больших размеров. Например, цепочка Биткоина уже занимает размер в 100Gb, что является внушительным числом для среднестатистического ПК.

При этом нет никаких ограничений на формат хранимых данных. К примеру, Ethereum позволяет хранить в блокчейне не только транзакции, но и полноценные Тьюринг-полные программы, называемые смарт-контрактами, которые позволяют очень тонко настроить блокчейн на прикладную задачу, например, распределенный DNS-сервер.

Использованныеисточники:
1. Melanie Swan, Blockchain: Blueprint for a New Economy.
2. Jacob William, Blockchain: The Simple Guide To Everything You Need To Know.

# ИЗВЛЕЧЕНИЕ ХАРАКТЕРНЫХ ПРИЗНАКОВ В ИСПОЛНЯЕМЫХ ФАЙЛАХ ДЛЯ ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ОБНАРУЖЕНИЯ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

*Белорусский государственный университет информатики и радиоэлектроники*
*г. Минск, Республика Беларусь*

*Гороховик Я.В*

*Воронов А.А. – к.т.н., доцент*

Вредоносное программное обеспечение в общем случае представляется бинарными исполняемыми файлами, которые, как правило, регистрируются в системе, распространяются по ней, копируя себя, и оказывают на систему вредоносное воздействие. Современные антивирусные системы определяют вредоносное ПО, обладая знаниями о различных шаблонах поведения вирусов, однако определение новых, ранее не выявленных угроз, представляет определённую сложность для них. Огромное количество эвристических методов, используемых антивирусными решениями, потребляют значительное количество памяти и других ресурсов процессора. Эта нагрузка может быть преодолена путём обучения искусственных нейронных сетей, обученных на характерных признаках вредоносного ПО, содержащихся в самих исполняемых файлах. Исполняемые файлы формата PortableExecutable (PE) содержат множество полей, которые могут использоваться для предсказания поведения программы. Подход к сбору этих признаков из PE-файлов описан в данной работе.

В настоящее время одной из главных проблем информационной безопасности является выявление нового вредоносного ПО и новых угроз. Известные вирусы не представляют особой угрозы, так как могут быть легко детектированы сигнатурным анализом. Однако, выявление новых угроз требует намного более сложных эвристических подходов. Имеются следующие методы выявления таких угроз:

1. нахождение сходств между семействами вирусов. Как правило, данный метод основывается на различных алгоритмах машинного обучения, таких как Байесовская сеть или генетических алгоритмы;
2. реализация алгоритмов, эмулирующих методы принятия решений аналитика-человека – фактически, создание экспертной системы;
3. анализ файла в «песочнице». Для этого необходимо реализовать перехваты важных функций режимов пользователя и ядра. Этот метод предполагает изучения реального поведения файла в виртуальной среде[1].

Однако, каждый из перечисленных методов имеет свои ограничения:

1. первый метод может стоить значительного количества ошибок первого рода и занимать ресурсы ЭВМ, что приемлемо для тестовой среды, но не для домашнего компьютера;
2. второй метод наиболее удачен, так как предполагает изучение действительного поведения, как бы его проводил аналитик, однако, данный метод сложен и требует много памяти и ресурсов ЦП;
3. третий метод в значительной степени зависит от качества реализации соответствующего эмулятора ЦП и перехватов системных вызовов в драйверах и библиотеках. Он эффективен, однако стоит много ресурсов и денег.

В данной работе предлагается подход к извлечению характерных особенностей из исполняемых файлов для создания и обучения системы, которая принимает большое количество признаков PE-файла для определения его легитимности.

Предлагаемый метод включает сбор характерных признаков исполняемого файла следующими способами:

1. просмотр таблицы импорта для нахождения системных вызовов, наиболее часто используемых вредоносным ПО (AdjustTokenPrivileges, CreateRemoteThread, GetProcAddress, VirtualProtectEx, WriteProcessMemory и т.д.);
2. просмотр структуры PE в целом для нахождения наиболее важных полей.

Для получения особенностей таблицы импорта исполняемого файла создаётся список наиболее часто используемых вредоносным ПО функций в алфавитном порядке. Если в таблице импорта встречается такая функция, она кодируется единицей, если нет, то нулём. Полученная строка из единиц и нулей принимается за

бинарное представление входного вектора нейронной сети. Длина входного вектора зависит от количества выбранных функций.

Выбор наиболее значимых полей PE структуры осуществлялся с помощью Критерия Фишера.

Критерий Фишера является статистическим критерием и используется для сравнения дисперсий двух вариационных рядов, то есть для определения значимых различий между групповыми средними в установке дисперсионного анализа[2]. Критерий Фишера вычисляется по следующей формуле:

$$R_i = \frac{\left|\mu_{i,p} - \mu_{i,n}\right|}{\sigma_{i,p} + \sigma_{i,n}}$$

В нашем случае, Критерий Фишера используется для сравнения дисперсий характеристик положительных («легитимных») и отрицательных (вредоносных) примеров.

Данные для обучения представлены вредоносными и обычными файлами. Каждый файл производит IAT-сигнатуру – характеристику таблицы импорта и характеристики, представленные полями PE-структуры.

К примеру, при выборе 96 наиболее популярных Win32 API функций, для вредоносного файла Trojan-Spy.Win32.Zbot.dsba сигнатура таблицы импорта представляет собой следующую строку: 01000000 00000101 00000000 00001000 00010001 00000100 00000100 00000001 00000100 00000000 11000000 11001110.

Данная строка кодируется в 12 байт, которые подаются на вход нейронной сети: 0x40, 0x05, 0x00, 0x08, 0x11, 0x04, 0x04, 0x01, 0x04, 0x00, 0xC0, 0xCE.

Для проверки, какие характеристики PE-заголовка наиболее значимы, были взяты 10 вредоносных и 10 обычных исполняемых файлов.

В таблицах 1 и 2 представлены значения полей PE-заголовков для «легитимных» файлов. Таблицы 3 и 4 содержат значения полей вредоносных файлов.

Таблица 1 – Значения полей PE-структуры первых пяти «легитимных» файлов

| Feature Name | File 1 | File 2 | File 3 | File 4 | File 5 |
|---|---|---|---|---|---|
| MajorLinkerVersion | 2 | 8 | 2 | 2 | 10 |
| MinorLinkerVersion | 22 | 0 | 25 | 56 | 0 |
| SizeOfInitializedData | 472576 | 745472 | 18608128 | 107008 | 79872 |
| SizeOfUnInitializedData | 19456 | 0 | 0 | 3584 | 0 |
| MajorOSVersion | 4 | 4 | 5 | 4 | 5 |
| MinorOSVersion | 0 | 0 | 0 | 0 | 2 |
| MajorImageVersion | 1 | 1 | 0 | 1 | 0 |
| MinorImageVersion | 0 | 0 | 0 | 0 | 0 |
| Checksum | 695126 | 0 | 19630716 | 141070 | 225362 |
| DLLCharacteristics | 0 | 0 | 33088 | 320 | 33088 |

Таблица 2 – Значения полей PE-структуры следующих пяти «легитимных» файлов

| Feature Name | File 1 | File 2 | File 3 | File 4 | File 5 |
|---|---|---|---|---|---|
| MajorLinkerVersion | 10 | 10 | 10 | 8 | 8 |
| MinorLinkerVersion | 10 | 10 | 0 | 0 | 0 |
| SizeOfInitializedData | 30208 | 1536 | 10752 | 2048 | 733184 |
| SizeOfUnInitializedData | 0 | 0 | 0 | 0 | 0 |
| MajorOSVersion | 6 | 6 | 5 | 4 | 4 |
| MinorOSVersion | 2 | 2 | 2 | 0 | 0 |
| MajorImageVersion | 6 | 6 | 0 | 0 | 0 |
| MinorImageVersion | 2 | 2 | 0 | 0 | 0 |
| Checksum | 161757 | 36371 | 38736 | 49763 | 76922 |
| DLLCharacteristics | 320 | 1344 | 320 | 34112 | 320 |

Таблица 3 - Значения полей PE-структуры первых пяти вредоносных файлов

| Feature Name | File 1 | File 2 | File 3 | File 4 | File 5 |
|---|---|---|---|---|---|
| MajorLinkerVersion | 5 | 6 | 3 | 2 | 7 |
| MinorLinkerVersion | 2 | 0 | 0 | 25 | 10 |
| SizeOfInitializedData | 64000 | 121344 | 4608 | 944640 | 65536 |
| SizeOfUnInitializedData | 0 | 1024 | 0 | 0 | 0 |
| MajorOSVersion | 4 | 4 | 4 | 4 | 4 |
| MinorOSVersion | 0 | 0 | 0 | 0 | 0 |
| MajorImageVersion | 0 | 0 | 0 | 0 | 0 |
| MinorImageVersion | 0 | 0 | 0 | 0 | 0 |
| Checksum | 0 | 0 | 0 | 0 | 0 |
| DLLCharacteristics | 0 | 0 | 0 | 0 | 0 |

Таблица 4 - Значения полей PE-структуры следующих пяти вредоносных файлов

| Feature Name | File 1 | File 2 | File 3 | File 4 | File 5 |
|---|---|---|---|---|---|
| MajorLinkerVersion | 2 | 5 | 8 | 2 | 6 |
| MinorLinkerVersion | 25 | 0 | 0 | 25 | 0 |
| SizeOfInitializedData | 12288 | 95232 | 135168 | 61440 | 212992 |
| SizeOfUnInitializedData | 274432 | 0 | 0 | 651264 | 0 |
| MajorOSVersion | 1 | 4 | 4 | 4 | 4 |
| MinorOSVersion | 0 | 0 | 0 | 0 | 0 |
| MajorImageVersion | 0 | 0 | 0 | 0 | 0 |
| MinorImageVersion | 0 | 0 | 0 | 0 | 0 |
| Checksum | 0 | 0 | 716819 | 0 | 0 |
| DLLCharacteristics | 0 | 0 | 0 | 0 | 0 |

После извлечения признаков из двадцати исполняемых файлов были высчитаны среднеквадратичные и стандартные отклонения «легитимных» и вредоносных файлов. С помощью критерия фишера были вычислены ранги полей PE-заголовков исполняемых файлов.

В качестве входных данных нейронной сети были выбраны 7 полей с наибольшими рангами. Выбранные характерные признаки PE-заголовка исполняемых файлов и значения их рангов представлены в таблице 5.

Таблица 5 – Результаты применения Критерия Фишера (l – «легитимные файлы», m – вредоносные файлы)

| Feature Name | Mean(l) | Mean(m) | Standard(l) | Standard(m) | Rank |
|---|---|---|---|---|---|
| **MajorLinkerVersion** | **7** | **4.6** | **3.376** | **2.107** | **0.438** |
| MinorLinkerVersion | 12.3 | 8.7 | 17.123 | 11.055 | 0.128 |
| SizeOfInitializedData | 2078078.4 | 171724.8 | 2216994.091 | 264048.899 | 0.329 |
| **SizeOfUnInitializedData** | **2304** | **92672** | **5816.329** | **203366.116** | **0.432** |
| **MajorOSVersion** | **4.7** | **3.7** | **0.781** | **0.9** | **0.595** |
| **MinorOSVersion** | **0.8** | **0** | **0.978** | **0** | **0.816** |
| **MajorImageVersion** | **1.5** | **0** | **2.291** | **0** | **0.655** |
| **MinorImageVersion** | **0.4** | **0** | **0.8** | **0** | **0.500** |
| Checksum | 2174862.3 | 71681.9 | 5824453.419 | 215045.7 | 0.348 |
| **DLLCharacteristics** | **10291.2** | **0** | **15153.838** | **0** | **0.678** |

Представленные признаки могут быть поданы на вход нейронной сети для обучения.

В данной работе были представлены два алгоритма выбора характерных признаков исполняемых файлов. Первый алгоритм используется для извлечения из исполняемого файла сигнатуры таблиц импорта. Второй алгоритм использует Критерий Фишера для нахождения наиболее важных полей PE-заголовка исполняемого файла.

Список использованных источников:

1. Koret, J. The Antivirus Hacker's Handbook / J. Koret, E. Bachaalany – John Wiley & Sons, Inc., Indianapolis, Indiana. – 2015. – P. 165-175.

2. Stopel D. Improving Worm Detection with Artificial Neural Networks through Feature Selection and Temporal Analysis Techniques / D. Stopel, Z. Boger, R. Moskovitch, Y. Shahar, Y. Elovici // Deutsche Telekom Laboratories at Ben-Gurion University, Be'er Sheva, Israel. – 2006.

# МЕТОД МНОГОСТРУЙНОГО МОДЕЛИРОВАНИЯ В 3D ПЕЧАТИ

*Белорусский государственный университет информатики и радиоэлектроники*
*г. Минск, Республика Беларусь*

*Дубовский А.Л.*

*Селезнёв И. Л. – к.т.н., доцент*

Технологии объемной печати перестали быть только средством прототипирования и перешли в область промышленного производства. Трёхмерная печать, которая является процессом создания трёхмерных объектов практически любой геометрической формы на основе цифровой модели, быстро развивается в последние годы. Современные аддитивные технологии позволяют реализовать ресурсосберегающий, инновационный подход к проектированию и изготовлению деталей по сравнению с традиционными методами, используя различные материалы, такие как: акрил, нейлон, бетон, гидрогель, бумага, гипс, деревянное волокно, лед, металлический порошок.

Аддитивные технологии (AF — AdditiveManufacturing) — это технологии послойного синтеза, обеспечивающими практически безотходное материало- и энергоэффективное производство многих видов изделий из металлических, полимерных и композитных материалов. С момента появления в середине 80-ых годов стереолитографии и технологии послойного наплавления, техника аддитивного производства непрерывно совершенствовалась [1]. По данным WohlersAssociates, современный мировой рынок аддитивных