КЛАССИФИКАТОР ДЛЯ СТАТИЧЕСКОГО ОБНАРУЖЕНИЯ КОМПЬЮТЕРНЫХ ВИРУСОВ, ОСНОВАННЫЙ НА МАШИННОМ ОБУЧЕНИИ

Евгений Путин, Адиль Тимофеев

Аннотация: *Обсуждаются оригинальные подходы к построению системы обнаружения компьютерных вирусов. Обосновывается актуальность разработки и реализации эффективного классификатора как для известных, так и для ранее неизвестных вирусов (вирусов нулевого дня). Даётся общий подход к построению классификатора для статического обнаружения вирусов с использованием машинного обучения с учетом анализа последних исследований в данной области. Описываются результаты новых исследований, и обосновывается выбор итоговой агрегационной модели – стекового классификатора.*

Ключевые слова: *классификатор, обнаружение, компьютерные вирусы, машинное обучение*

ACM Classification*: E4. Coding and Information Theory*

Введение

В настоящее время использование глобальной сети Интернет является неотъемлемой частью нашей повседневной жизни. Через браузеры можно скачивать различный контент, в том числе программное обеспечение. Сегодня многие компьютерные системы становятся уязвимы к «зловредным» программам, т.е. программам, нацеленным на нанесение вреда конечному пользователю, или компании. Зловредные программы могут быть категоризированы на несколько групп:

- Вирусы – компьютерные программы, которые размножают себя и внедряется в файлы пользователя или операционной системы;
- Черви – саморазмножающиеся компьютерные программы, которые способны посылать себя на другие компьютеры по локальной сети или Интернету;
- Трояны – программы, которые маскируют себя под желанную функциональность, но на самом деле реализуют другие «невидимые» операции, такие как неавторизированные доступ.
- Шпионы – программы, установленные в компьютере без осведомления пользователя для того, чтобы собирать о нем нужную информацию.

Каждая из указанных групп имеют свою уникальную специфику, но всех их объединяет операционная система Windows (в нашем случае 32-х битная), а значит и единый формат представления исполняемых файлов. Поэтому в дальнейшем под компьютерным вирусом будем просто понимать любую программу (win32) которая причиняет вред.

На Рис. 1 представлена гистограмма роста компьютерных вирусов [1]

Гистограмма проверяемости файлов на заражённость и инфицируемость представлена на Рис. 2 [2].

Как видно из рис. 2, самый проверяемый тип файлов Win32 Exe, что не удивительно из-за множества уязвимостей в Windows OS.

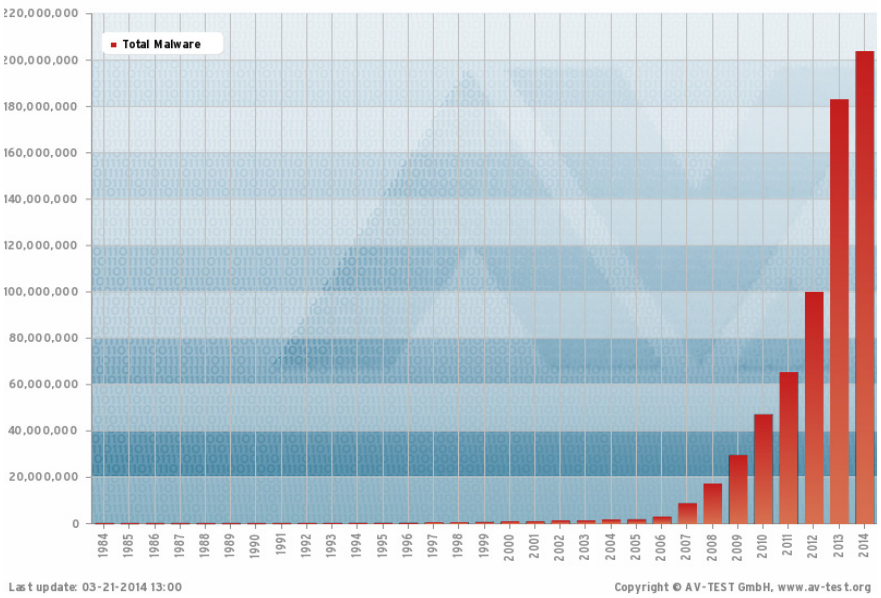

Рисунок 1. Гистограмма роста количества вирусов во времени

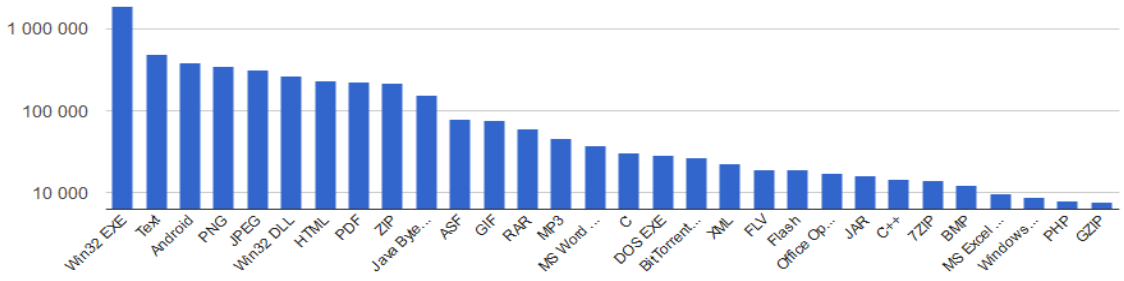

Рисунок 2. Гистограмма проверяемости файлов на зараженность и инфицируемость (по оси Y - количество проверяемых файлов за последние 7 дней (30.03.14), по оси X – типы проверяемых файлов).

1. Подходы к обнаружению вирусов и актуализация

Методы обнаружения вирусов могут быть разделены на 2 класса :

1. Методы, основанные на сигнатурах.
2. Методы, основанные на выявлении аномалий.

В большинстве современных антивирусах центральное место занимает сигнатурный подход. Он даёт 100% точность обнаружения на уже известных вирусах. Но сигнатурный анализ бесполезен на тех вирусах, которые не известны антивирусу, т.е. не известны их сигнатуры. Будем называть такие вирусы, вирусами нулевого дня (zero-day viruses). Аномальный подход наоборот позволяет обнаруживать вирусы нулевого дня. Чаще всего аномальные методы строят классификаторы или решающие правила. Существует немало таких методов [3], но в последнее время наибольший потенциал представляют методы, использующие машинное обучение.

Сигнатурные методы обнаруживают вирус при помощи поиска сигнатур уже известных вирусов в специальном словаре – базе сигнатур вирусов.

Аномальные методы обнаруживают вирус, используя знания, правила и спецификации о нормальном поведении той или иной программы.

Преимущества и недостатки сигнатурного анализа:

1. Позволяет определять конкретный вирус с высокой точностью и малой долей ложных срабатываний.
2. Беззащитен перед полиморфными вирусами.
3. Требует регулярного и крайне оперативного обновления базы сигнатур.
4. На неизвестные вирусы требуются эксперты для ручного анализа вирусов и выделения сигнатур.
5. Неспособен выявить какие-либо новые вирусы, атаки.
6. На разные версии одного и того же вируса необходимы разные сигнатуры.
7. С учетом того, что база сигнатур огромна, сигнатурный анализ очень ресурсоемкая операция.

Преимущества и недостатки аномального анализа:

1. Возможность обнаружения ранее неизвестных вирусов (вирусов нулевого дня (zero-day viruses)).
2. Высокая вероятность ложных срабатываний, т.е. таких при которых доброкачественная программа была распознана как вирус.
3. Высокая сложность обучения системы.
4. Лечение неизвестного вируса практически всегда является невозможным.
5. Для уже обученной системы анализ выполняется сравнительно быстро (нужно лишь извлечь признаки).

И сигнатурные и аномальные методы внутри себя могут использовать три различных подхода для обнаружения вирусов:

1. Статический подход. Используя этот подход, подозрительная программа анализируется статически (т.е. без запуска самой программы) как обычный файл.
2. Динамический подход. При этом подходе, подозрительная программа анализируется динамически, т.е. во время ее выполнения в реальном времени.
3. Гибридный подход. Объединение статического и динамического подходов в разных частях анализа «зловредной» программы.

Ниже на рис. 3 приведена «карта ума» (mind map) по подходам к обнаружению вирусов.

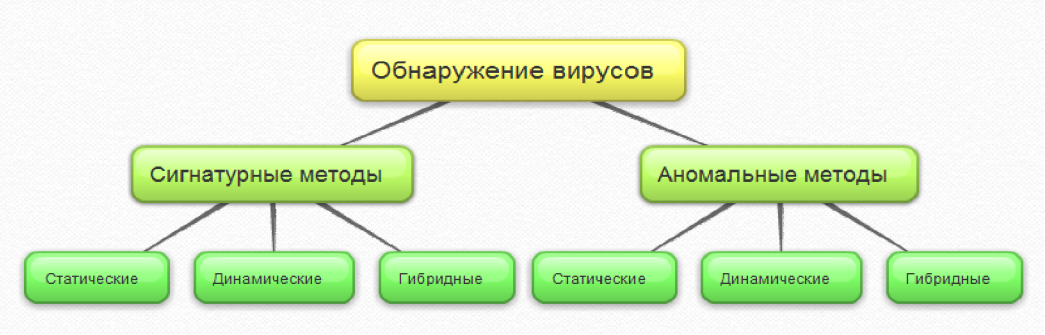

Рисунок 3. Карта ума по методам обнаружения вирусов.

В современном мире разрабатываются таргетированные вирусы и совершаются таргетированные атаки, т.е. такие атаки, которые нацеленные на конкретную компанию, организацию, страну.

К примеру, по опубликованным данным газеты New York Times [4] известный компьютерный червь Stuxnet, который по некоторым предположениям был специально разработан против компьютерной сети ядерного проекта Ирана, использовал 4 уязвимости нулевого дня и 3 известных. Более детальный обзор

вируса Stuxnet и уязвимостей нулевого дня можно найти в последнем [5] Также очень хорошо цифровая актуализация описана в отчете Лаборатории Касперского [6].

Из всего вышесказанного следует необходимость развития, усовершенствования и создания новых методов обнаружения вирусов в области аномального анализа, в том числе и в статическом подходе.

Обнаружение вирусов при помощи методов машинного обучения (МО) представляет огромный потенциал в статистическом обнаружении вирусов, потому что все методы МО призваны обобщать решение конкретной задачи на общий класс аналогичных задач, и показывают практически лучшие результаты во всех задачах классификации и прогнозирования.

2. Общий подход к созданию классификатора

В начале исследований формируется и определяется выборка вирусов и чистых фалов.

Создание классификатора для обнаружения вирусов условно можно разделить на три стадии:

1. Формирование признакового описания файла (features extraction). Результатом этой стадии является вектор, содержащий признаковые характеристики рассматриваемого объекта. В задаче построения классификатора статического обнаружения вирусов признаками могут выступать следующие объекты:

- Строки – исполняемый файл рассматривается как обычная строка или последовательность строк. Признаки – числовые характеристики строк (например, частота нулей в подстроке);
- Структурные элементы Portable Executable файла. Эта специальная информация встроенная во все файлы Win32 и Win64. Она необходима для загрузчика операционной системы Windows и для самого приложения. Подробная документация доступна (см. например,[7]). Признаки, извлеченные из структурной информации PE файлов могут быть следующими: сертификат, date/time stamp, файловый указатель – позиция внутри файла информация компоновщика, тип CPU, логическая информация (выравнивание секций, размер, секции кода, отладочные флаги), информация об импорте – список тех DLL, которые использует исполняемый файл, и экспорте – те функции которые предоставляет другим приложениям, таблицу релокаций (перемещений) директории ресурсов – иконки, кнопки и прочее.
- N-граммы на уровне байт. Сегменты последовательных байт из разных мест внутри исполняемого файла длины N. Каждая N-грамма рассматривается как признак. К примеру, количество байтовых биграмм – 256*256 = 65536 штук. Таким образом, будем рассматривать 65536 признаков.
- N-граммы на уровне опкодов. Опкод (opcode – operation code) – специфичный для CPU операционный код, который выполняет специальную машинную команду (например, mov, push, add).

2. Выбор признаков (feature selection). В течение этой фазы вектор, созданный на стадии 1 вычисляется, а избыточные и нерелевантные признаки выбрасываются из рассмотрения. Выбор признаков имеет много преимуществ: увеличение выполнения обучающейся модели за счет сокращения количества необходимых операций и, как следствие, увеличение скорости обучения, повышение обобщающей способности за счет сокращение размерности пространства признаков, удаление “выбросов”, лучшая интерпретируемость и т.п. Задача этой стадии заключается в том, чтобы из уже имеющихся признаков выбрать наиболее значимые (информативные). Существует несколько подходов к выделению информативности признаков. Наиболее популярными являются корреляционные и фильтровые методы [4].

3. Построение математической модели, классификатора, который использует разряженный вектор, полученной стадии 2. Для построения классификатора могут использоваться следующие математические модели:

- Деревья решений (decision trees, DT);
- Случайный лес (random forest, RF);
- Градиентный бустинг (gradient boosting machines, GBM);
- Логистическая регрессия и ее оптимизации (logistic regression, LR);
- Метод опорных векторов (support vector machines, SVM);
- K-ближайших соседей (K-nearest neighbor, kNN);
- Adaboost;
- Наивный Байес (Naive Bayes, НБ, NB);
- Нейронные сети (НС, NN).

3. Анализ исследований в области создания классификатора статического обнаружения вирусов

По результатам анализа проделанных ранее исследований можно сделать вывод, что не во всех работах присутствуют описанные выше фазы. Например, часто выбор признаков опускается. При этом авторы полагают, что большое количество признаков существенно не ухудшит результирующую модель, так как существуют устойчивые к «передозированным» признакам модели.

Следует сказать, что анализ и сравнение результатов исследований является сложной задачей, поскольку разные авторы используют различные обучающие множества. Кроме того, сравнение моделей происходит по различным метрикам.

В своей работе 2001 года [8] автор предложил следующий метод:

1. Для каждого исполняемого файла рассматриваются три различных признака:

- Список DLL, использующихся внутри исполняемого файла;
- Список системных вызовов DLL;
- Количество различных системных вызовов внутри каждой DLL.

2. Выбор признаков, осуществлялся специальным индуктивным алгоритмом Ripper [9] для нахождения паттернов в данных DLL.

3. В качестве математической модели выступал Наивный Байесовский (NB) классификатор, который использовался для нахождения паттернов в строковых данных, а N-граммы последовательностей байт были использованы, как вход для Мультиномиального Наивного Байеса.

Обучающее множество состояло из 4266 файлов, из которых было 3265 вирусов и 1001 чистых файлов. Точность классификации для такого множества, подхода и модели была 97,11%.

В работе за 2006 год [10] использовался N-граммный по байтам подход. Их обучающее множество состояло из 1971 чистых файлов и 1651 зловредных. Выбор признаков проходил с использованием Information Gain. Были выбраны 500 наиболее часто встречаемых N-грамм. Ученые пробовали множество различных моделей (Naïve Bayes, SVM, KNN и др.), но лучшая из них – Adaboost дала 0.98 точности с 0,05% ложных срабатываний.

В работе [11] авторы использовался близкий к подходу, описанному в [8], а именно:

1. Из каждого исполняемого файла извлекались:

- Структурные признаки формата PE: вся информация о заголовках PE и секциях;
- Список всех используемых DLL;
- Список всех функций внутри каждой DLL.

2. Все эти признаки были отфильтрованы по Information Gain (IG) и были выбраны 20 наиболее информативных признаков.

3. Рассматривалось несколько моделей (SVM, NB, DT), причём лучшей оказалась DT (C4.5 реализация) которая показала результат в 99,6% точности с 2,7% ложных срабатываний.

Исследователи использовали открытую вирусную коллекцию VX heaven [12] (которая на 2010 год насчитывала ~ 230 тысяч PE вирусов, червей и агентов) и собранную коллекцию чистых файлов в 10592. При этом много внимания уделялось выбору значимых признаков и построению моделей на различных подмножествах признаков, в том числе уменьшению размерности (пространства признаков), что дало чуть худший результат, чем ранее приведенный.

В фундаментальной работе [13] исследовался подход, основанный на опкодах. В качестве обучающего множества была взята VX heaven коллекция. В работе [13] дается детальный анализ использования и выделения опкодовых N-граммов. Перебором различных N лучший результат достигается на N = 2. При этом производилось построение множества моделей (RF, DT, NB, KNN, SVM) с оптимизациями, и сравнение моделей. Была выбрана лучшая модель, а именно SVM, с полиномиальным ядром для биграмм дала 95,90% точности с 0,03% ложных срабатываний.

В работе [14] использовался строковый подход. Для большей масштабируемости подхода использовались ансамбли SVM с бэгингом (Bagging, Bootstrap Aggregating). Обучающее множество состояло из 39838 исполняемых файлов, из которых 8320 было чистых файлов и 31518 вирусов, червей, троянов и агентов фалов. Идея подхода состояла в том, чтобы извлекать из файлов интерпретируемые строки и использовать их в качестве прецедентов для обучения ансамбля SVM с последующей агрегацией. К примеру строка “<html> <script language = ‘javascript’> window.open(‘readme.eml’)” всегда присутствует в червях “Nimda”. Авторы работы [12] извлекли 13448 строк из всего обучающего множества, а для селекции признаков воспользовались алгоритмом Max-Relevance, который отранжировал признаки, выбрав таким образом 3000 значимых признаков. Итоговой результат с примененной моделью – 92,22% точности.

4. Краткое изложение авторского исследования

За основу в проведённых исследованиях была взята VX Heaven коллекция вирусов, в которой насчитывается ~ 230 тысячи PE вирусов. Чистые файлы собирались с различных версий операционной системы Windows (95, XP, Vista, 7), свободного проекта Cygwin и другого свободного программного обеспечения взятого с сайта download.com. Выбирались исключительно файлы формата Poratable Executable, т.е. файлы с расширением exe, dll, sys. Всего было собрано 17746 чистых файлов.

Таким образом все обучающее множество представляло 248 тысяч исполняемых файлов.

Целью работы было:

- Исследовать существующие подходы статического анализа обнаружения вирусов с применением машинного обучения;
- Провести собственные независимые эксперименты;
- Разработать собственный классификатор, способный агрегировать различные признаковые описание одних и тех же файлов;

- Сравнить результаты приведенных исследований с собственными наработками.

Заметим, что у каждого из ранее описанных подходов есть свои принципиальные недостатки:

- У подхода с выделением строк недостаток в том, что на разные классы вирусов приходятся разные строковые описатели, а в целом необходимо формировать общее представление о всех рассматриваемых объектов. По этой причине он дает, худший результат, чем остальные подходы, но лучше может классифицировать конкретный тип зловредных программ;
- У подхода с N-граммами байт, хотя он дает хорошие результаты, очень слабая интерпретация с точки зрения исследователя. Например, трудно заключить, почему последовательность 0000 0000 0000 0001 более значима чем 0000 0000 0000 0010 и совершенно неочевидно, какой будет точность при увеличении исходного множества исполняемых файлов;
- У подхода с N-граммами опкодов хорошая интерпретация (за счет того что есть потенциально опасные последовательности опкодов которые известны), но не самый лучший результат по точности и дизассемблированию кода;
- У подхода со структурными признаками, хорошие результаты. Однако выделенную структурную информацию можно подделать. Если разработчик вируса пользуется стандартными средствами разработки (к примеру, Visual Studio), то структурная информация вирусного файла не сильно отличается от чистого, так как по умолчанию используются стандартные средства компиляции и линкера.

Из приведённого следует, что единственно верного решения нет, и имеет смысл рассматривать комбинации подходов. В предлагаемом подходе рассматривается комбинации N-грамм опкодов и структурных признаков PE файла.

Следует отметить, что весь статический анализ очень чувствителен к запакованным исполняемым файлам и применению обфускации на бинарном уровне. Запакованный исполняемый файл – это исполняемый файл внутри которого находится подпрограмма запаковывающая его и делающая недоступным правильное чтение его структурных элементов. Также бессмысленно дизассемблировать запакованный файл, потому что сначала будет дизассемблироваться сам пакер, а это не то, что требуется. Применение обфускации на бинарном уровне портит дизассемблирование файла, но не отражается на структурных элементах. Поэтому в исследовании рассматриваются только незапакованные исполняемые файлы. Для этих целей была специально разработана программа, которая и распаковывает все файлы.

Условно исследование можно разделить на 3 части:

1. Исследование структурного подхода и построение классификаторов для него.
2. Исследование N-грамм для опкодов и построение классификаторов для них.
3. Агрегация классификаторов с 1 и 2 стадии в один общий стековый классификатор.

Для структурного подхода рассматривались 173 признака – это вся информация о заголовках PE: Optional, File, Dos headers , отдельно поля DataDirectory и 8 основных секций: text(code), data, idata, edata, reloc, rsrc, debug, bss и tls. Поля секций которых не было в исполняемом файле обнулялись. Затем, выбрасывая из рассмотрения признаки с малым количеством уникальных значений (<10) , строилось множество моделей более менее устойчивых к передозировке признаками. Признаки, у которых был большой разброс в значениях, факторизовались, т.е. происходила перенумерация с присвоением. Лучшие результаты в 99,6% и 99,2% точности показали модели RF и GBM, соответственно. Также проводились исследования связанные с уменьшением размерности, в частности с использованием PCA

анализа. После PCA осталось 17 признаков и обучение проводилось гораздо быстрее, но результат был хуже - 98,93% и 98,87% соответственно на тех же моделях.

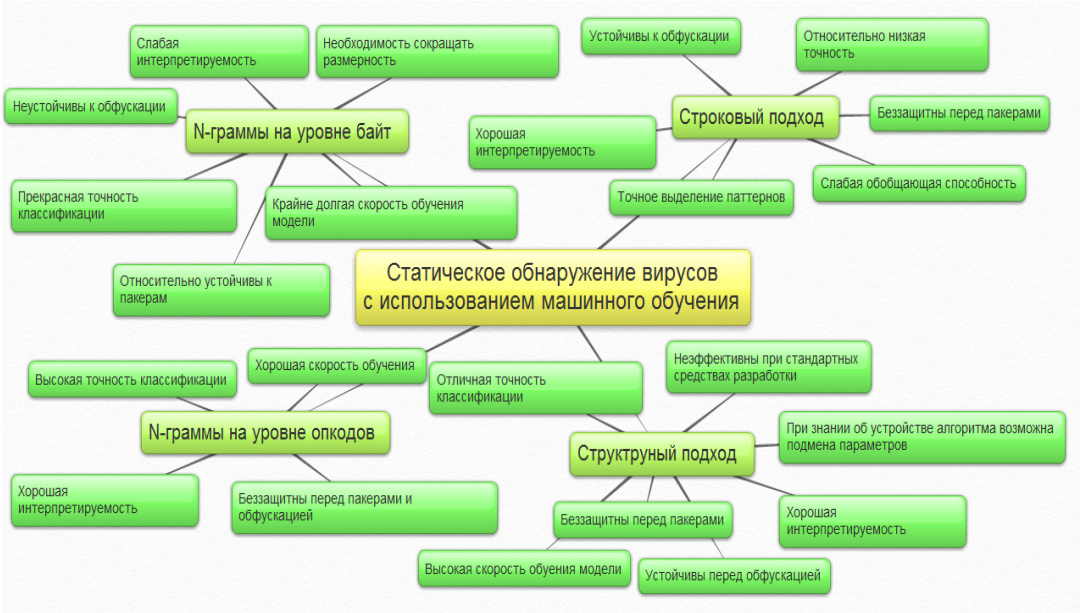

Рисунок 4. “Карта ума” подходов машинного обучения для статического обнаружения вирусов

Для N-граммного подхода на опкодах, все файлы дизассемблировались и брались 1-, 2-, 3-, 4-граммы (больше брать бессмысленно из-за затрат на скорость обучения и ухудшения общей точности) опкодов и для каждой граммы независимо строились модели.

На первой стадии фильтрации проводился корреляционный анализ, на выявление линейных зависимостей в данных, они устранялись и далее удалялись признаки, которые имели меньше 20-30 уникальных значений, это гарантировало отсев выбросов и нулевых или близких к ним, незначащих признаков. Все признаки шкалировались на один интервал для ускорения работы градиентных методов. Для 1-грамм уменьшение размерности не требуется, а для 2-, 3- и 4-грамм проводился PCA анализ. Полученная точность в 97,38% для RF и 96,21 для GBM очень близка к исследованиям [13].

Также для каждого из подходов анализировалась значимость признаков для конкретной модели методом случайного перемешивания.

Суть метода в следующем: обучаем какую-то модель, считаем общую эталонную точность модели на тренировочном множестве, далее в цикле по каждому признаку случайным образом перемешиваем каждый следующий признак, а другие оставляем в нормальном состоянии, смотрим на точность с перемешенным признаком и если эта точность мало отличается от эталонной, то признак для модели незначимый.

Таким образом, можно выделить признаки, главенствующие при обучении для конкретной модели, оставить их, а на всех остальных построить PCA и использовать как дополнительные признаки. Этот подход является оптимальным с точки зрения общей точности и затрат на обучение, но дает результат точности хуже чем подход по всем признакам сразу (95,40% и 94,90% для RF и GBM, соответственно).

Настройка гиперпараметров каждой модели происходила при помощи кросс-валидации с 5 шагами.

Далее строился агрегированный классификатор. На вход ему подается матрица из предсказаний других классификаторов. Суммарно было построено 20 классификаторов (по 10 на каждый подход) и поэтому столбцов у матрицы предсказаний было 20.

Суть агрегированного классификатора в том, чтобы стабилизировать и сгладить предсказания разных классификаторов натренированных на разных признаковых подмножествах описывающих одну и ту же задачу. Такой прием в машинном обучении называется Stacking или StackedGeneralization [15]. Он позволяет улучшить обобщающую способность классификатора, в некоторых случаях улучшая общую точность до 5%, во многом благодаря сглаживанию и уменьшению количества ложных срабатываний. С учетом того, что данные с предсказаний классификаторов распределены близко к линейному, то чаще всего для стекового классификатора выбирается линейная модель, что и происходит в нашем случае.

В итоге стековый классификатор дал общую точность в 99,87% с 0,01% ложных срабатываний.

Заключение

Итоговый стековый классификатор, как и ожидалось, дал лучшую общую точность и меньшее число ложных срабатываний, за счет оптимального объединения построенных на предыдущих шагах классификаторов.

В планах на будущее:

- Провести исследование с целью адаптации классификатора статического обнаружения вирусов с использованием машинного обучения на случаи запакованных, обфусцированных файлов. Такая цель может быть достигнута более глубокой предфильтрацией исходных файлов, к примеру можно выделить общие сегменты (байт, опкодов) присущие чистым файлам и соответственно вирусам и рассматривать в качестве признаков распределение этих общих сегментов внутри каждого файла;
- Провести исследование с большим количеством используемых моделей и автоматической селекцией признаков. К примеру, нейронные сети, а также глубокие нейронные сети представляют огромный потенциал в автоматической селекции признаков;
- Провести исследование с целью распознавания конкретных классов вирусов.

Работа выполнена при поддержке грантов РФФИ № 14-08-01276 и № 12-08-01167-а.

Библиография

[1] http://www.av-test.org/en/statistics/malware/

[2] https://www.virustotal.com/en-gb/statistics/

[3] Nwokedi Idika, Aditya P. Mathur “A Survey of Malware Detection Techniques” 2007

[4] http://www.nytimes.com/2011/01/16/world/middleeast/16stuxnet.html?pagewanted=all

[5] http://go.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf

[6] http://media.kaspersky.com/documents/business/brfwn/ru/Advanced-persistent-threats-not-your-average-malware_Kaspersky-Endpoint-Control-white-paper-ru.pdf

[7] http://msdn.microsoft.com/en-us/library/gg463119.aspx

[8] M. G. Schultz, E. Eskin, E. Z., and S. J. Stolfo, ”Data mining methods for detection of new malicious executables,” in Proceedings of the IEEE Symp. on Security and Privacy, pp. 38-49, 2001.

[9] W. Cohen “Fast effective rule induction.” Proc. 12th International Conference on Machine Learning, pp. 115-23, San Francisco, CA: Morgan Kaufmann Publishers, 1995.

[10] J.Z. Kolter and M.A. Maloof. “Learning to detect and classify malicious executables in the wild.” The Journal of Machine Learning Research, 7:2721–2744, 2006.

[11] Usukhbayar Baldangombo, Nyamjav Jambaljav, and Shi-Jinn Horng “A Static Malware Detection System Using Data Mining Methods”, International Journal of Artificial Intelligence & Applications; Jul2013, Vol. 4 Issue 4, p 113

[12] vxheaven.org

[13] Igor Santos, Felix Brezo, Xabier Ugarte-Pedrero, Pablo G. Bringas “Opcode Sequences as Representation of Executables for Data-mining-based Unknown Malware Detection” Published in Information Sciences: an International Journal Volume 231, May, 2013 Pages 64-82

[14] Y. Ye, L. Chen, D. Wang, T. Li, Q. Jiang, and M. Zhao. “Sbmds: an interpretable string based malware detection system using svm ensemble with bagging” Journal in Computer Virology, 5 (4):283–293, 2009.

[15] Georgios Sigletos Georgios Paliouras Constantine D. Spyropoulos “Combining Information Extraction Systems Using Voting and Stacked Generalization”, Journal of Machine Learning Research 6 (2005) 1751-1782

Сведения об авторах

Тимофеев Адиль Васильевич *– главный научный сотрудник лаборатории речевых и многомодальных интерфейсов Санкт-Петербургского института информатики и автоматизации Российской академии наук, Профессор кафедры информатики математико-механического факультета Санкт-Петербургского государственного университета, доктор технических наук, профессор, Заслуженный деятель науки РФ, 199178, Россия, Санкт-Петербург, 14-я линия, д. 39, СПИИРАН, e-mail: tav@iias.spb.su*

Путин Евгений Олегович *– студент V курса математико-механического факультета Санкт-Петербургского государственного университета, 199178, Россия, Санкт-Петербург, 14-я линия, д. 29, математико-механический факультет СПбГУ, e-mail: putin.evgeny@gmail.com*

Classifier for static detection of computer viruses, based on machine learning

Evgeniy Putin, Adil Timofeev

Abstract*: Original approaches to building a system for detection of computer viruses are discussed in this work. The actuality of developing and implementing an effective classifier for both known and previously unknown viruses (zero-day) is grounded. A general approach to construction of a classifier for static virus detection using machine learning, based on analysis of recent research, is outlined in the paper. The results of new research are presented. The choice of the final aggregation model - Stack classifier - is grounded.*

Keywords*: classifier, detection, computer viruses, machine learning*