САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-Механический факультет
Кафедра Системного Программирования

Система автоматизированного массового тестирования проекта CODA

Курсовая работа студента 344 группы

Комарова Константина Михайловича

Научный руководитель:
Баклановский М.В.
старший преподаватель
кафедры системного программирования

Санкт-Петербург
2015

Оглавление

Введение....3
Постановка задачи....3
Обзор существующих решений....4
Реализация тестирующей системы....5
Обоснование выбора языка....5
Архитектура системы....5
Тестирование....7
Краткое описание метрик CODA....7
Подготовка теста....7
Этапы курсовой работы....8
Пробы....8
Финальный тест....8
Интерпретация отчетов....10
Выбор оценки эффективности системы вручную....10
MissCount > 10....10
MaxPatLen <= MidPatLen and MissCount > 10....10
Обучение нейронной сети....11
Заключение....13
Список литературы....14
Приложения....15
Приложение 1. Настройка виртуальной машины....15
Приложение 2. Диаграмма RecModels....15

Введение

Борьба с вредоносными программами – одна из самых острых проблем компьютерной безопасности. Для борьбы с ними чаще используют так называемый сигнатурный подход. Он заключается в анализе существующих угроз и поиске шаблонов атак: поиске известных файлов компьютерных вирусов, опасных сетевых пакетов, блокировке известных вредоносных воздействий на систему. Однако с появлением новейших видов вредоносного ПО сигнатурные методы противодействия все больше показывают свою неэффективность и технологическое отставание. Они позволяют противостоять лишь известным, старым угрозам. К счастью, существуют и другие методы борьбы с вредоносными программами. Например, методы, основанные на поиске аномалий в работе программ. Такой метод называется аномальным.

CODA - система противодействия вредоносным программам, основанная на поведенческом аномальном методе обнаружения [1]. Эффективность обнаружения системы оценивалась только вручную на небольшом количестве тестов, однако этого недостаточно для доказательства эффективности алгоритмов. Кроме того, без проведения массовых тестов невозможно построить автоматический классификатор процессов на вредоносные и легитимные.

Цель данной курсовой работы – оценить эффективность работы алгоритмов аномального обнаружения программ, реализованных в CODA.

Постановка задачи

Для достижения заданной цели ставились следующие задачи:

- Разработать платформу для массового запуска и тестирования вредоносных объектов
- Протестировать метрики системы на большом количестве данных
- Выбрать метод оценки эффективности

Обзор существующих решений

Тестированием средств обеспечения компьютерной безопасности сейчас занимается множество компаний. Например, **anti-malware.ru** и **av-comparatives.org**. Цель проводимых ими тестирований - предоставить пользователю объективную информацию о функциональных возможностях популярных на рынке решений, на основании которой пользователь сможет самостоятельно принимать решение об их использовании.

Компании проводят большое количество различных тестов, например:

1. Тест угроз реального времени (Real-World Threats). Он состоит в том, что тестировщик собирает несколько сотен веб-ссылок на зараженные сайты из различных источников. Как правило, на такие ссылки каждый из нас натыкается в поисковиках, получает по e-mail, ICQ или другие средства интернет коммуникации, включая социальные сети. Затем тестировщик последовательно переходит по ссылкам на опасные веб-страницы и фиксирует все изменения тестовой системы. На основании полученных данных делается вывод, успешно ли тестируемый продукт обнаружил угрозу.
2. Тест файлового сканнера антивируса. Тестировщик собирает коллекцию вредоносных программ и с помощью файлового сканера, встроенного в тестируемый антивирус, проверяет, сколько из них обнаруживаются.

Компании предоставляют только результаты тестирований, сами тестирующие утилиты и другие специальные средства не выкладываются. А значит, у нас не остается другого выхода, как реализовывать тестирующую систему самостоятельно, имея на вооружении лишь виртуальную машину.

В качестве платформы виртуализации мы будем использовать VirtualBox, потому что он, во-первых, бесплатен, во-вторых, потребляет меньшее количество ресурсов, в сравнение с аналогами, что является важным критерием, поскольку тестирующая система предполагает одновременный запуск нескольких виртуальных машин.

Реализация тестирующей системы

Обоснование выбора языка

В качестве языка для реализации тестирующей системы был выбран **Perl**, поскольку написание тестирующей платформы тесно связано с обработкой различного типа текстовых отчетов, для анализа которых и предназначен язык Perl, и работой с командной строкой. Кроме того, немаловажным фактором стало наличие большого систематизированного архива готовых решений и документации [2].

Архитектура системы

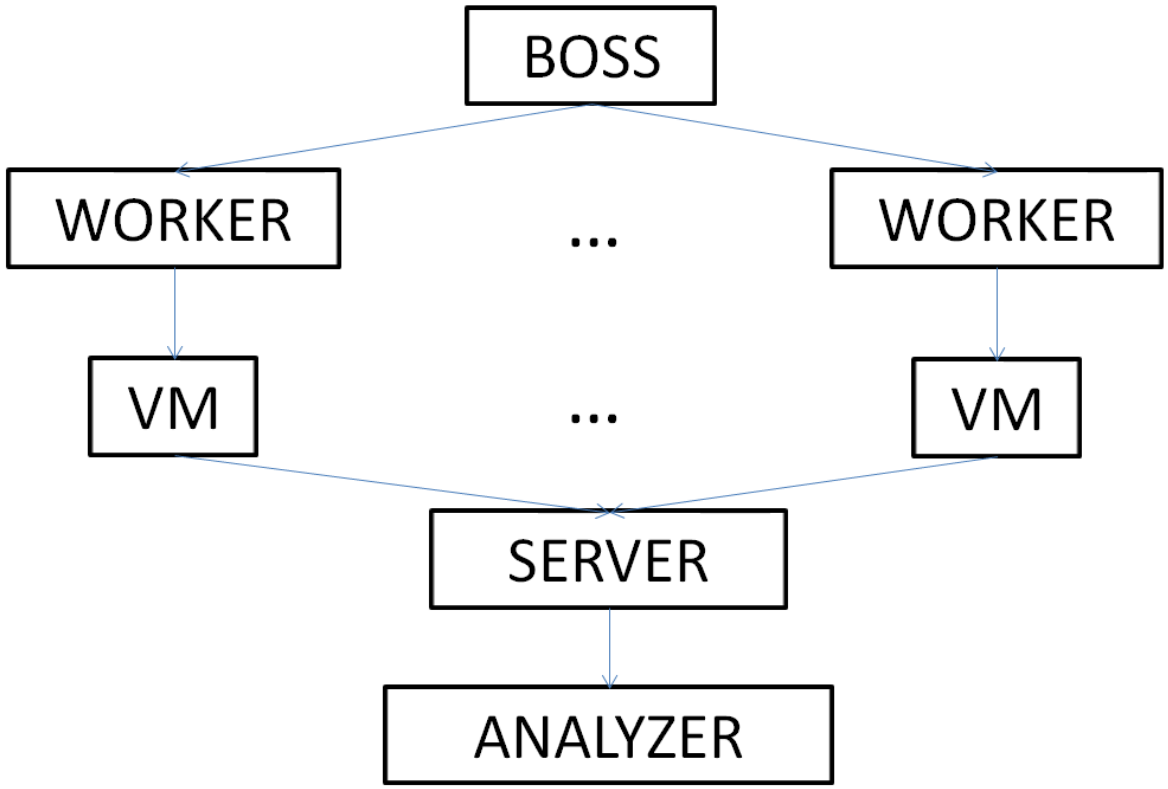

Рисунок 1. Архитектура системы

Основной модуль системы основан на принципе построения многопоточных приложений Boss-Worker, то есть выделяется один поток, который будет играть роль управляющего (BOSS). Он принимает входные данные всей программы. Эти данные BOSS передает с помощью очереди в виде специально сформированных задач одному или нескольким потокам WORKER. Каждый WORKER занимается запуском/остановкой своей виртуальной машины, получением из очереди заданий и запуском их на ней.

Для виртуальной машины создано приложение-клиент, которое занимается непосредственно запуском вредоносных программ, системы CODA, сбором полученных от CODA отчетов и отправке их на дальнейшую обработку специальному компоненту SERVER.

Компонент SERVER находится на основной машине, его задачей является сбор отчетов, одновременно со всех виртуальных машин, преобразованием их в удобный формат для дальнейшего анализа.

Компонент ANALYZER предназначен для непосредственного разбора отчетов и составления различных оценок.

Тестирование

Краткое описание метрик CODA

Для лучшего понимания следующих частей потребуются основные сведения о работе системы CODA.

CODA может работать в двух основных режимах: режим монитора и режим сигнализации.

В режиме монитора она собирает и сохраняет все системные вызовы всех процессов системы. Далее CODA строит модель системы на основе собранных данных.

В режиме сигнализации она по самому длинному потоку процесса находит соответствующую ему модель и оценивает соответствие всего процесса модели. Модель процесса – это множество шаблонов последовательностей его системных вызовов. На основе этой информации она выдает 11 различных метрик.

В данной работе будут использоваться лишь следующие:

1. MissCount – число вызовов, не вошедших ни в один шаблон
2. PatCount – число найденных в текущем потоке шаблонов из модели
3. MaxPatLen – наибольшая длина найденного в текущем потоке шаблона из модели
4. MidPatLen – средняя длина всех найденных в текущем потоке шаблонов из модели

Чем выше PatCount и MaxPatLen и ниже MissCount, тем выше дожна быть оценка схожести между рассматриваемым процессом и процессом который анализировался в режиме монитора. Говоря более простыми словами, чем ниже PatCount и MaxPatLen и выше MissCount, тем подозрительнее ведет себя процесс.

Подробное описание алгоритма можно найти в [1].

Подготовка теста

Для подготовки среды тестирования был создан набор чистых виртуальных машин под управлением VirtualBox, на которые была установлена и настроена операционная система Microsoft Windows XP Pro SP2. На каждую машину была установлена программа-клиент, управляющая процессом тестирования на данной виртуальной машине и передающая отчеты на обработку серверу.

В качестве данных для тестов использовалась онлайн база вредоносных программ [4].

Для построения модели здоровой системы CODA была запущена в режиме монитора в течение 20 минут при активном использовании системы. Было сгенерировано 6410785 вызовов, которые использовались как обучающий набор для мониторов CODA.

Далее тестирование каждой вредоносной программы проводилось в течение 1 минуты. При этом записывался лог CODA, запущенной в режиме сигнализации. Исполняемый файл вредоносной программы переименовывался в «virus.exe».

Этапы курсовой работы

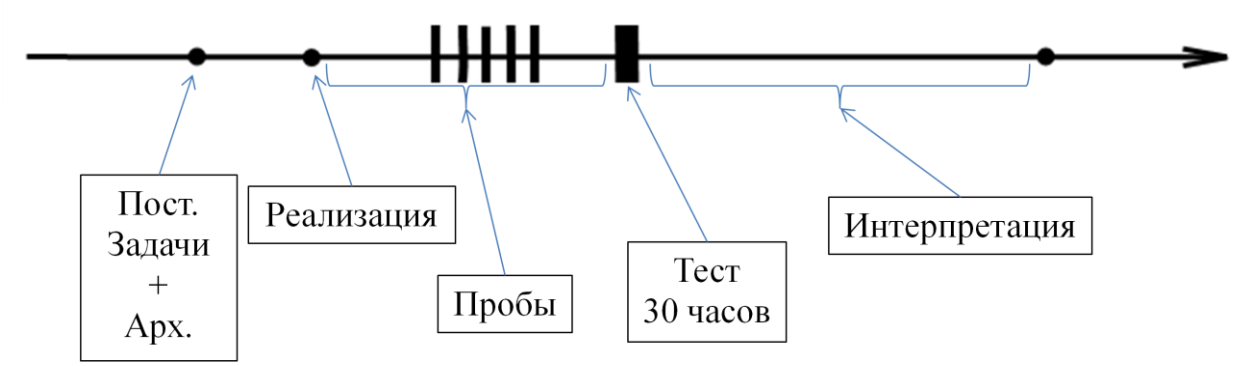

Рисунок 2. Timeline курсовой работы

Первые два этапа курсовой работы (постановка задачи, архитектура, реализация) подробно описаны выше. Однако наибольший интерес представляет дальнейшая деятельность.

Пробы

На этом этап были получены первые отчеты и первые интересные сведения о работе вредоносных программ.

Оказалось, что довольно большая часть вредоносных программ попросту не запускается на виртуальной машине. Это происходило по разным причинам: не совпадение архитектуры гостевой ОС, отсутствие необходимых файлов и пр. Имея теперь это знание, ожидаемым положительным результатом тестов стало то, что хотя бы треть вредоносных программ запустится.

Для передачи данных из виртуальной машины на сервер первоначально было решено использовать сокеты. Решение было реализовано, однако в ходе тестирования системы оказалось, что некоторые вредоносные программы разрывают подключение клиента к серверу, в результате чего сформированные отчеты попросту не приходят на сервер. Кроме того, работа сетевого адаптера становится не стабильной и большая часть последующих отчетов также не приходит на сервер. Данную проблему удалось решить, применив способ передачи данных с помощью файлов через общую папку. В результате чего сама архитектура системы даже упростилась – компонент SERVER был заменен файловой системой, а для объединения файлов в один отчет стал использоваться отдельный скрипт. Общая папка так же не осталась без внимания вредоносных программ и постоянно засорялась их копиями и другими файлами.

Этот этап был, пожалуй, самым интересным.

Финальный тест

Накопив знания, полученные на предыдущем этапе, стало возможным проведение более длительных тестов. Для этого было отобрано около 1,8 тыс. вредоносных программ. Тестирование длилось более 30 часов. В результате было получено несколько важных знаний о проведении массовых тестов.

Оказалось, что VirtualBox, работая довольно продолжительное время (около 8 час.), постоянно находясь под нагрузкой тестирующей системы, начинает вести себя

неожиданным образом: зависать, иногда портить файлы виртуальных машин. Проблемы были настолько серьезными, что приходилось останавливать процесс тестирования и принудительно выполнять перезагрузку. Используя полученное знание, в тестирующей системе был реализован механизм резервного копирования проделанной работы. Проблему с VirtualBox полностью решить так и не удалось, поэтому единственным разумным выходом из сложившейся ситуации было вмешательство в процесс тестирования и корректировка работы виртуальной среды каждые 8 часов.

В результате, из 1,8 тыс. вредоносных программ запустилось только 38%, таким образом, было получено 690 отчетов о работе вредоносного ПО, которые мы будем анализировать на следующем этапе.

Интерпретация отчетов

Выбор оценки эффективности системы вручную

Для получения оценок эффективности работы системы CODA был проведен анализ полученных отчетов. Для каждого процесса CODA высчитывает 11 метрик. Описание использованных для составления оценки метрик можно найти выше.

Первоначально была предпринята попытка составления эффективной оценки вручную. Для этого были исследованы различные комбинации метрик. Далее речь пойдет лишь о наиболее удачных попытках.

MissCount > 10

Данная оценка распознает как зараженные 685 из 690 полученных отчетов. То есть она имеет уровень срабатываний 99,27%. Однако не все так хорошо, как может показаться на первый взгляд. Уровень ложных срабатываний так же высок – 7 из 24 легитимных процессов были помечены этой оценкой.

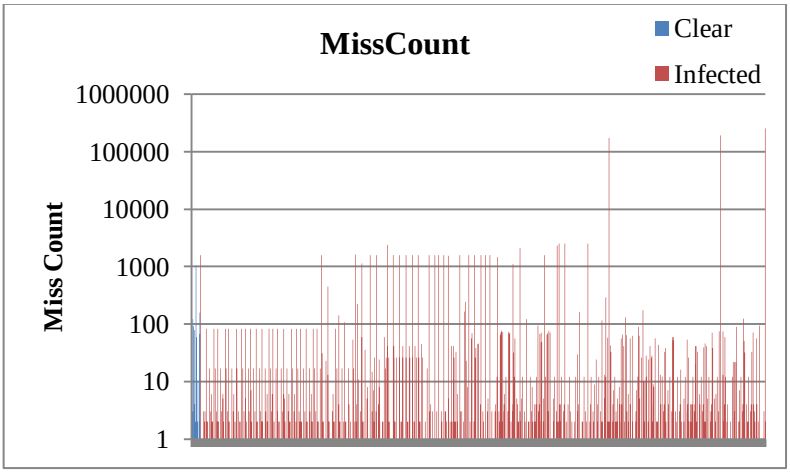

Рисунок 3. MissCount

MaxPatLen <= MidPatLen and MissCount > 10

Это самая удачная, на мой взгляд, оценка, которую удалось составить вручную. Она распознает как зараженные 568 из 690 полученных отчетов (82,31%), не имея при этом ложных срабатываний. Кроме того эта оценка в 230 отчетах (33,33%) выделяет ровно те процессы которые не встречаются на чистой системе.

Получить эту оценку помогла следующая интересная закономерность:

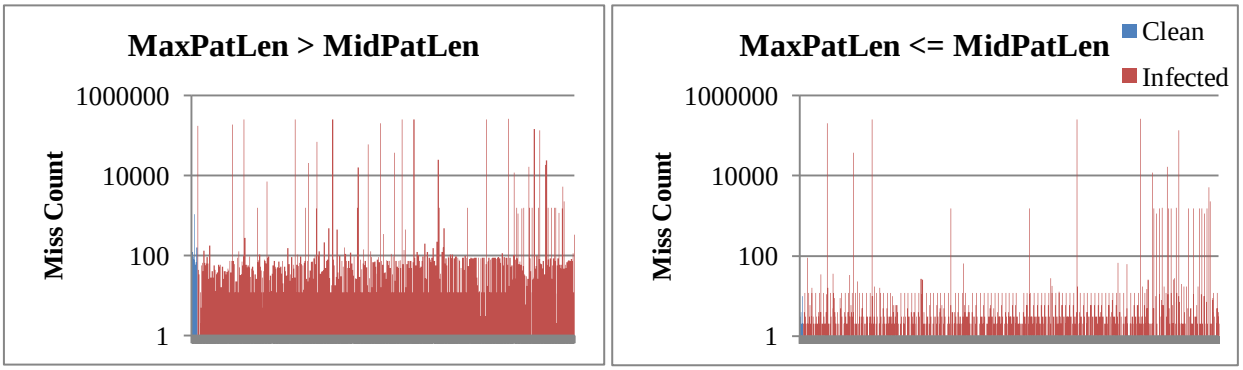

Рисунок 4. Закономерность между MissCount, MaxPatLen, MidPatLen

Оказывается, что процессам, у которых MaxPatLen > MidPatLen соответствует в основном большее значение MissCount, и наоборот. Всплески на правой диаграмме (Рис. 4) в большинстве случаев приходятся на процессы вредоносных программ.

Обучение нейронной сети

Как показал ручной анализ, данные в отчетах имеют смысл. Можно применить более серьезные методы анализа – построим автоматический классификатор на основе нейронных сетей.

Было обучено три нейросети со следующими параметрами (количество нейронов на каждом уровне):

1. 11 | 50 | 50 | 1
2. 11 | 30 | 20 | 10 | 1
3. 11 | 100 | 100 | 1

В качестве передаточной функции использовалась симметричная сигмоидальная функция активации.

На вход каждой сети подавались вектора из 11 чисел. Каждый вектор – отчет CODA о процессе.

Каждая нейросеть обучалась на:

1. 141 векторе легитимных процессов, которым давалась оценка -1
2. 139 векторах процессов с именем «virus.exe», которым давалась оценка 1

Далее выбирался пороговый параметр T и проводился набор тестов. В качестве проверочных данных использовались отчеты CODA о 57685 процессах, нам достоверно известно, что 1042 из них – отчеты вирусов (имя «virus.exe»). В ходе каждого теста для вычисления ошибки сети, рассчитывалось два значения:

1. Detection rate – отношение правильно распознанных процессов вредоносных программ к общему количеству процессов вредоносных программ
2. Hit rate – отношение правильно распознанных легитимных процессов к общему количеству легитимных процессов

Более формально:

$Detection\ Rate = \frac{TruePositive}{TruePositive+FalsePositive}$ $Hit\ Rate = \frac{TrueNegative}{TrueNegative+FalseNegative}$, где

TruePositive – количество процессов вредоносных программ распознанных как вредоносные,
FalsePositive – количество легитимных процессов распознанных как вредоносные,
FalseNegative – количество процессов вредоносных программ распознанных как легитимные,
TrueNegative – количество легитимных процессов распознанных как легитимные.

Далее перебирался порог срабатывания T, полученные результаты представлены на Рис. 5.

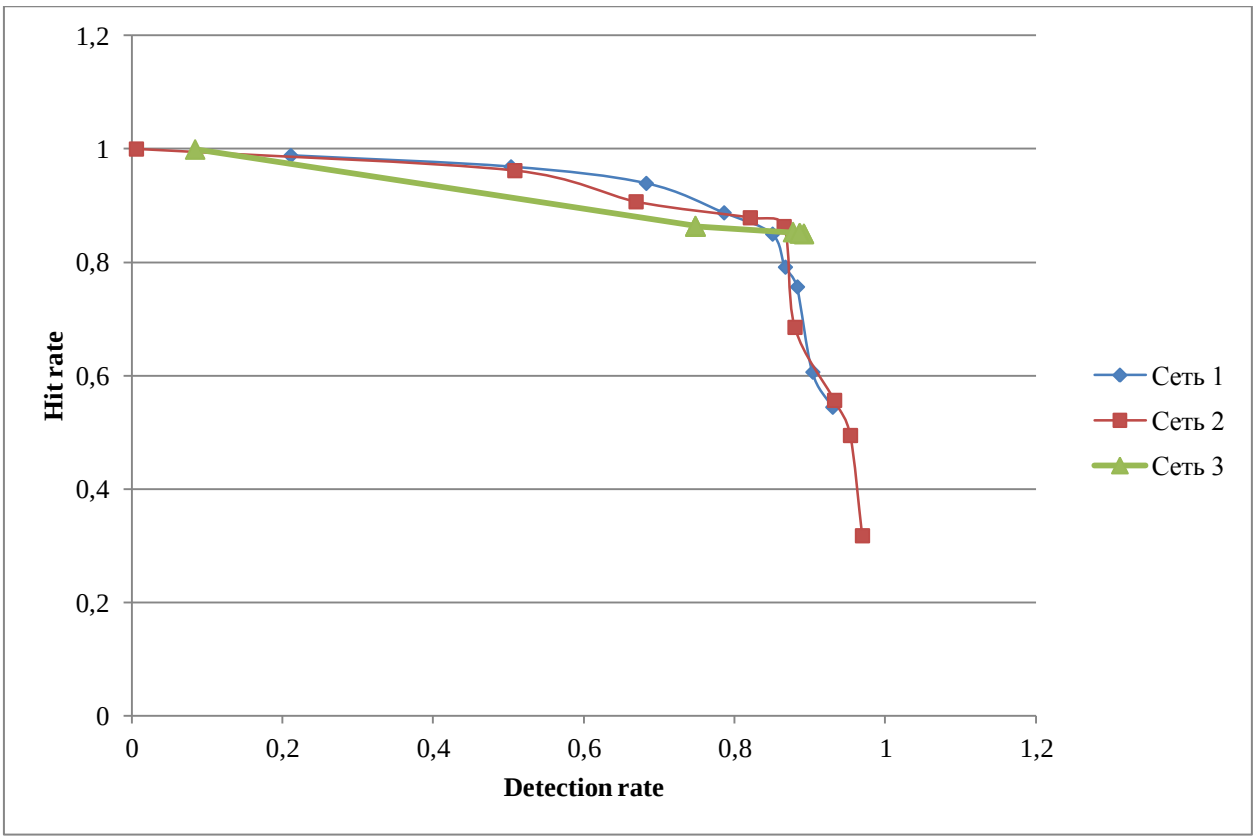

Рисунок 5. Перебор порога срабатывания

Нейросети №3 было добавлено слишком много нейронов, в результате чего она переобучилась, но, тем не менее, она частично все же подтверждает основную линию закономерности.

Как видно, ни одна из сетей не дает 100% значения Detection Rate. На первый взгляд совершенно непонятно почему это происходит. Было высказано предположение, что часть вредоносных программ, оказавшись в неестественных условиях (например, недоступен управляющий сервер) переходят в режим стагнации. Таким образом, эта самая часть вредоносных программ, «впав в спячку», ведет себя как совершенно нормальный легитимный процесс, и мы получаем *FalsePositive*. Обратившись к предыдущим этапам курсовой работы и проведя более длительные запуски тестирующей системы на некоторых экземплярах, это предположение было подтверждено.

Так же видно, что все графики имеют резкий излом на уровне Detection Rate около 85%. Дело в том, что проверочный набор данных недостаточно хорошо размечен. Вредоносная программа кроме своего основного процесса «virus.exe» зачастую создает дополнительные процессы, заражает другие, которые мы ошибочно считаем легитимными. Задача создания правильной разметки данных является нетривиальной, и в данной курсовой работе не ставится.

Заключение

В рамках данной курсовой работы были достигнуты следующие результаты:

- Реализована платформа для массового запуска и тестирования вредоносных объектов
- С помощью полученной платформы проведено тестирование системы CODA с использованием 1,8 тыс. вредоносных программ (38% запустилось) и были замерены метрики системы
- Получен метод оценки эффективности работы, который показал что точность алгоритма не менее 82,31% при 0 ложных срабатываний
- Подтверждена состоятельность метода обнаружения вредоносных программ реализованных в CODA

Помимо плановых результатов удалось выделить основные проблемы, которые предстоит решить при дальнейшем масштабировании тестирующей системы до 1 млн. объектов.

В качестве дальнейшего развития, можно различными способами улучшать реализованную тестирующую систему, масштабировать ее на более серьезные производительные мощности, тестировать с помощью нее новые алгоритмы обнаружения и получать более состоятельные результаты.

Список литературы

[1] М. В. Баклановский, А. Р. Ханов, “Поведенческая идентификация программ”, Модел. и анализ информ. систем, 21:6 (2014), 120–130

[2] Comprehensive Perl Archive Network. URL: http://www.cpan.org/ (Дата обращения 20 мая 2015)

[3] Документация к VirtualBox. URL: http://www.virtualbox.org/manual/ (Дата обращения 20 мая 2015)

[4] Онлайн база вредоносных программ. URL: https://vxheaven.org/vl.php (Дата обращения 20 мая 2015)

[5] K.-L Du and M.N.S. Swamy. Neural Networks in a Softcomputing Framework. SpringerVerlag London Limited, 2006.

Приложения

Приложение 1. Настройка виртуальной машины

Для корректного функционирования тестовой платформы необходимо специальным образом настроить каждую виртуальную машину:

- Настроить общие папки, создать снимок состояния
- Установить на гостевую ОС интерпретатор Perl
- В целях безопасности следует отключить доступ машины к интернету
- Для использования в гостевой ОС учетных записей без пароля, необходимо соответствующим образом настроить групповые политики виртуальной машины. Более подробно [3] в разделе 14. Known limitations

Приложение 2. Диаграмма RecModels

На данной диаграмме собраны значения метрики RecModels для различных процессов. С помощью данной метрики не удалось построить хороших оценок алгоритма.