DOI 10.37882/2223-2966.2023.05.26

МЕТОД АНАЛИЗА ПРОГРАММ НА НАЛИЧИЕ ВРЕДОНОСНОГО КОДА С ПРИМЕНЕНИЕМ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ И ПРАВИЛ YARA

A METHOD OF ANALYZING PROGRAMS FOR THE PRESENCE OF MALICIOUS CODE USING MACHINE LEARNING METHODS AND YARA RULES

N. Ponomarev
E. Tarov

Summary: This article discusses a new method of analyzing programs for the presence of malicious code using machine learning algorithms and the YARA tool. To do this, it is proposed to form YARA rules based on hex pairs of the code of the programs under consideration and to train the model on these data. The article also discusses two important parameters that need to be set to form the YARA rule: the number of hex pairs taken and the number of logical constructions in the rule. This method can be used to effectively detect malware and may be useful for information security professionals.

Keywords: information security, YARA, malware, machine learning, bayesian algorithm..

Пономарев Николай Александрович
Санкт-Петербургский государственный университет телекоммуникаций им. профессора М.А. Бонч-Бруевича
nikapon13@gmail.com
Таров Евгений Викторович
Санкт-Петербургский государственный университет телекоммуникаций им. профессора М.А. Бонч-Бруевича
tarov25@mail.ru

Аннотация: В данной статье рассматривается новый метод анализа программ на наличие вредоносного кода с использованием алгоритмов машинного обучения и инструмента YARA. Для этого предлагается формирование правил YARA на основе hex-пар кода рассматриваемых программ и проведения обучения модели на этих данных. В статье также обсуждаются два важных параметра, которые необходимо задать для формирования правила YARA: количество взятых hex-пар и количество логических конструкций в правиле. Этот метод может быть использован для эффективного обнаружения вредоносного ПО и может быть полезен для специалистов в области информационной безопасности.

Ключевые слова: информационная безопасность, YARA, вредоносные программы, машинное обучение, байесовский алгоритм.

Введение

На сегодняшний день отчетливо видна тенденция в повышении значимости вопроса обеспечения информационной безопасности (далее — ИБ). Данная тенденция обусловлена непосредственной важностью информации и соответственно ее защиты для современного общества. Одной из составляющих частей ИБ является анализ различных программ на выявление в них компонентов кода, способных нанести какое-либо деструктивное воздействие. Данная часть ИБ имеет большое значения для защиты информации из-за быстрого развития вредоносного программного обеспечения (далее — ПО) и увеличения количества кибератак с использованием вредоносных программ [1–7].

Однако несмотря на быстрые темпы развития вредоносного ПО, существует возможность классификации данных программ на основании совокупности деструктивных воздействий, которые они способны нанести, и принципу их распространения. Данная классификация позволяет выделить следующие основные семейства вредоносного ПО [8, 9]:

— троянские программы (например, Cryptolocker, TROJAN-SPY:W32/ZBOT);
— вирусы (например, Virus.Win32.Virut.ce);
— черви (например, Net-Worm.Win32.Kido.ih, Email-Worm.Win32.Runouce.b).

Для исследования программ на наличие в них деструктивных частей кода существуют специальные методы — статический и динамический анализ [9]. Где основная отличительная черта данных методов заключается в том, что статический метод опирается на анализ непосредственно содержимого самого кода с целью выявления логики работы (алгоритма) программы и возможности прогнозирования возможных деструктивных воздействия исследуемого ПО. В свою же очередь динамический метод основывается на анализе программы в режиме реального времени, то есть во время выполнения данного кода, и по тому, как ведет себя исследуемое ПО, определяется, содержит ли она в себе деструктивный код или нет [9–12]. Однако данные методы анализа ПО имеют достаточно ощутимые минусы:

1. необходимо участие специального человека (аналитика), из-за чего результаты анализа ПО напрямую зависят от уровня квалификации и опыта аналитика;
2. неспособность автоматизировать процесс выявления содержания деструктивных частей кода (классифицировать программы) из-за чего тратится большое количество времени на анализ программ.

Цель работы

Для того чтобы устранить недостатки классических методов и инструментов анализа ПО, необходимо проводить исследования в данной области и разрабатывать новые, более эффективные методы. В данной статье предлагается метод анализа ПО на наличие вредоносного кода, основанный на специальных алгоритмах машинного обучения (далее — МО) с использованием достаточно популярного и универсального инструмента YARA.

Описание технологии Yara

YARA применяется в таких компаниях, как Kaspersky, Avast, Virus Total, Group-IB. Данная технология позволяет автоматизировать процесс анализа ПО при помощи уникальных правил, которые представляют собой специальные условные конструкции, включающие в себя текстовые или двоичные шаблоны. Эти шаблоны формирует аналитик на основе своих критериев по результату анализа программ. Например, можно создать правило YARA, которое позволит автоматически выявлять к какому семейству относится исследуемое ПО. Однако, формирование данных правил занимает продолжительный промежуток времени из-за анализа большого числа образцов вредоносного кода и составления специальной метрики для формирования правил. Потому для автоматизации процесса анализа образцов вредоносного ПО и формирования правил YARA необходимо воспользоваться специальными алгоритмами МО.

Описание модели МО

Существует большое множество методов, на которых основываются современные модели МО. Например, в работах [13, 14] рассматриваются такие алгоритмы, как наивный байесовский алгоритм, метод Rocchio, k-NN, SVM, дерево решений, нейронные сети, генетические алгоритмы и т.д. Данные методы обучения модели МО применяются для классификации текста на специальные группы (классы) на основании вероятностной оценки применяемого метода. Для формирования автоматизированной системы был выбран наивный байесовский метод обучения модели МО из-за простоты в реализации, быстроты работы и высокой точности данного метода [14].

Наивный байесовский алгоритм — это алгоритм, который позволяет сопоставлять входной текст к наиболее вероятной категории по определенным признакам при помощи теоремы Байеса. Суть работы данного алгоритма заключается в следующем: пусть у нас есть определенный набор категорий (C_1, C_2, C_3, …, C_n) и текст D, состоящий из множества слов (d_1, d_2, d_3, …, d_k). В таком случае по теореме Байеса можно определить вероятность того, что данный текст D принадлежит определенному классу C_i по следующей формуле:

$$P\left(\frac{C_i}{D}\right) = \frac{P(C_i) \times P\left(\frac{D}{C_i}\right)}{P(D)},$$

где $P(C_i)$ — априорная вероятность того, что текст D принадлежит категории C_i без предварительного анализа текста; $P\left(\frac{D}{C_i}\right)$ — вероятность существования текста D при условии, что он принадлежит категории C_i; $P(D)$ — коэффициент масштабирования.

Для формирования правила YARA предварительно преобразуем код рассматриваемой программы в hex формат. В таком случае в качестве примеров хорошего (первая категория) и вредоносного (вторая категория) ПО будут выступать их hex-формы. После обучения модели она будет способна сформировать правило YARA на основе наиболее вероятных hex-пар для конкретного семейства вредоносных программ, однако для этого ей нужно вручную задать два параметра:

1. число взятых hex-пар — это количество наиболее вероятных hex-пар для вредоносного ПО, на основании которых формируется правило YARA;
2. количество элементов в правиле — это число логических конструкций в правиле.

В таком случае процесс обучения модели и формирования правил YARA можно интерпретировать как процесс, представленный на рисунке.

Рис. 1. Процесс формирования правил YARA

Метрика оценки

Для возможности оценки ПО можно ввести коэффициент n, который будет представлять собой количество срабатываний правил YARA на одну hex-пару исследуе-

мого кода. Данный коэффициент будет рассчитываться по следующему правилу:

$$n = \frac{N_y}{N},$$

где N — общее число hex-пар в анализируемом коде, N_y — число срабатываний правила YARA;

Результаты

На основе предложенного алгоритма была реализована модель МО при помощи языка программирования Python версии 3.10. После чего произведено обучение данной модели на некоторых примерах вредоносного ПО — троянов. Также во время обучения вручную были проставлены значения числа hex-пар и количества элементов в правиле. В результате чего были сформированы 3 YARA-правила. Для каждого из них были рассчитаны значения коэффициента n относительно обычных кодов и вредоносных. Результативные данные можно представить в виде следующей таблицы.

Таблица 1.

Результаты измерения коэффициента *n*

Число взятых hex-пар	Кол-во элементов в правиле	*n* для обычного кода	*n* для вредоносного кода
10	5	0.94	1.00
	10	0.94	1.02
	20	0.94	1.05
	40	0.94	1.10
13	5	0.85	0.97
	10	0.85	0.98
	20	0.85	1.01
	40	0.85	1.03
16	5	0.63	0.91
	10	0.63	0.92
	20	0.63	0.94
	40	0.63	0.96

Вывод

В таблице представлены значения коэффициента *n* в зависимости от числа взятых hex-пар, количества элементов в правиле, а также от типа ПО. Как можно наблюдать из полученных результатов, значение *n* практически не меняется для случая обычного кода, но для числа hex-пар, равным 10, оно практически равно значению *n* вредоносного кода, что никак не позволяет сделать какую-либо оценку. Однако для значения hex-пар — 16 разница между значением *n* вредоносного и обычного кода очень заметна.

В таком случае можно предложить следующий алгоритм анализа исследуемого кода:

Пусть у нас имеется некоторый код, который мы хотим проанализировать и выяснить, является ли он вредоносным. Для этого необходимо:

1. перевести исследуемый код в hex-форму;
2. задать параметры модели — 16 hex-пар и 20 чисел в правиле (самый оптимальный вариант);
3. получить соответствующее YARA правило и применить его на исследуемом коде;
4. рассчитать значение коэффициента n;
5. сравнить значение данного коэффициента со значением 0.75. Если полученное из шага 4 значение будет меньше предложенного, то данный код с большой вероятностью не вредоносный, иначе же можно сказать, что данное ПО может нести деструктивный характер.

Здесь значение 0.75 получилось в результате большого числа экспериментов, которые показали, что значение коэффициента *n* для обычного ПО не превышает данного порога, в то время как для вредоносного не опускается ниже него.

Также стоит уточнить, что пункты 2 и 3 могут быть реализованы заранее при помощи любой базы данных. Для этого необходимо сформировать YARA-правила для каждого из типа вредоносного ПО. После чего поместить данные правила в базу данных. В таком случае алгоритм анализа исследуемого кода будет выглядеть следующим образом:

1. перевести исследуемый код в hex-форму;
2. взять из базы данных все правила YARA;
3. рассчитать несколько значений коэффициента *n*;
4. сравнить полученные значения со значением порога 0.75. Если хотя бы одно из них больше порога, то сделать вывод о том, что данный код является вредоносным.

Заключение

Представленный метод анализа программ на наличие вредоносного кода с применением методов машинного обучения и правил YARA показал свою эффективность в обнаружении различных семейств вредоносных программ. Данный метод основан на использовании предварительного преобразования кода программы в hex-формат и обучения модели на основе хорошего и вредоносного ПО. После обучения модель может сформировать правило YARA на основе наиболее вероятных hex-пар для конкретного семейства вредоносных программ.

Данный метод имеет преимущества в сравнении с традиционными методами анализа вредоносных программ,

так как он позволяет автоматически создавать правила YARA, что экономит время и усилия специалистов по информационной безопасности. Кроме того, использование методов машинного обучения позволяет увеличить точность обнаружения вредоносных программ.

В дальнейшем, данный метод может быть усовершенствован и расширен путем использования более сложных алгоритмов машинного обучения и учета других характеристик программного кода.

ЛИТЕРАТУРА

1. Шабловский, Я.К. Обзор технологии SOC (Security Operations Center) / Я.К. Шабловский, А.М. Гельфанд //. — 2021. — № 33. — С. 1316–1321. — EDN SSXUNS.
2. Интернет вещей (IoT): угрозы безопасности и конфиденциальности / А.М. Гельфанд, А.А. Казанцев, А.В. Красов, В.Р. Уляшева // Актуальные проблемы инфотелекоммуникаций в науке и образовании : сборник научных статей: в 4х томах, Санкт-Петербург, 24–25 февраля 2021 года / Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича. Том 1. — Санкт-Петербург: Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, 2021. — С. 215–220. — EDN TFJJHA.
3. Области применения аналитики больших данных в критических информационных инфраструктурах / А.М. Гельфанд, А.А. Казанцев, С.А. Кузнецов, Д.Н. Смирнов // Актуальные проблемы инфотелекоммуникаций в науке и образовании (АПИНО 2022): Сборник научных статей XI Международной научно-технической и научно-методической конференции. В 4-х томах, Санкт-Петербург, 15–16 февраля 2022 года / Под редакцией А.В. Шестакова, сост. В.С. Елагин, Е.А. Аникевич. Том 4. — Санкт-Петербург: Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, 2022. — С. 438–440. — EDN VCJXDG.
4. Израилов К.Е. Анализ состояния в области безопасности программного обеспечения // В сборнике: Актуальные проблемы инфотелекоммуникаций в науке и образовании. II Международная научно-техническая и научно-методическая конференция. 2013. С. 874–877. EDN: SMCVQZ
5. Цифровые технологии и проблемы информационной безопасности / Т.И. Абдуллин, В.Д. Баев, М.В. Буйневич [и др.]; Под редакцией Е.В. Стельмашонок, И.Н. Васильевой. — Санкт-Петербург: Санкт-Петербургский государственный экономический университет, 2021. — 163 с. — ISBN 978-5-7310-5243-6. — EDN NXZPBQ.
6. Манжуева, О.М. Современные проблемы информационной безопасности / О.М. Манжуева // Тенденции развития науки и образования. — 2022. — № 87–7. — С. 144–146. — DOI 10.18411/trnio-07-2022-303. — EDN PPIOCD.
7. Защита информации в компьютерных системах / М.В. Буйневич, И.Н. Васильева, Т.М. Воробьев [и др.]. — Санкт-Петербург: Санкт-Петербургский государственный экономический университет, 2017. — 163 с. — ISBN 978-5-7310-4070-9. — EDN YLGBGO.
8. Артес, Н.О. Сравнительный анализ взаимодействия разных видов вредоносных программ на систему «Windows» / Н.О. Артес, С.М. Елсаков // Южно-Уральская молодежная школа по математическому моделированию: Сборник трудов II всероссийской научно-практической конференции, Челябинск, 28–29 мая 2015 года / Под редакцией Ю.М. Ковалева. Том 2. — Челябинск: Издательский центр ЮУрГУ, 2015. — С. 11–18. — EDN UHDTFF.
9. Защита программ и данных. Часть 1. Способы анализа: учебное пособие / М.В. Буйневич, К.Е. Израилов, А.В. Красов; СПбГУТ. — СПб., 2020. — 72 с.
10. Буйневич, М.В. Сравнительный анализ подходов к поиску уязвимостей в программном коде / М.В. Буйневич, К.Е. Израилов, Д.И. Мостович // Актуальные проблемы инфотелекоммуникаций в науке и образовании : сборник научных статей V международной научно-технической и научно-методической конференции, Санкт-Петербург, 10–11 марта 2016 года. Том 1. — Санкт-Петербург: Санкт-Петербургский государственный университет телекоммуникаций им. проф. М.А. Бонч-Бруевича, 2016. — С. 256–260. — EDN WZILNF.
11. Буйневич, М.В. Основы кибербезопасности: способы защиты от анализа программ: Учебное пособие для обучающихся высших учебных заведений по укрупненной группе специальностей и направлений «Информационная безопасность» / М.В. Буйневич, К.Е. Израилов. — Санкт-Петербург: Санкт-Петербургский университет Государственной противопожарной службы Министерства Российской Федерации по делам гражданской обороны, чрезвычайным ситуациям и ликвидации последствий стихийных бедствий имени Героя Российской Федерации генерала армии Е.Н. Зиничева, 2022. — 76 с. — ISBN 978-5-907489-43-1. — EDN YFDEZH.
12. Буйневич, М.В. Основы кибербезопасности: способы анализа программ: Учебное пособие для обучающихся высших учебных заведений по укрупненной группе специальностей и направлений «Информационная безопасность» / М.В. Буйневич, К.Е. Израилов. — Санкт-Петербург: Санкт-Петербургский университет Государственной противопожарной службы Министерства Российской Федерации по делам гражданской обороны, чрезвычайным ситуациям и ликвидации последствий стихийных бедствий имени Героя Российской Федерации генерала армии Е.Н. Зиничева, 2022. — 92 с. — ISBN 978-5-907489-42-4. — EDN ELHJZG.
13. Дементьев, В.Е. Выбор алгоритмов машинного обучения для классификации текстовых документов / В.Е. Дементьев, С.Х. Киреев // Техника средств связи. — 2022. — № 2(158). — С. 22–52. — EDN XVIVUX.
14. Применение алгоритмов машинного обучения для обнаружения вредоносных программ в операционной системе Windows с помощью PE-заголовка / Д.Ч. Ле, М.Х. Фам, Ч.З. Динь, Х.Ф. До // Информационно-управляющие системы. — 2022. — № 4(119). — С. 44–57. — DOI 10.31799/1684-8853-2022-4-44-57. — EDN YFIBQJ.

Журнал «Современная наука: актуальные проблемы теории и практики»