ЗАДАНИЕ НА ВКР

АННОТАЦИЯ

СОДЕРЖАНИЕ

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ГПСЧ - Генератор псевдослучайных чисел

DGA - Domain generation algorithm

LSTM - Long short-therm memory

GRU - Gated recurrent unit

TOR - The onion router

DDOS - Distributed denial of service

HTTP - HyperText transfer protocol

IP - Internet protocol

GMT - Greenwich mean time

t-SNE - t-distributed stochastic neighbor embedding

СЛОВАРЬ ТЕРМИНОВ

Ботнет: Некоторая сеть, в том числе компьютерная, состоящая из устройств - ботов, со специально запущенным вредоносным программным обеспечением.

Вредоносное доменное имя: Доменное имя, искуственно полученное в результате работы алгоритма генерации доменных имен и используемое для установки соединения с управляющем сервером ботнета.

Нейронная сеть: Математическая модель, отображающая деятельность билогического мозга, и позволяющая аппроксимировать какой-либо закон природы.

LSTM: Модель рекуррентной нейронной сети, способная к обучению долгосрочных зависимостей.

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 программным 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 кибератак. Они управляются создателем при помощи специальных управляющих командных серверов (command and control servers). Большинство крупных ботнетов используются для монетизации различными способами, такими как: распределенные атаки отказа в обслуживании (DDoS атаки), атаки на клиентов дистанционного банковского обслуживания, для спама и проведения фишинговых атак. Эти и другие угрозы, которые несут в себе ботнеты, освещены в статье [1].

Для удержания контроля и управления зараженными машинами ботнеты используют множество способов. Это могут быть одноранговые сети, почтовые протоколы, социальные сети или анонимные сети, такие как TOR и i2p. Однако, самым распространенным на данный момент методом являются алгоритмы генерации доменных имен (domain generation algorithms).

Они позволяют удерживать контроль над управляющими серверами. Например, одним из первых случаев их использования 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. Это позволяет семейству этих вредоносных программ обходить существующие механизмы распознавания вредоносных доменных имен. Подробнее история развития алгоритмов генерации доменных имен рассмотрена в статье [18].

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 экспериментальных данных. При этом необходимо составить выборки не только для легитимных и вредоносных имен, но и выборки по отдельным классам вредоносных программ. Под вредоносным доменным именем в данной работе понимается доменное имя, искуственно полученное в результате работы алгоритма генерации доменных имен. Вторая задача - поиск и реализация существующих подходов, их сравнительный анализ на подготовленных данных. Третья - разработка и апробация собственной модели на основе рекуррентных нейронных сетей.

Новизна данной работы заключается в том, что для классификации предлагается использование модели, построенной на основе рекуррентной нейронной сети. Это является новым направлением исследования для решения проблемы обнаружения и распознавания 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 перспектив-

ным направлением в решении проблем 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 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 3 рассмотрены теоретические основы и описана архитектура разработанной модели. Затрагиваются вопросы, связанные с нейронными сетями, рекуррентными нейронными сетями, моделями LSTM и GRU.

Глава 4 посвящена практической реализации моделей. В разделе 4.1 обсуждаются вопросы выбора параметров для классификаторов и полученные результаты. Раздел 4.2 посвящен разработанной модели, получению количественной оценки качества классификации на сформированной выборке доменных имен.

1 АЛГОРИТМЫ ГЕНЕРАЦИИ ДОМЕННЫХ ИМЕН

Алгоритмы генерации доменных имен (DGA) представляют собой алгоритмы, используемые вредоносным программным обеспечением для генерации большого количества псевдослучайных доменных имен, которые позволят им установить соединение с управляющим командным центром. Тем самым они обеспечивают мощный слой защиты инфраструктуры ботнетов. С первого взгляда концепция создания большого количества доменных имен для установки связи кажется несложной, но методы, используемые для создания произвольных строк, часто скрываются за разными слоями обфускации. Это делается, в первую очередь, 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 алгоритмов генерации доменных имен 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 значения времени и даты в формате GMT. Владелец ботнета таким же способом получает текущие значения времени и даты в формате GMT. Далее, эти значения, попадают в сам алгоритм 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 команд.

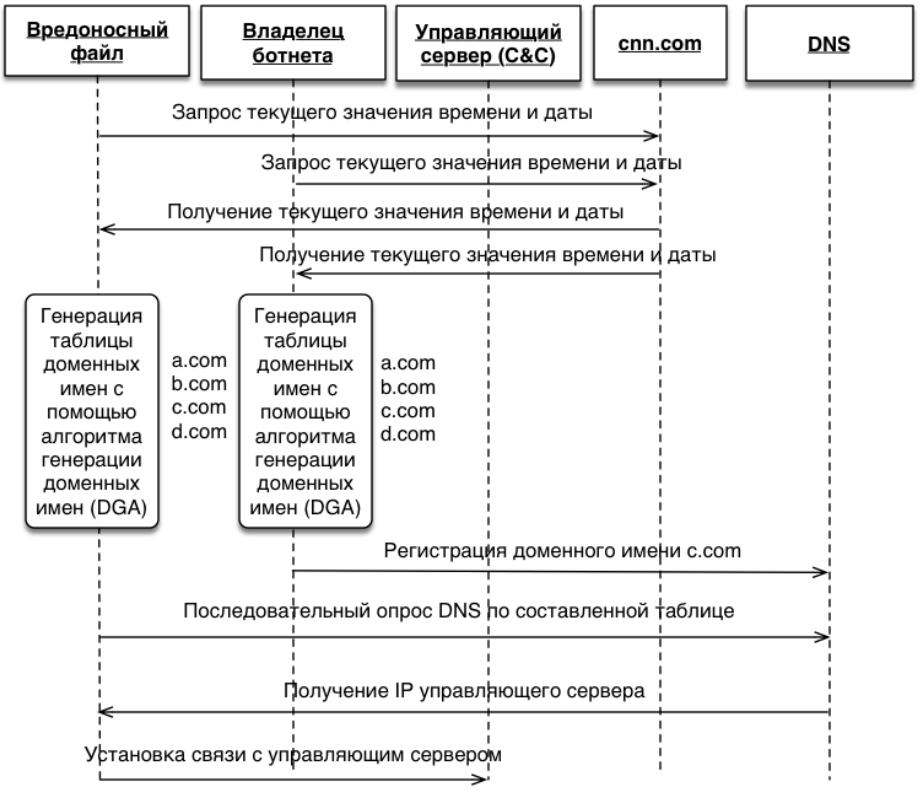

Рисунок 1.1 – Общий принцип работы

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. Данные модели были 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. Этот вирус одним из первых применил технику DGA. Процесс генерации доменного имени можно описать блок-схемой - рис. 1.2, а его архитектура идентична принципу, описанному в разделе 1.1.

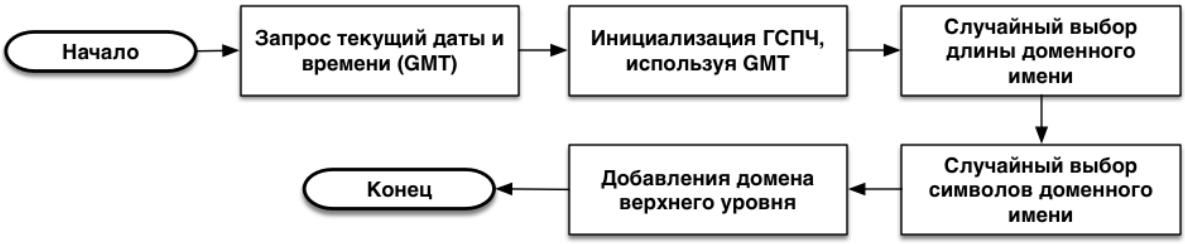

Рисунок 1.2 – Принцип работы conficker DGA

Cryptolocker - имя вредоносных программ вида троян-вымогатель (ransomware). Целью данного вируса являются системы Microsoft Windows. Cryptolocker полностью шифрует содержимое файловой системы жертвы и требует выплаты денежной суммы для получения ключа дешифрования. Cryptolocker DGA 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 1.1.

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

В данной работе за значение ключа взята константа 0x41. Каждый символ рассчитывается по формуле 1.2.

$$chr(ord(a) + (year\ xor\ month\ xor\ date)\ \%\ 25) \qquad (1.2)$$

А длина доменного имени составляет значение, рассчитанное по формуле 1.3.

$$date >> 3\ xor\ year >> 8\ xor\ year >> 11\ and\ 3 + 12 \qquad (1.3)$$

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

PushDo - 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 вектору инициализации присваивается значение системного времени, которое получается путем вызова WINAPI функции GetLocalTime, операций битового сдвига и XOR. После этого, вектор инициализации попадает на вход функции хеширования MD5, вывод которой, в свою очередь, попадает в генератор доменного имени и в 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.

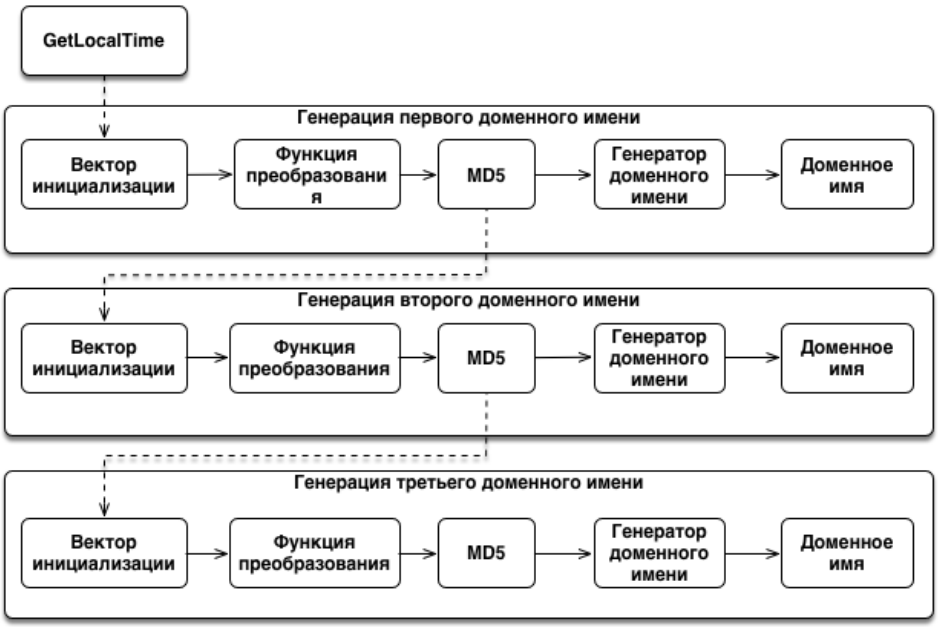

Рисунок 1.3 – Принцип работы PushDo DGA

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 версией 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 каждый день PushDo DGA генерирует 30 доменных имен, зависящих не только от системного времени, но и от зашитых значений - специальных строк генератора.

Zeus - семейство вредоносных программ, ворующих аутентификационные 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, используемым в ботнете PushDo. Алгоритм также использует системное время, алгоритм хеширования MD5 и генератор доменного имени. Однако, в отличие от PushDo, системное время является вектором инициализации для всех доменных имен, генерируемых каждую неделю, а номер доменного имени является параметром salt 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;
}
```

Листинг 1.1 - Генератор доменного имени

Семейство Ramdo, впервые обнаруженное в декабре 2013 года, кардинально отличается от других ботнетов. Ramdo использует 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, основанном на регистре сдвига и булевой операции исключающее или. Для формализации модели DGA данного семейства в работе использован исходный код из работы [12]. Кроме этого, идентичную модель DGA использует и ботнет 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. 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-gram. Rovnix первым из ботнетов начал 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 до достижения определенной длины. Ботнет Matsnu имеет схожий подход, вредоносная программа исполь-

зует 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 до достижения необходимой длины доменного имени. Стоит отметить, что подходы этих двух ботнетов действительно с легкостью обходят системы распознавания, 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 настоящее время существует множество работ, связанных с анализом алгоритмов генерации доменных имен. Проблемы автоматического анализа алгоритмов DGA и пути их решения можно найти в статье [11]. Большие объемы данных, которые можно получить путем мониторинга сетевой активности ботнетов или путем обратной разработки алгоритмов генерации доменных имен, позволяют использовать методы машинного обучения. Идея их применения освещена в работе [8]. 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, компания OpenDNS применяют данные методы в своём облачном DNS сервере. Это позволяет ей эффективно бороться с растущим числом ботнетов. А компания Click Security в своей работе [14] предлагают использовать решающие деревья для бинарной классификации на принадлежность доменов к вредоносным. Для этого ими предложен способ выделения признаков из домена. Стоит отметить работу [5], которая рассматривает возможность классификации, используя метод опорных векторов (support vector machine) и выделения из доменов признаков N-gram. Схожий подход описывается и в работе [16]. Однако, в отличие от работы [5], имеет большую практическую значимость и предполагает использование алгоритма C4.5 (алгоритм для построения деревьев решений). Подход, основанный на анализе морфем в статье [9], является неактуальным, так как последние исследования, опубликованные в работе [18], показывают, что алгоритмы генерации доменных имен совершенствуются с целью обхода существующих способов обнаружения. В результате анализа существующих работ автором выделены 5 наиболее перспективных подходов к классификации, а именно:

- Naive Bayes
- Logistic Regression

– Random Forest
– Extra Tree Forest
– Voting Classification.

В качестве параметров для этих моделей выделяются такие признаки, как: длина доменного имени, энтропия доменного имени и признаки, полученные с использованием моделей N-gram и TF-IDF (term frequency–inverse document frequency). Получение этих признаков описано в разделе 4.1.

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 активируется 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, важным отличием 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 слои 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 достаточное количество нелинейных связей, с определенной точностью способен аппроксимировать любую непрерывную функцию. Исходя из этого, многослойный перцептрон часто используется для аппроксимации 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 j мы обозначим это как h_j:

$$h_j = \sum_{i=1}^{I} w_{ij} x_i + b_j, \tag{3.1}$$

где w_{ij} и b_j - веса нейронной сети. Далее h_j подаётся в функцию активации. Таким образом, каждый скрытый нейрон состоит из сумматора и функции активации. В качестве функции активации могут выступать различные функции, но наиболее часто используемые - сигмоид и гиперболический тангенс:

$$\sigma(h) = \frac{1}{1 + e^{-h}} \tag{3.2}$$

$$tanh(h) = \frac{e^{2h} - 1}{e^{2h} + 1} \tag{3.3}$$

Одно из свойств сигмоида позволяет усиливать слабые сигналы, независимо от входных слишком сильных сигналов. В отличие 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, данное свойство позволяет строить нейронные сети с несколькими скрытыми слоями, в отличии от нейронных сетей с линейной функцией активации, 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 функций заключается в их дифференцируемости. Это позволяет обучать нейронную сеть при помощи алгоритма градиентного спуска. А то, что производные этих функций могут быть легко выражены через самих себя, существенно сокращает вычислительную сложность метода обратного распространения ошибки.

$$\frac{\partial \sigma(h)}{\partial x} = \sigma(h)(1 - \sigma(h)) \tag{3.4}$$

$$\frac{\partial tanh(h)}{\partial h} = 1 - tanh(h)^2 \tag{3.5}$$

После первого скрытого слоя значения передаются в следующий слой, где они опять проходят процесс суммирования и попадают в функцию активации. Они проходят все скрытые слои, пока не попадут в выходной слой нейронной сети.

Количество выходных нейронов и вид выходной функции 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:

$$y_k = \frac{e^{h_k}}{\sum_{k'=1}^{K} e^{h_{k'}}}, \tag{3.6}$$

где K - количество нейронов выходного слоя. В результате на выходе мы имеем K выходных значений, которые мы можем интерпретировать как вероятности принадлежности к конкретному классу. Поэтому многослойный перцептрон так часто используется для задач классификации. В процессе управляемого обучения нейронной сети ставится задача минимизации целевой функции. Часто целевую функцию называют функцией потерь. Пусть z - правильно выбранный класс, тогда функцию потерь для бинарной классификации можно представить в виде:

$$\mathcal{L}(y, z) = (z - 1) \ln(1 - y) - z \ln y \tag{3.7}$$

Для задачи многоклассовой классификации используют перекрёстную энтропию (cross entropy). Пусть z - вектор, представляющий собой нули на всех позициях, кроме одной - позиции, которая определяет соответствующий класс и принимает значение единицы. Так, для класса номер 3 (выборка из 9 классов) вектор z примет значение $[0, 0, 1, 0, 0, 0, 0, 0, 0]$. Тогда перекрестная энтропия примет вид:

$$\mathcal{L}(y, z) = -\sum_{K} z_k \ln y_k \tag{3.8}$$

Поскольку все операции 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, для того, чтобы увеличить вероятность выбора правильного класса при

задаче классификации. Для эффективного расчета градиента в данной работе используется алгоритм обратного распространения ошибки, описанный в работе [10]. Последовательность таких шагов в конце концов приведет к минимуму того или иного типа. Определенную трудность здесь вызывает выбор величины шага. При большой длине шага сходимость будет более быстрой, но имеется опасность перепрыгнуть через решение или уйти в неправильном направлении. Напротив, при малом шаге, вероятно, будет выбрано верное направление, однако при этом потребуется очень много итераций. На практике величина шага выбирается пропорциональной крутизне склона - градиенту функции потерь.

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.

3.1 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 мы имеем вектор скрытых состояний h_t.

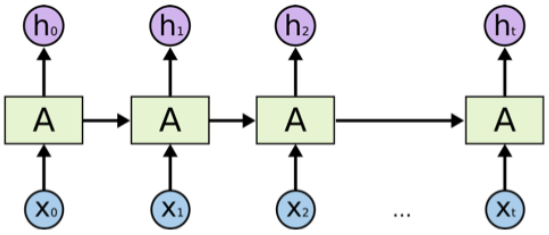

Рисунок 3.1 – Рекуррентная нейронная сеть

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 возникают проблемы вычисления градиента (затухания или резкое его возрастание). Решение этих проблем было найдено в 1997 году учеными Hochreiter & Schmidhuber. В своей работе они предложили новую модель рекуррентной нейронной сети, а именно 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 полносвязнных подсетей, известных как блоки памяти. Вместо одного слоя нейронной сети, в данной модели используется 4 слоя, взаимодействующих особым образом. Главное в модели LSTM – это ячейки памяти, обозначим их как вектор c. На каждом шаге t модель имеет возможность удалить или добавить информацию в память, и этот процесс управляется структурами, которые называют gates. Всего имеется 3 таких структуры: input gate, forget gate, output gate.

На первом этапе в моделе LSTM мы решаем, какую информацию хотим выбросить из каждой ячейки памяти в c_t. Для этого вектор h_{t-1} попадает в forget gate:

$$f_t = \sigma(x_t W_{xf} + h_{t-1} W_{hf} + b_f) \tag{3.9}$$

Проходя через сигмоид, на выходе мы получаем значение от 0 до 1. 1 означает “забыть все”, в то время как 0 означает “оставить все”. На следующем этапе мы определяем, какую информацию мы хотим добавить в каждую ячейку памяти c_t. Для этого выполняется 2 операции. Во-первых, input gate решает, какую информацию мы обновляем (формула 3.10). Во-вторых, слой

с функцией активации гиперболический тангенс создает вектор значений $\tilde{c}_t$, который будет добавлен в c_t (формула 3.11).

$$i_t = \sigma(h_{t-1}W_{hi} + x_tW_{xi} + b_i) \tag{3.10}$$

$$\tilde{c}_t = tanh(h_{t-1}W_{hc} + x_tW_{xc} + b_c) \tag{3.11}$$

Теперь, используя информацию, полученную на предыдущих этапах (формулы 3.9, 3.10, 3.11), 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 $i_t \odot \tilde{c}_t$, чтобы обновить c_{t-1}, где $\odot$ - операция поэлементного умножения матриц. В итоге получим:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{3.12}$$

В завершении, вектор h_{t-1} попадает в output gate, проходя через функцию активации сигмоид (формула 3.13), а новый вектор h_t получается путем перемножения выхода output gate со значением $tanh(c_t)$ (формула 3.14).

$$o_t = \sigma(h_{t-1}W_{ho} + x_tW_{xo} + b_o) \tag{3.13}$$

$$h_t = o_t \odot tanh(c_t) \tag{3.14}$$

Таким образом, автором описана распространенная версия 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, предложенная в работе [13]. Данная модель отличается тем, что forget gate и input gate объединены в update gate и описывается следующими формулами:

$$u_t = \sigma(h_{t-1}W_{hu} + x_tW_{xu} + b_u) \tag{3.15}$$

$$r_t = \sigma(h_{t-1}W_{hr} + x_tW_{xr} + b_r) \tag{3.16}$$

$$\tilde{h_t} = tanh(r_t \odot (h_{t-1}Wh) + x_tWx + b_h) \tag{3.17}$$

$$h_t = (1 - u_t) \odot h_{t-1} + u_t \odot \tilde{h_t} \tag{3.18}$$

Говоря про модель GRU, стоит отметить, что она гораздо проще в реализации, требует меньшей вычислительной мощности и оперативной памяти. Несмотря на это, её эффективность не уступает модели LSTM, а в некоторых случаях даже превосходит её, и, как позывают практические исследования, представленные в работе [4], невозможно однозначно сделать вывод, какая из этих двух моделей является наилучшей.

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 1 000 000 Legit доменов и 100 000 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 признаков: length, entropy, alexa gram, word gram, diff. Первым признаком выступает длина доменного имени. Второй признак - энтропия, рассчитывалась по формуле 4.1, где $p(i)$ - вероятность возникновения события i, то есть появления символа из исходного алфавита с n символами.

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

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 [17]. Для выделение данных признаков автором 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
```

Листинг 4.1 - Получение признаков Alexa gram, Word gram

В результате выполнения программы получено 2 признака:

– 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 признаки, автор решил добавить в модель признак diff:

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

Для анализа каждого из предложенных признаков были построены наглядные графики. Исходя из рис. 4.1 можно заметить, что вредоносные доменные имена не имеют длину менее 6 символов, что оправдывается высокой стоимостью таких доменных имен и повышенной вероятностью появления коллизий. Графики энтропии рис. 4.1 и рис. 4.2 доказывают, что, основываясь лишь на этом признаке, невозможно выделить вредоносные доменные имена

в отдельный класс, так как множество значений энтропии вредоносных доменных имен является подмножеством множества значений энтропии легитимных доменных имен. А признак diff на рис 4.3 был добавлен в модель после анализа признаков alexa gram и word gram. Вместе, эти признаки, позволяют выделить отдельные доменные имена в класс вредоносных.

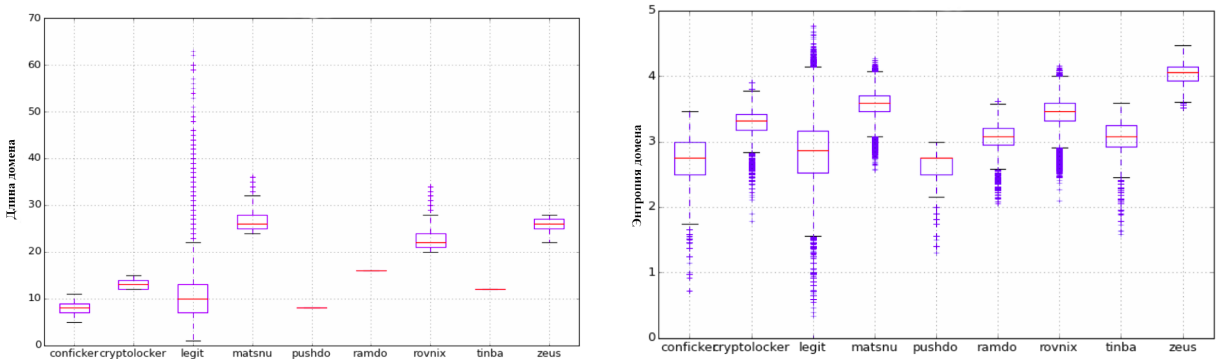

Рисунок 4.1 – Длина и энтропия доменных имен

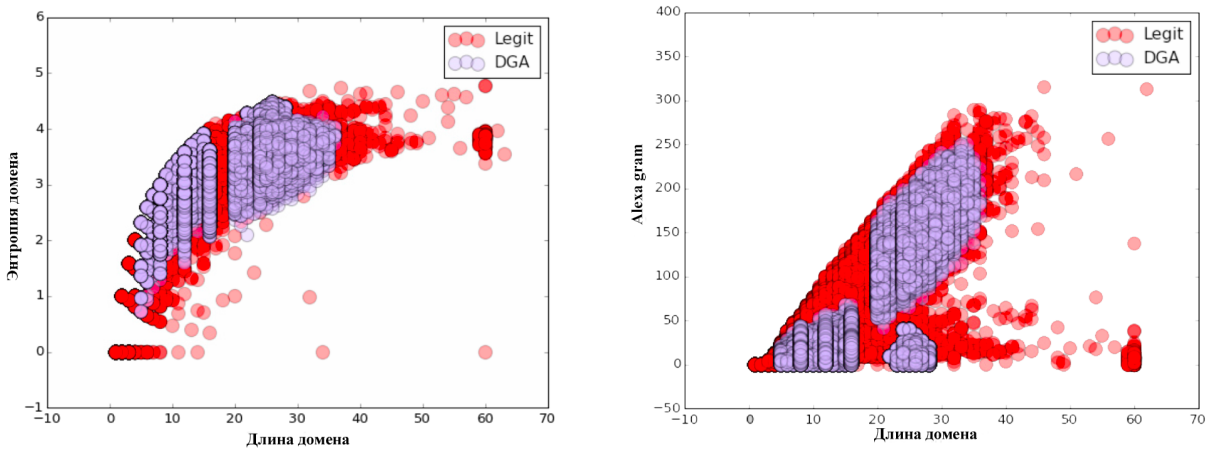

Рисунок 4.2 – Зависимость энтропии и признака alexa gram от длины доменного имени

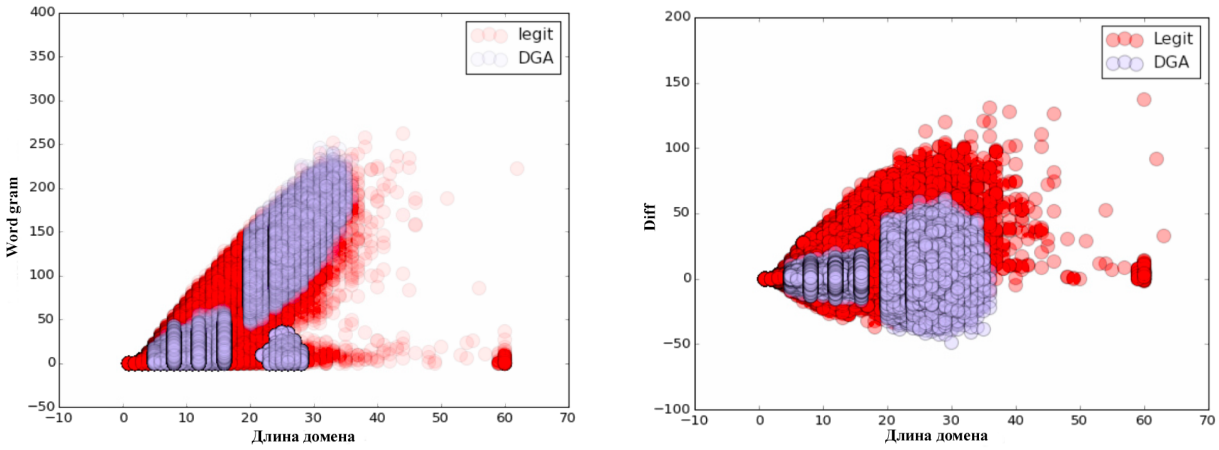

Рисунок 4.3 – Зависимость признаков word gram и diff от длины доменного имени

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%	89%
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. Однако, 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).

Таблица 4.2 – Матрица неточностей

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

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 отнесения доменнного имени к классу вредоносных.

4.2 Рекуррентные нейронные сети

В качестве эксперимента автор предлагает собственную модель, на основе модели рекуррентной нейронной сети. Каждый домен в этом 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 9 классов обучающей выборки (легитимные доменные имена и 8 семейств вредоносных программ). Полученная модель представлена на рис. 4.4.

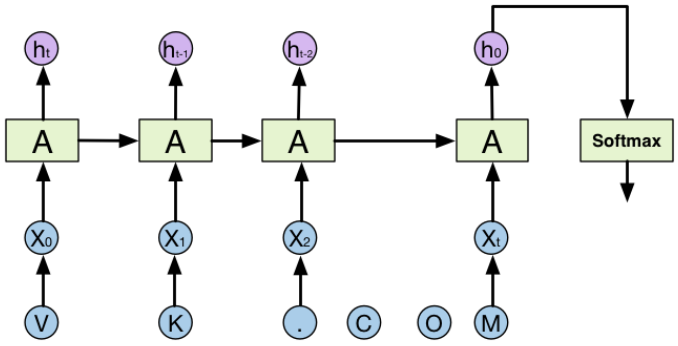

Рисунок 4.4 – Схема Модели 1

В ходе практического исследования данной модели автором найдено оптимальное количество скрытых нейронов. Оно равно 128 скрытым нейронам. Кроме этого были протестированы различные алгоритмы стохастического градиентного спуска, такие как:

- SGD;
- Adadelta;
- Rmsprop;
- Adam.

В результате наилучший результат показал алгоритм Adam - рис. 4.5, предложенный в статье [6]. Он является простым в реализации, вычислительно эффективным алгоритмом, имеет небольшие требования к памяти и хорошо подходит для задач, связанных с большим количеством данных и признаков.

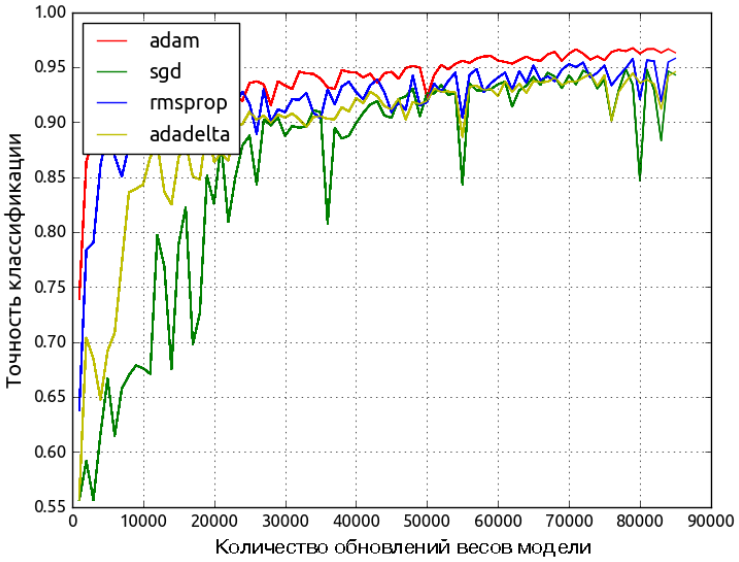

Рисунок 4.5 – Алгоритмы стохастического градиентного спуска

Далее, обучающая выборка была разбита на произвольные блоки (mini-batch) по 100 доменных имен. Обновление весов сети происходило для каждого такого блока. В ходе обучения данной модели автором был получен следующий результат - рис. 4.6.

Рисунок 4.6 – Результат обучения Модели 1

Однако были предприняты попытки по улучшению существующей модели. Так, одним из возможных методов повышения качества классификации является Модель 2, которая добавляет реверсивный проход по всей длине доменного имени. Это делается для того, чтобы иметь представление не только о прошлых входных значениях, но и о будущем контексте. Такая модель изображена на рис. 4.7.

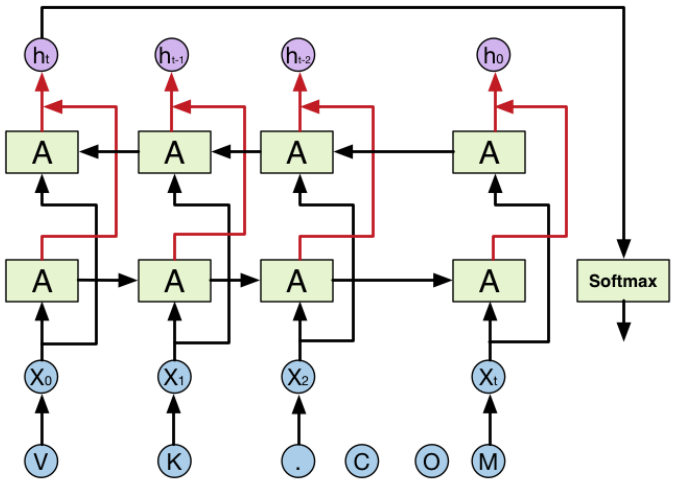

Рисунок 4.7 – Схема Модели 2

Как в Модели 1, так и в Модели 2 на вход в Softmax слой попадают состояние лишь последнего скрытого слоя нейронной сети. Поэтому в качестве эксперимента автор предлает использовать механизм внимания [20], заключающийся в том, что все скрытые состояния нейронной сети складываются, умножаясь на определенный коэффициент, который также является обучаемым параметром нейронной сети:

$$h_a = a_0 h_0 + ... + a_t h_t \tag{4.3}$$

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

В результате применения механизма внимания наша модель принимает вид рис. 4.8.

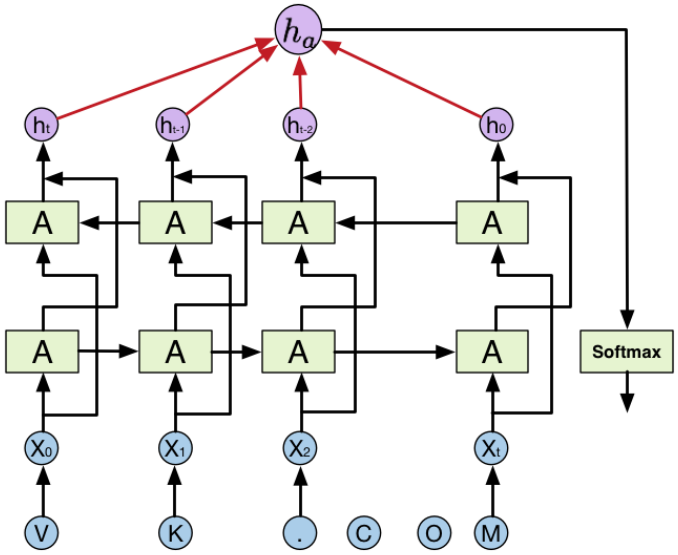

Рисунок 4.8 – Схема Модели 3

Далее была разработана еще одна модель - Модель 4 рис. 4.9. В отличии от Модели 3, данная модель применяет механизм внимания два раза, как на прямой - h_a, так и на обратный проход нейронной сети - h_b.

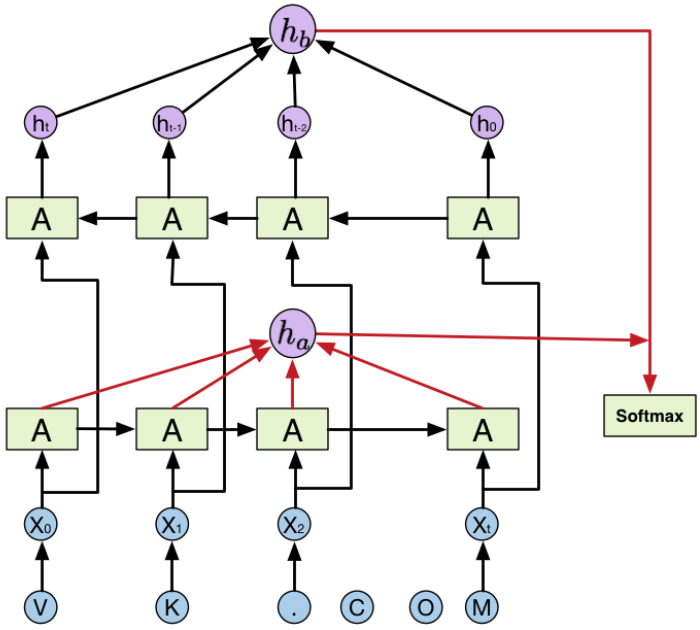

Рисунок 4.9 – Схема Модели 4

Для оценки сложности полученных моделей было подсчитано количество обучаемых параметров - таблица 4.3. Она даёт представление о том, насколько отличаются по сложности каждая из предложенных моделей.

Таблица 4.3 – Сложность моделей

Модель	Количество обучаемых параметров
Модель 1	67322
Модель 2	133114
Модель 3	249082
Модель 4	332282

Используя алгоритм нелинейного снижения размерности и визуализации многомерных переменных t-SNE (t-distributed stochastic neighbor embedding, стохастическое вложение соседей с распределением Стьюдента), описанный в работе [7], была произведена визуалиазация данных на первых и последних шагах обучения модели при задаче многоклассовой классификации рис. 4.10. Алгоритм t-SNE обеспечивает эффективный метод визуализации сложных наборов данных. Он успешно обнаруживает скрытые структуры в данных, демонстрирует группы и компенсирует нелинейные отклонения по измерениям, сохраняя пропорции расстояний.

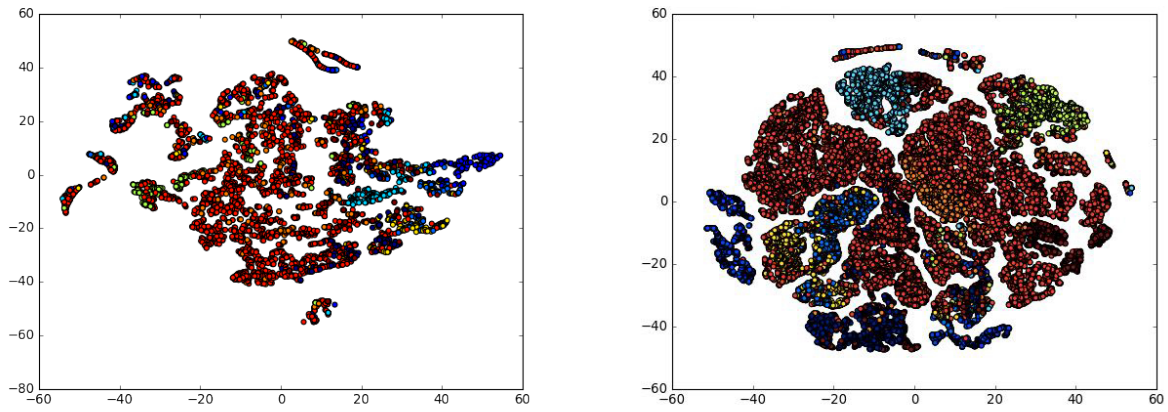

Рисунок 4.10 – Визуализация данных на первых и последних шагах обучения модели

Для оценки качества классификации все 4 модели были построены на 1 графике рис. 4.11: многоклассовая классификация - левый график, бинарная классификация - правый график.

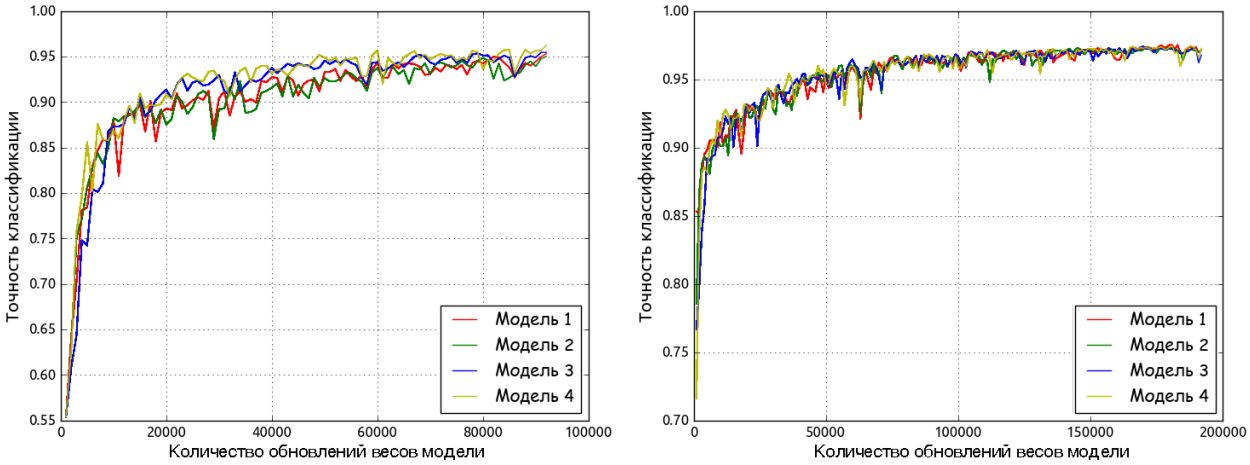

Рисунок 4.11 – Результаты обучения многоклассовой и бинарной классификации

В результате новая модель, построенная на основе рекуррентной нейронной сети повысила качество бинарной классификации до 97%, а Модель 4 показала своё превосходство над другими моделями в многоклассовой классификации.

ЗАКЛЮЧЕНИЕ

В рамках данной работы автором была разработана модель на основе методов машинного обучения для распознавания и классификации вредоносных доменных имен, полученных при помощи анализа алгоритмов генерации доменных имен, а также получена количественная оценка качества классификации на сформированной выборке доменных имен. Ниже представлен детальный список результатов:

- Изучены механизмы работы алгоритмов генерации доменных имен;
- Методом обратной разработки получены и реализованы на языках высокого уровня алгоритмы генерации доменных имен восьми семейств ботнетов;
- Составлена обучающая выборка, состоящая из легитимных и вредоносных доменных имен;
- Рассмотрены существующие подходы к распознаванию и классификации вредоносных доменных имен;
- Произведена реализация существующих подходов и получены результаты по каждому из них;
- Разработана собственная модель, на основе рекуррентной нейронной сети;
- Произведена реализация и апробация собственной модели распознавания вредоносных доменных имен.

В итоге, разработанная модель показывает результат, превосходящий существующие подходы. Данная модель позволяет на тестовой выборке правильно распознать до 97% вредоносных доменных имен. Что касается практической применимости, то данная разработка может быть применена и внедрена в такие продукты как: DNS сервера, системы обнаружения вторжений, антивирусные продукты. Это позволит повысить защищенность пользователей от угроз, связанных с распространёнными семействами ботнетов. Такие технологии только выходят на рынок информационной безопасности.

Одним из ярких примеров применения похожего подхода является облачный DNS сервис от компании OpenDNS. Однако стоит отметить, что процесс распознавания все же остается итерационным, так как постоянное появления всё новых DGA ведет к ухудшению качества модели. Поэтому совершенствование модели должно производиться на всем этапе её жизненного цикла.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Н. О. Гончаров. Современные угрозы ботсетей. [Электронный ресурс] // Молодежный научно-технический вестник №10. - октябрь 2014. URL: http://sntbul.bmstu.ru/file/out/734781. Режим доступа: свободный, дата обращения: 20.04.2016.

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

3. P. Barthakur, M. Dahal, and M. K. Ghose. An efficient machine learning based classification scheme for detecting distributed command & control traffic of p2p botnets. // International Journal of Modern Education and Computer Science. - 5(10):9, 2013.

4. Junyoung Chung. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. // NIPS 2014 Deep Learning and Representation Learning Workshop. - 2014.

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

6. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. // 3rd International Conference for Learning Representations. - San Diego, 2015.

7. L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. // Journal of Machine Learning Research 9(Nov):2579-2605, 2008.

8. M. Stevanovic and J. Pedersen. Machine learning for identifying botnet network traffic. // Technical report, Aalborg Universitet. - 2013.

9. Z. Wei-wei, G. Jian, and L. Qian. Detecting machine generated domain names based on morpheme features. // International Workshop on Clouad Computing and InformationSecurity (CCIS 2013). - Oct. 2013.

10. 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 // Lawrence Erlbaum Publishers. - pages 433–486, 1995.

11. S. Yadav, A. K. K. Reddy, A. L. N. Reddy, and S. Ranjan. Detecting algorithmically generated domain-flux attacks with dns traffic analysis. // IEEE/ACM Transactions on Networking (Volume:20 , Issue: 5). - 20(5):1663–1677, Oct. 2012.

12. Behind the Ramdo DGA. [Электронный ресурс] // Damballa. URL: https://www.damballa.com/behind-ramdo-dga-domain-generation-algorithm/ Режим доступа: свободный, дата обращения: 20.04.2016.

13. Kyunghyun Cho. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. [Электронный ресурс] // arXiv.org. URL: https://http://arxiv.org/abs/1406.1078/ Режим доступа: свободный, дата обращения: 13.04.2016.

14. Click Security. Exercise to detect algorithmically generated domain names. [Электронный ресурс] // Github. URL: https://github.com/ClickSecurity/datahacking/ Режим доступа: свободный, дата обращения: 12.02.2016.

15. Fidelis Threat Advisory #1016. Pushdo It To Me One More Time. [Электронный ресурс] // Threatgeek. URL: http://www.threatgeek.com/2015/04/fidelis-threat-advisory-1016-pushdo-it-to-me-one-more-time.html Режим доступа: свободный, дата обращения: 20.04.2016.

16. J. Jacobs. Building a dga classifier. [Электронный ресурс] // Data Driven Security. URL: http://datadrivensecurity.info/blog/posts/2014/Oct/dga-part3/, Oct. 2014 Режим доступа: свободный, дата обращения: 12.02.2016.

17. Christian S. Perone. Machine Learning Text feature extraction (tf-idf). [Электронный ресурс] / Christian S. Perone. // Terra Incognita by Christian S. Perone. URL: http://blog.christianperone.com/?p=1589 Режим доступа: свободный, дата обращения: 25.01.2016.

18. Raff. generation Dgas: A evolution. [Электронный ресурс] // Seculert. URL: http://www.seculert.com/blog/2014/11/dgas-a-domain-generation-evolution, Nov. 2014. Режим доступа: свободный, дата обращения: 12.02.2016.

19. Understanding LSTM Networks. [Электронный ресурс] // Colah Blog. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs. Режим доступа: свободный, дата обращения: 20.04.2016.

20. Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, Yoshua Bengio Show, Attend and Tell: Neural Image Caption Generation with Visual Attention [Электронный ресурс] // arXiv.org. URL: https://arxiv.org/abs/1502.03044 Режим доступа: свободный, дата обращения: 13.04.2016.

21. Chunting Zhou, Chonglin Sun, Zhiyuan Liu, Francis C.M. Lau. A C-LSTM Neural Network for Text Classification. [Электронный ресурс] // arXiv.org. URL: http://arxiv.org/abs/1511.08630 Режим доступа: свободный, дата обращения: 15.04.2016.