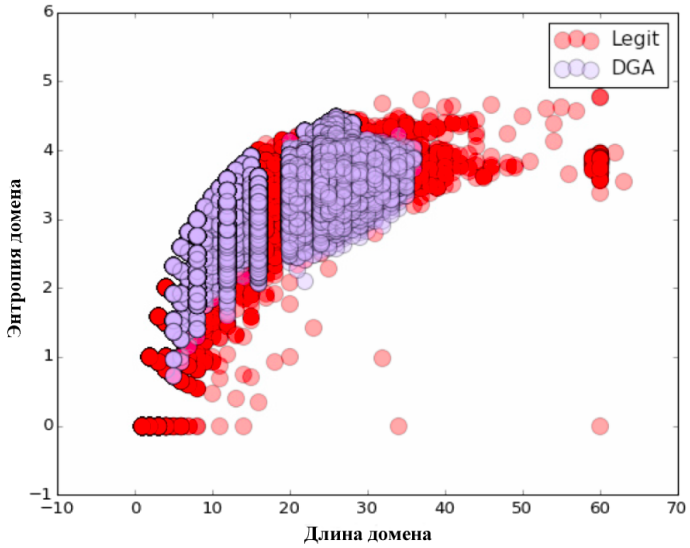

Рис. 4.3 – Зависимость энтропии доменных имен от длины доменных имен

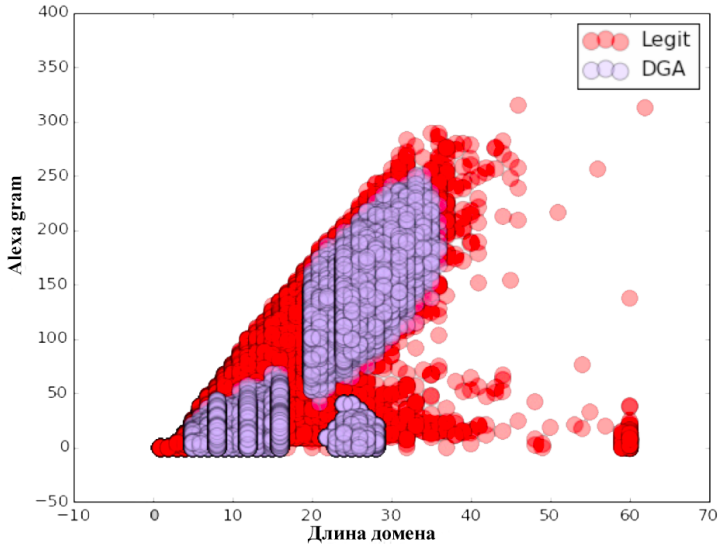

Рис. 4.4 – Зависимость параметра alexa gram от длины доменного имени

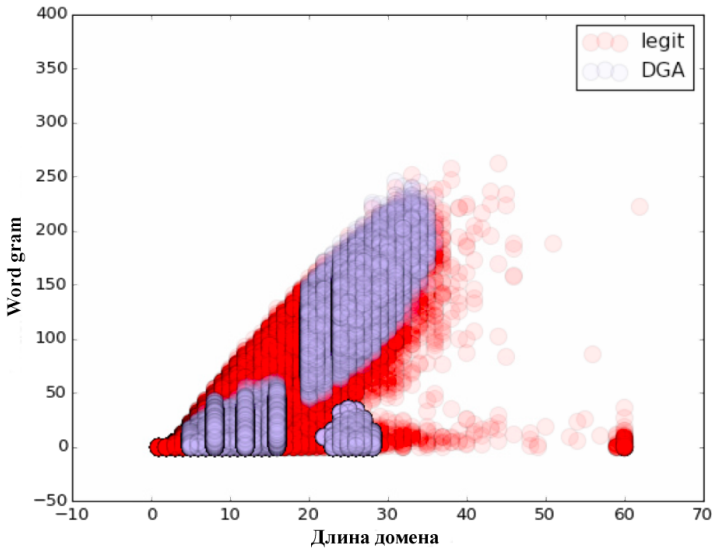

Рис. 4.5 – Зависимость параметра word gram от длины доменных имен

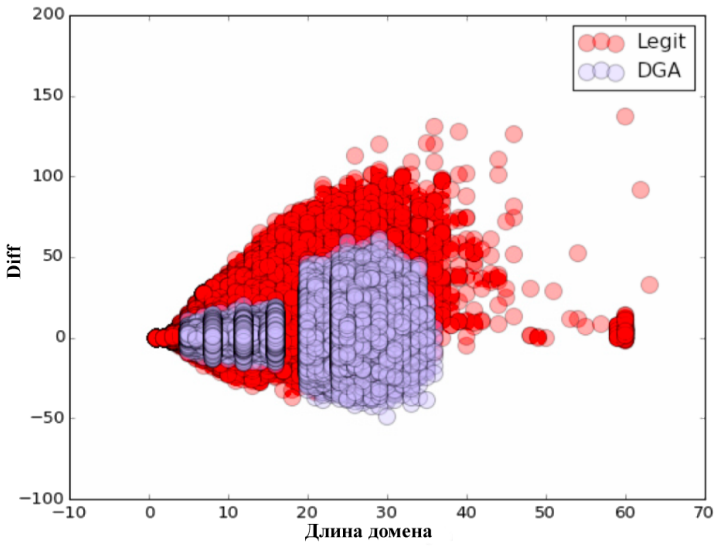

Рис. 4.6 – Зависимость параметра diff от длины доменных имен

Исходя из рис 4.1 можно заметить, что вредоносные доменные имена имеет длину больше 6 симоволов, что оправдывается выосокой стоимостью таких доменных имен и повышением вероятности появления коллизий. Графики 4.2 и 4.3 показывают, что основываясь лишь на энтропии невозможно выделить вредоносные доменные имена в отдельный класс, так как энтропия легитимных доменных имен также лежит в довольно широких пределах. А параметр diff на рис 4.6 был добавлен в модель после анализа рис 4.4 и рис 4.5. Вместе, эти параметры, позволяют выделить отдельные доменные имена в класс вредоносных.

Процесс классификации проводился по принципу 80/20, т.е. обучение моделей проводилось на 80% исходных данных, а тестирование на оставшихся 20%. В качестве алгоритмов классификации использовались такие алгоритмы как:

- Logistic Regression
- Random Forest
- Naive Bayes
- Extra Tree Forest
- Voting Classification

После тестировании и оценки качества классификации были получены следующие результаты.

Таблица 4.1 – Качество классификации

Алгоритм	Бинарная	Многоклассовая
Logistic Regression	86,7%	78,8%
Random Forest	95%	89,3%
Naive Bayes	75,2%	75,6%
Extra Tree Forest	94,6%	88,9%
Voting Classification	94,7%	90%

Автор провёл классификацию не только на вредоносные и легитимные домены, но и по принадлежности к определенному семейству DGA. В многоклассовой классификации наилучший результат показал алгоритм Voting Classification. Данный алгоритм реализован в библиотеке Scikit learn для языка программирования Python и представляет собой классификатор на основе результатов других статистических оценок. Однаков, при распознавании вредоносных доменов лучше всего себя показал алгоритм Random Forest, заключающийся в использовании ансамбля решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга Бреймана, и метод случайных подпространств, предложенный Tin Kam Ho.

Для наглядности количественной оценки модели была построена матрица неточностей (Confusion matrix), которая содержит правильно распознанные хорошие примеры (True Positive), правильно распознанные плохие примеры (True Negative) и ошибки в распознавании (False Positive, False Negative).

Наибольший интерес представляет показатель False Positive, так как именно он, по мнению автора, определяет удобство использования данной модели в системе мониторинга инцидентов информационной безопасности. Изменяя различные параметры модели, в будущем можно добиться снижения этого показателя, не затрагивая качество классификации отнесения доменнных имен к классу вредоносных.

Таблица 4.2 – Матрица неточностей

Имя	Точность
Legit/Legit	95%
Legit/DGA	5%
DGA/Legit	4,3%
DGA/DGA	95,7%

4.2 Long short-term memory

В качестве эксперимента автор предлагает собственную модель, на основе моделе рекуррентной нейронной сети, а именно Gated Recurrent Unit. Каждый домен в нашем случае рассматривается как последовательность символов из фиксированного словаря, которая подаётся на вход рекуррентной нейронной сети. Обучение такой нейронной сети производится методом обратного распространения ошибки таким образом, чтобы максимизировать вероятность правильного выбора соответствующего класса.Для классификации состояние последнего скрытого слоя передается в Softmax слой, выход которого мы можем интерпретировать как вероятности принадлежности домена к одному из классов. Получившиеся модель представлена на рис 4.7.

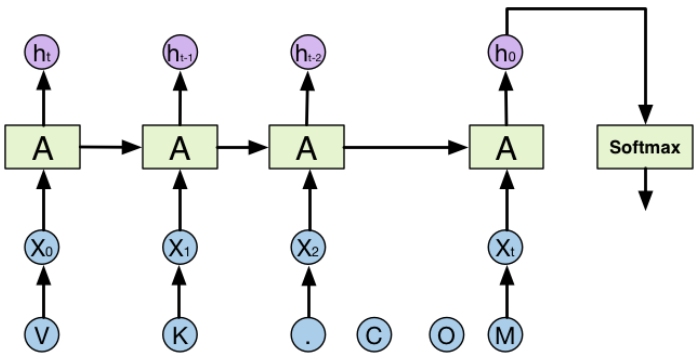

Рис. 4.7 – Модель GRU

В ходе практического исследования данной модели автором найдено оптимальное значение скрытых нейронов. Данное значения равно 128 скрытым

нейронам. Также были протестированы различные алгоритмы стохастического градиентного спуска. Наилучший результат показал алгоритм Adam, предложенный в статье[18]. Обучающая выборка была разбита блоки по 100 доменных имен. В ходе обучения данной модели автором был получен следующий результат - рис 4.8.

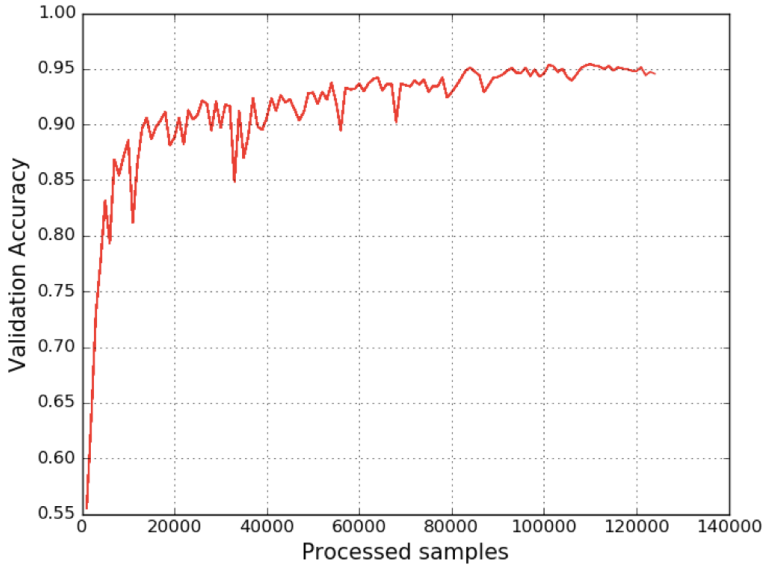

Рис. 4.8 – результат

Однако были предприняты попытки по улучшению существующей модели. Так, одним из возможных методов, является модель Biderection GRU, которая добавляет реверсивный проход по всей длине доменного имени. Это делается для того, чтобы иметь представление не только и прошлых входных значениях, но и будущем контексте. Схематично такая модель изображена на рис 4.9.

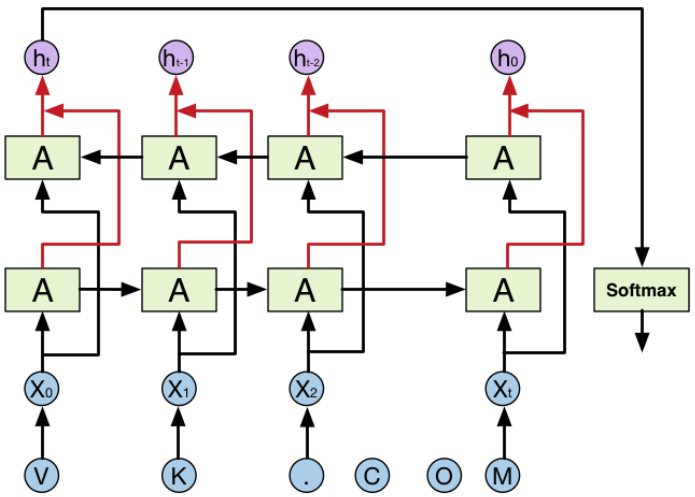

Рис. 4.9 – Модель Biderection GRU

В результате обучения такой модели был получен следующий результат 4.10.

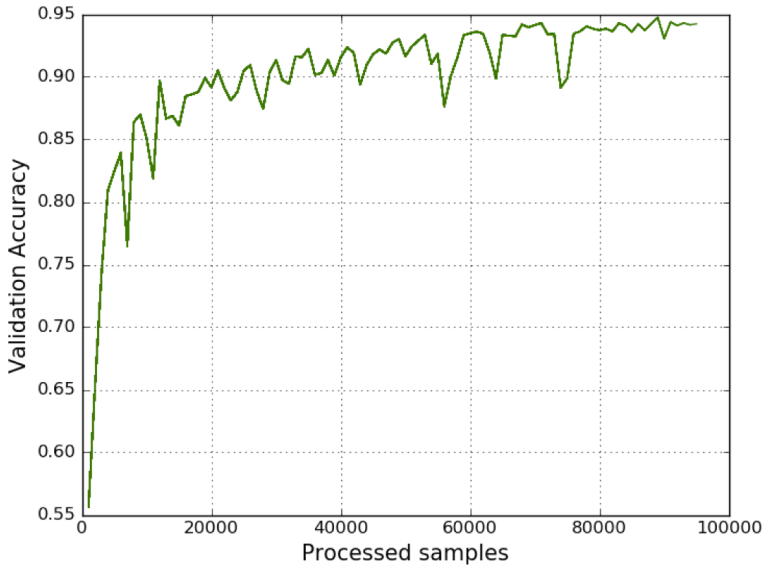

Рис. 4.10 – результат

Однако, как в модели GRU, так и в модели Biderection GRU на вход в softmax слой попадают скрытые состояния лишь последнего скрытого слоя нейронной сети. Для решений этой проблемы автор предлает использование механизма внимания, заключающийся в том, что все скрытые состояния нейронной сети складываются, умножаясь на определенный коэффициент (формулы 4.3 и

4.4), который также является обучаемым параметром нейронной сети.

$$H = a_0 h_0 + ... + a_t ht \tag{4.3}$$

$$\sum_{i=1}^{t} a_i = 1 \tag{4.4}$$

В результате применения механизма внимания наша модель принимает вид рис 4.11, а результаты обучения такой модели представлены на рис 4.12.

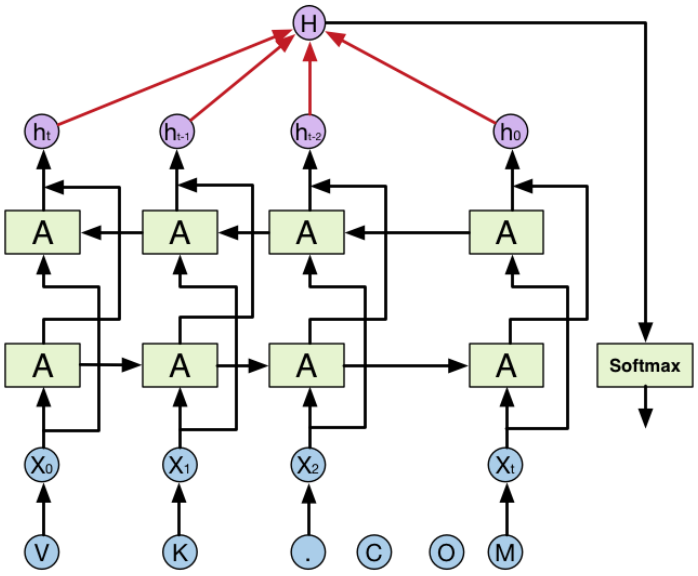

Рис. 4.11 – Модель GRU Attention

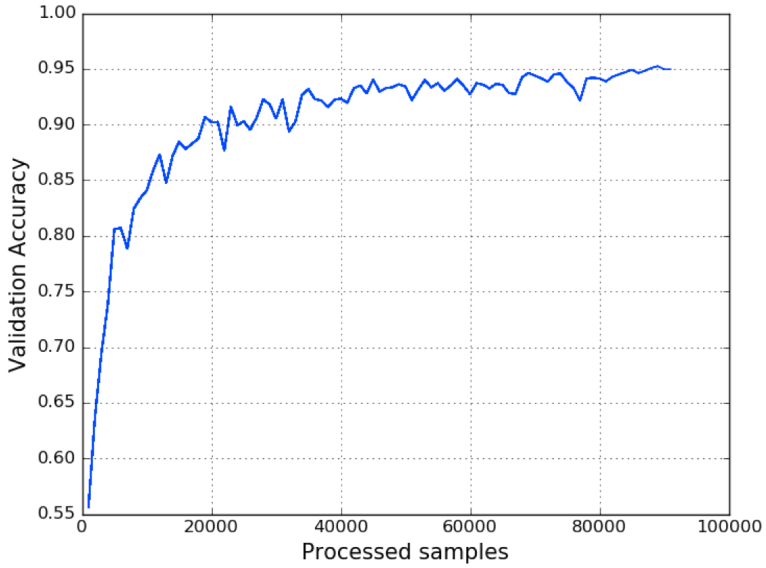

Рис. 4.12 – результат

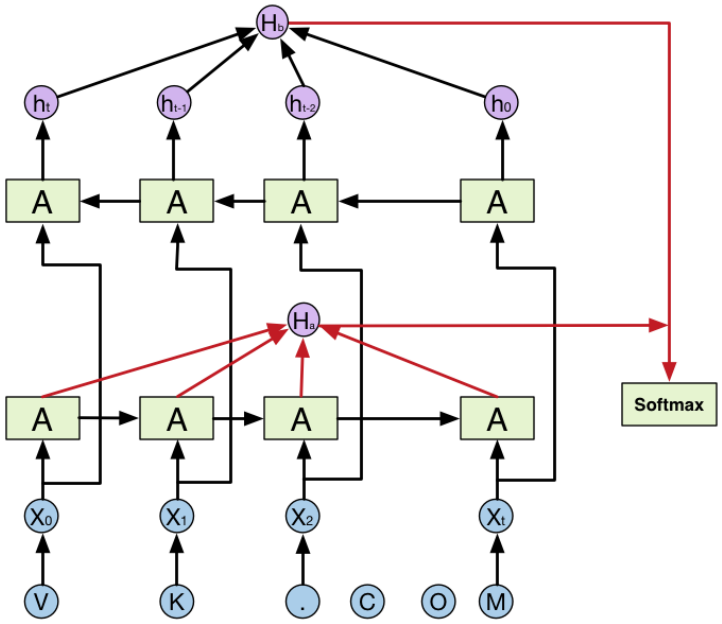

Рис. 4.13 – Модель 2 Attention

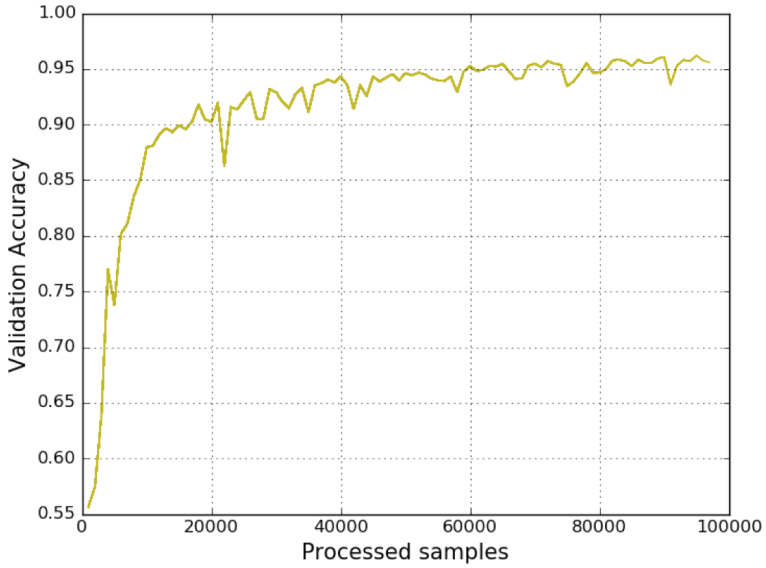

Рис. 4.14 – результат

4.3 Сравнительный анализ

ЗАКЛЮЧЕНИЕ

Выводы, значение полученных результатов Рекомендации по применению

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. T. Barabosch, A. Wichmann, F. Leder, and E. Gerhards-Padilla Automatic extraction of domain name generation algorithms from current malware.// STO-MP-IST-111 Information Assurance and Cyber Defence, 2012.

2. P. Barthakur, M. Dahal, and M. K. Ghose. An efficient machine learning based classification scheme for detecting distributed command & control traffic of p2p botnets.// 5(10):9, 2013.

3. Click Security. Exercise to detect algorithmically generated domain names.// 2014.

4. N. Davuth and S.-R. Kim. Classification of malicious domain names using support vector machine and bigram method.// 7(1):51–58, January 2013.

5. J. Jacobs. Building a dga classifier. [Электронный ресурс] // URL: http://datadrivensecurity.info/blog/posts/2014/Oct/dga-part3/, October 2014 (дата обращения:)

6. Raff. generation Dgas: A evolution. [Электронный ресурс] // URL: http://www.seculert.com/blog/2014/11/dgas-a-domain-generation-evolution, November 2014. (дата обращения:)

7. M. Stevanovic and J. Pedersen. Machine learning for identifying botnet network traffic //, 2013.

8. Z. Wei-wei, G. Jian, and L. Qian. Detecting machine generated domain names based on morpheme features.// pages 408–411, october 2013.

9. S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan. Detecting algorithmically generated domain-flux attacks with dns traffic analysis.// 20(5):1663–1677, Oct. 2012.

10. Н. О. Гончаров. Современные угрозы ботсетей.// 10, октябрь 2014.

11. Machine Learning Text feature extraction (tf-idf) [Электронный ресурс] // URL: http://blog.christianperone.com/?p=1589 (дата обращения:)

12. Understanding LSTM Networks [Электронный ресурс] // URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата обращения:)

13. Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau A C-LSTM Neural Network for Text Classification // 30 Nov 2015

14. Fidelis Threat Advisory 1016// Pushdo It To Me One More Time// 16 Apr 2015

15. Behind the Ramdo DGA // [Электронный ресурс] // URL: https://www.damballa.com/behind-ramdo-dga-domain-generation-algorithm/ (дата обращения:)

16. Daniel Lowd, Pedro Domingos Naive Bayes Models for Probability Estimation // International Conference on Machine Learning // 2005

17. R. J. Williams and D. Zipser.// Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity. In Y. Chauvin and D. E. Rumelhart, editors, Back-propagation: Theory, Architectures and Applications// pages 433–486. Lawrence Erlbaum Publishers// 1995

18. Diederik P. Kingma and Jimmy Ba/ ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION// Thu, 01 Jan 2015