**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра математического моделирования и анализа данных**

**РАСПОЗНОВАНИЕ ВРЕДОНОСТНЫХ КОДОВ В КАНАЛАХ ПЕРЕДАЧИ ИНФОРМАЦИИ**

Курсовая работа

Берковича Станислава Александровича

студента 3 курса,

специальность

«компьютерная безопасность»

Научный руководитель:

заведующий НИЛ статистического анализа и моделирования, кандидат физ.-мат. наук

Абрамович Михаил Семенович

Минск, 2015

**ОГЛАВЛЕНИЕ**

[РЕФЕРАТ 2](#_Toc419701891)

[ВВЕДЕНИЕ 3](#_Toc419701892)

[1. АНАЛИТИЧЕСКИЙ ОБЗОР ВРЕДОНОСНЫХ КОДОВ 5](#_Toc419701893)

[2. МЕТОДЫ ОБНАРУЖЕНИЯ ВРЕДОНОСНЫХ ФАЙЛОВ 7](#_Toc419701894)

[3. ПРИЗНАКИ ВРЕДОНОСНОГО КОДА 9](#_Toc419701895)

[4. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ОБНАРУЖЕНИЯ НА ОСНОВЕ ДИСКРИМИНАНТНОГО АНАЛИЗА 10](#_Toc419701896)

[5. ПОСТРОЕНИЕ ПРИЗНАКОВ КЛАССИФИКАЦИИ 12](#_Toc419701897)

[6. РЕЗУЛЬТАТЫ 13](#_Toc419701898)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 15](#_Toc419701899)

# РЕФЕРАТ

**Цель работы:**  разработать математическую модель, методы и алгоритм обнаружения вредоносных кодов в каналах передачи информации.

**Задачи:**

1. Провести аналитический обзор вредоносных кодов.
2. Разработать математическую модель распознавания вирусов на основе дискриминантного анализа.
3. Построить информативные признаки для классификации вредоносных кодов.
4. Разработать программу вычсления признаков классификации.
5. Провести численные эксперименты.

**Методы исследования:** методы статистической классификации с обучением.

# ВВЕДЕНИЕ

С начала 2000-х годов и по сегодняшний день одним из ключевых инструментов киберпреступности являются ботнеты[]. Ботнетом называется сеть зараженных узлов, на которых запущен автономный процесс, выполняющий команды злоумышленника. Узел, на котором запущен такой процесс, принято называть ботом или узлом-зомби.

В настоящей работе рассматривается проблема обнаружения и фильтрации ботнетов на этапе их распространения, эффективная для борьбы с ботнетами в виду своей превентивности. В тот момент, когда осуществляется стадия вредоносной активности ботнета, ущерб атакуемым узлам уже нанесен либо полностью, либо частично. В этом случае задача обнаружения и фильтрации ботнетов сводится к борьбе с последствиями вредоносной активности. Превентивное обнаружение позволит минимизировать наносимый ботнетом ущерб.

В данной рассматривается проблема обнаружения и фильтрации ботнетов, распространяющихся посредством удаленного эксплуатирования уязвимостей работы с памятью. Уязвимости работы с памятью возникают тогда, когда некоторый код в программе записывает в память больше данных, чем было предусмотрено разработчиком приложения. Типичными примерами таких уязвимостей являются переполнение стека, переполнение кучи, а так же некоторых других служебных структур.

Вредоносный код, эксплуатирующий уязвимости работы с памятью, традиционно называется шелл кодом (shellcode).

В данной работе рассматривается задача обнаружения и фильтрации ботнетов, распространяющихся посредством шелл кодов, на каналах передачи данных.

Возможный пример применения решения поставленной проблемы- монитор в рамках IDS/IPS. Каждый пакет, проходящий по каналу, анализируется на предмет содержания в нем шелл кодов. Если шелл код в пакете обнаружен, то пакет сбрасывается. Это, в свою очередь, накладывает определенные ограничения на решение рассматриваемой проблемы. В виду того, что высокоскоростные каналы характеризуются большим объемом передаваемой информации, проходящей по каналу в единицу времени, алгоритмическая сложность решения крайне критична. В случае высокой алгоритмической сложности возможно два сценария. При первом проходящие по каналу данные будут обработаны частично, что не дает гарантий обнаружения ботнета в обработанном объеме данных, даже при условии его передачи по каналу. Обработка же всего объема передаваемых данных влечет за собой уменьшение пропускной способности канала до значения пропускной способности средства обнаружения распространения ботнетов, либо приводит к запозданию решения о вредоносности данных. Последнее означает, что даже если в проходящем трафике были обнаружены шелл коды, то данные уже были переданы далее, а ущерб, возможно, уже нанесен.

Кроме того, для средства обнаружения и фильтрации шелл кодов крайне важен показатель ложных срабатываний. При больших объемах анализируемых данных, даже незначительный процент ложных срабатываний метода влечет за собой значительное абсолютное значение ложных сигналов, что может сказаться на качестве обслуживания легитимных пользователей.

# АНАЛИТИЧЕСКИЙ ОБЗОР ВРЕДОНОСНЫХ КОДОВ

Виды криминальной деятельности для которых используются ботнеты:

* Фишинг - кража финансовой и/или приватной информации пользователей распространенного программного обеспечения.
* Организация DDoS-атак. DDoS-атака (Distributed Denial of Service) - распределенная атака, нацеленная на исчерпание ресурсов узла жертвы
* Рассылка спама. Спам - процесс рассылки сообщений, содержащих коммерческий или иной контент большому числу лиц, не выражавшим желания их получать.
* Клиффорд - процесс перехода на ссылки рекламодателей или другие сайты лицом, не заинтересованном в посещении данных ресурсов[].

Жизненный цикл любого ботнета включает в себя несколько стадий[]:

* *Разработка*

Планирование архитектуры ботнета и его реализация, выбирается механизм распространения ботнета и конкретная уязвимость, которая будет эксплуатироваться при распространении.

* *Распространение*

Осуществляется эксплуатирование уязвимостей узлов и внедрения на них ботов.

* *Взаимодействие*

Взаимодействие между ботами и их мастером (в случае его наличия в архитектуре ботнета).

* *Вредоносная активность*

Осуществление вредоносной активности.

* *Успешное завершение атаки*

Типичная структура шелл кода:

1. *Активатор*

Участок тела шелл кода, гарантирующий корректное выполнение полезной нагрузки шелл кода

1. *Декриптор*

Часть тела шелл кода, необходимая для расшифровки зашифрованной части шелл кода, в случае ее присутствия.

1. *Полезная нагрузка*

Участок кода, выполняющий вредоносную активность

1. *Зона адресов возврата*

Зона адресов возврата содержит предполагаемый злоумышленником абсолютный адрес.

# МЕТОДЫ ОБНАРУЖЕНИЯ ВРЕДОНОСНЫХ ФАЙЛОВ

Как правило, методы обнаружения делят на две группы: обнаружение конкретных вредоносных кодов и универсальное обнаружение. Однако обнаружение конкретных вредоносных кодов на практике редко реализуем, т.к. существует огромное число модификаций одного и того же вируса. Вторая группа методов - универсальное обнаружение - позволяет обнаружить вредоносный код независимо от его типа и она более востребована на практике.

Универсальный подход очень важен ввиду его гибкости и способности быстро приспосабливаться к новым или полностью неизвестным вредоносным программам. Основным преимуществом универсальных методов обнаружения является возможность простой адаптации, основанной на переобучении классификатора на новые вредоносные коды и форматы контейнеров без необходимости разработки их явных моделей.

Универсальное обнаружение является методом мета-обнаружения, т.е. это метод, который может быть настроен (после обучения на исходных и модифицированных изображениях) на обнаружение любого вредоносного кода. Цель универсального обнаружения состоит в классификации заданных кодов в два класса: пустые (данные, не несущие вред) контейнеры и модифицированные (вредоносные коды).

Задача универсального обнаружения может быть рассмотрена как задача статистического распознавания некоторых признаков вредоносных кодов. Этот подход основан на использовании статистических методов классификации с обучением. Основная идея применения этих методов в обнаружении вредоносных кодов состоит в следующем.

На основании обучающей выборки, представляющей собой наблюдения из двух классов: класса пустых контейнеров и класса модифицированных контейнеров формируются признаки классификации, которые позволяют отличить пустой контейнер от модифицированного и строятся статистические решающие правила, которые в дальнейшем применяются для классификации вновь поступивших контейнеров в один из рассматриваемых классов.

Можно предложить общую схему классификации, которая состоит из следующих основных частей:

* Подбор пустых и модифицированных контейнеров в достаточном для анализа количестве.
* Выделение признаков.
* Выбор классификатора.
* Классификация.

# ПРИЗНАКИ ВРЕДОНОСНОГО КОДА

Некоторые признаки вредоносных кодов[]:

* *Наличие GetPC кода.*

GetPC - набор исполнимых инструкций, вычисляющих свое рас-

положение в адресном пространстве исполнимого процесса. GetPC код, как правило, необходим, чтобы возможно было подменить значение программного указателя на адрес самого кода непосредственно. Этот признак специфичен для вредоносных исполнимых инструкций, использующих техники самодекодирования и самомодификации.

* *Адрес возврата находится в определенном диапазоне значений.*

В шелл коде адрес возврата перезаписывается значением, которое находится в диапазоне адресного пространства исполнимого процесса.

* *Использование шаблонов.*

В шелл кодах часто используются зарезервированные ключевые слова или априорно известные числовые константы. Такие признаки являются специфичными для вредоносных объектов, использующих конкретные уязвимости[].

* *Если последняя инструкция в цепочке заканчивается командой перехода с прямой или абсолютной адресацией.*

Как правило, при эксплуатировании уязвимости задачей злоумышленника является не аварийное завершение запущенного процесса, а например, получения контроля над консолью с уровнем доступа ядра или какой-либо другой цели. Таким образом, код вредоносного объекта должен передавать управление системным вызовам, которые могут быть доступны явным вызовом библиотеки или путем прямого прерывания.

# МАТЕМАТИЧЕСКАЯ МОДЕЛЬ ОБНАРУЖЕНИЯ НА ОСНОВЕ ДИСКРИМИНАНТНОГО АНАЛИЗА

Рассмотрим математическую модель обнаружения на основе дискриминантного анализа[]. Пусть  – обучающая выборка, состоящая из пустых контейнеров (класс ), c распределением , - обучающая выборка, состоящая из модифицированных контейнеров (класс ), с p-мерным Гауссовским распределением , с векторами математических ожиданий и и ковариационными матрицами

, p – число признаков. Для модели линейного дискриминантного анализа предполагается, что ковариационные матрицы равны между собой: .

Линейная дискриминантная функция для j-го класса записывается в виде:

 (1)

где - априорная вероятность того, что контейнер x принадлежит классу .

Параметры дискриминантной функции  и константы  выражаются через вектора математических ожиданий  и ковариационную матрицу .

Решающее правило классификации в случае модели линейного дискриминантного анализа состоит в следующем: контейнер относится к классу , если выполняется условие; в противном случае x относится к классу .

Отметим, что эффективность построенных решающих правил дискриминантного анализа, определяется путем переклассификации обучающей выборки или путем классификации экзаменационной выборки.

Число параметров линейной дискриминантной функции определяется следующим выражением:

 (2)

Между числом параметров дискриминантной функции *l* и объемом обучающей выборки *n* должно выполняться соотношение. В противном случае вероятность ошибочной классификации при использовании дискриминантных функций существенно возрастет.

# ПОСТРОЕНИЕ ПРИЗНАКОВ КЛАССИФИКАЦИИ

Для построения признаков классификации было отобрано 52 чистых файла и 54 шеллкода. Шелкоды для Linux-систем были отобраны взяты с [1]. Все они представляли собой код на языке С. Шеллкоды были скомпилированы и дальнейшие действия производятся на исполнимых файлах для этих шеллкодов. Чистые файлы были выбраны из системных файлов в папке /bin ОС Linux Mint.

40 элементов каждого класса было отобрано для обучающей выборки. Также 12 и 14 файлов были взяты для экзаменационной выборки.

Признаком классификации является частота каждого байта в файле. Соответственно составлено 256 признаков для каждого файла.

Признаки классификации для файлов из выборки были построены при помощи программы из Приложения А.

Отбор информативных признаков проводился пошаговым методом дискриминантного анализа. Были отсеяны неинформативные признаки.

Информативными признаками были выбраны байты с индексами 5, 8, 17, т.е. байты вида 0000100, 00000111, 00010000.

# РЕЗУЛЬТАТЫ

Сравнение  средних значений частот байт проводилось с использованием t-критерия для независимых выборок. В таблице 1 представлены результаты сравнения для информативных признаков.

Таблица 1. Сравнение средних информативных признаков обучающей выборки

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Байт** | **Среднее** | **Среднее** | **t-знач.** | **ст.св.** | **p** | **N набл.** | **N набл.** |
| **5** | 0.007614 | 0.003972 | 27.5642 | 78 | 0.000000 | 40 | 40 |
| **8** | 0.004265 | 0.002035 | 12.8273 | 78 | 0.000000 | 40 | 40 |
| **17** | 0.002043 | 0.014213 | -11.9902 | 78 | 0.000000 | 40 | 40 |

По таблице 1 видно, что среднее значение информативных признаков имеют существенное различие.

Была произведена классификация файлов из обучающей и экзаменационной выборки. Результаты показали правильную классификацию для всех файлов из обеих выборок.

В результате можно сделать вывод, что данный подход, основанный на распознавании образов, позволяет безошибочно выявить  данный тип шеллкода.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Exploit database // http://www.exploit-db.com/
2. N. Daswani and M. Stoppelman. The anatomy of clickbot.a. Proc. of First Workshop on Hot Topics in Understanding Botnets
3. Rafael Rodr´ıguez-G´omez, Gabriel Maci´a-Fern´andez, and Pedro GarcaTeodoro. Analysis of botnets through life-cycle. In Javier Lopez and Pierangela Samarati, editors, SECRYPT, pages 257–262. SciTePress, 2011
4. К. В. Воронцов. Лекции по логическим алгоритмам классификации. МГУ, 2007.
5. С.А. Гайворонская. Методы обнаружения вредоносного исполнимого кода в высокоскоростных каналах передачи данных. Системы высокой доступности, 2011.
6. С.А. Гайворонская. Гибридный метод обнаружения шеллкодов, 2012

**ПРИЛОЖЕНИЕ А.** Программа подсчета признаков классификации

Данная программа написана на языке Ruby. Она компилирует вредоносные файлы (написанные на языке С ) из директории /shellcodes и помещает исполнимые файлы в директорию /shellcodes/output. Далее строит частоты распределения байт для исполнимых файлов из директории /shellcodes/output и чистых файлов из директории /simple.

require 'csv'

def compile(shellcodes)

shellcodes.each do |shellcode|

file\_name = File.basename(shellcode, '.c')

system("gcc #{shellcode} -w -o shellcodes/output/#{file\_name}")

end

Dir['shellcodes/output/\*']

end

def hist\_of\_bytes(file\_pathes)

res = Array.new(256, 0)

file\_pathes.each do |file|

s = File.open(file, 'rb') { |io| io.read }

s.each\_byte do |byte|

res[byte] += 1

end

end

sum = res.inject(:+)

res.map { |el| el.to\_f / sum }

end

`rm -rf shellcodes/output` if Dir.exist?('shellcodes/output')

shellcodes = Dir['shellcodes/\*']

simple\_files = Dir['simple/\*']

Dir.mkdir('shellcodes/output')

puts "Start compile files..."

compiled\_files = compile(shellcodes)

puts 'Start count shellcode hist...'

shellcode\_hist = hist\_of\_bytes(compiled\_files)

CSV.open('shellcode\_hist.csv', 'w') do |csv|

shellcode\_hist.each do |el|

csv << [el]

end

end

puts 'Start count simple files hist...'

simple\_hist = hist\_of\_bytes(simple\_files)

CSV.open('simple\_hist.csv', 'w') do |csv|

simple\_hist.each do |el|

csv << [el]

end

end

diff = []

shellcode\_hist.each\_with\_index do |el, idx|

diff << el - simple\_hist[idx]

end

CSV.open('diff.csv', 'w') do |csv|

diff.each do |el|

csv << [el]

end

end

puts diff.map {|el| el.abs }.max

puts diff.map {|el| el.abs }.min

puts

def count\_hist(file\_path)

hist\_of\_bytes([file\_path])

end

csv = CSV.open('table.csv', 'w')

compiled\_files.each do |shellcode|

hist = count\_hist(shellcode)

csv << [1, hist].flatten

end

simple\_files.each do |shellcode|

hist = count\_hist(shellcode)

csv << [2, hist].flatten

end