Обнаружение аномальных событий на хосте с использованием автокодировщика

А.О. Гурина, О.Ю. Гузев, В.Л. Елисеев

***Аннотация*—Традиционные средства обнаружения вторжений хорошо справляются с обнаружением известных атак, но их недостаточно для обнаружения атак «нулевого дня». Повышения уровня компьютерной безопасности можно достичь при использовании комплекса мер: сигнатурный анализ и обнаружение аномалий. В статье предлагается метод обнаружения аномальных событий на хосте под управлением Windows. Основополагающая идея метода заключается в построении модели нормального поведения хоста с использованием нейронных сетей - автокодировщиков. Модель нормального поведения хоста применяется для анализа степени аномальности новых событий хоста. В методе используются два автокодировщика разной архитектуры для анализа событий, разделенных на две группы. Способ оцифровки событий для групп отличается. Критерием аномальности нового события является превышение порога мгновенной ошибки реконструкции (Immediate Reconstruction Error, IRE), который рассчитывается отдельно для каждого автокодировщика на этапе обучения. Эффективность метода подтверждается на примере обнаружения вредоносного использования системных утилит, а также на примере обнаружения других нетипичных и подозрительных событий Windows. В статье подробно рассмотрены: метод обнаружения аномальных событий Windows, используемые способы оцифровки событий разных типов и архитектуры нейронных сетей, критерий аномальности и критерии качества обученных нейронных сетей, а также достоинства и недостатки предлагаемого метода.**

***Ключевые слова*—автокодировщик, обнаружение аномалий, обнаружение вредоносных программ, компьютерные атаки, мгновенная ошибка реконструкции.**

I. Введение

В настоящее время компьютеры достаточно часто подвергаются различного рода атакам. При этом различие в функциональном назначении компьютеров – персональных, промышленных, серверов, встроенных и мобильных – не влияет на важность угрозы. Поскольку в большинстве случаев компьютеры подключены к коммуникационным сетям, что и обуславливает значительное число атак, будем далее называть компьютеры хостами (англ. *hosts*). Распространенные виды атак используют вредоносные программы в таких целях, как: вымогательство, нарушение работы систем, получение доступа к ресурсам, захват контроля, кража личных и корпоративных данных, паролей и сведений о банковских картах [1]. Известно также, что компрометация хотя бы одного слабозащищённого хоста зачастую позволяет получить несанкционированный доступ ко всей корпоративной сети, что может привести к катастрофическим последствиям для информационной безопасности организации.

Большинство хостовых систем обнаружения вторжений в настоящее время основаны на обнаружении только тех злонамеренных действий, которые заранее были описаны сигнатурами, хранящимися в базе данных хостовой системы. Известным преимуществом таких систем является малое количество ложных срабатываний, однако такие системы не способны обеспечить полноценную защиту для хоста в условиях постоянно появляющихся новых атак, поскольку они по определению не распознают атаки, сигнатуры для которых ещё не разработаны или не загружены в базу данных системы. Учитывая многочисленность известных атак, направленных на хост [1], а также их ежедневный прирост, написание сигнатур на каждую атаку уже давно не кажется эффективным решением проблемы безопасности хоста, при том, что трудоёмкость и ресурсозатратность этого метода достаточно велики. Таким образом, для комплексной защиты хоста представляется рациональным и актуальным интегрировать два метода обнаружения вторжений: на основе сигнатур и на основе обнаружения аномалий.

Метод обнаружения аномалий призван решить проблему обнаружения неизвестных на текущий момент атак, поскольку он основан на предположении, что любое отклонение от «нормального» поведения является потенциальным вторжением и нарушением безопасности хоста. Опубликовано значительное число работ, где метод обнаружения аномалий успешно используется для обнаружения различных атак. Например, [2, 3].

В настоящее время перспективным подходом для решения многих задач машинного обучения считаются искусственные нейронные сети. В частности, нейронная сеть в режиме автоассоциативной памяти – автокодировщик (англ. *autoencoder*) может

Статья получена 28 апреля 2020 г.

Гурина А.О. работает в ОАО «ИнфоТеКС», учится на факультете Автоматики и Вычислительной Техники НИУ «МЭИ», Москва, Россия (e-mail: asya.gurina001512@yandex.ru).
Гузев О.Ю. работает в ОАО «ИнфоТеКС», Москва, Россия (e-mail: Oleg.Guzev@infotecs.ru).
Елисеев В.Л. работает в ОАО «ИнфоТеКС» и НИУ «МЭИ», Москва, Россия (e-mail: vlad-eliseev@mail.ru).

использоваться для обнаружения новизны, то есть, аномалий относительно обучающей выборки. Преимуществами методов обнаружения вторжений на основе нейронных сетей по сравнению с сигнатурными методами являются: высокая производительность и простота их использования в реальном времени при минимуме затрат.

Для решения задачи выявления вторжений предлагается метод обнаружения аномальных событий на хосте с использованием автокодировщиков. Автокодировщики уже не раз успешно применялись для решения задач обнаружения атак [4, 6]. Далее описан способ обнаружения аномалий, состоящий из двух этапов: построения эталонной модели поведения хоста на основе некоторых типов событий из системного журнала хоста и использования обученной модели для обнаружения аномалий. Эффективность способа была подтверждена результатами тестирования прототипа нейросетевого модуля обнаружения аномалий.

Рассмотрим подробнее первый этап предлагаемого способа.

II. Построение модели нормального поведения хоста

A. Выбор событий для анализа

Выявить компьютерные атаки с помощью алгоритмов возможно только лишь в случае, если системные события, вызываемые атаками, попадают для анализа в систему обнаружения вторжений. Различные атаки могут вызывать разные типы событий, поэтому для расширения числа потенциально обнаруживаемых атак необходимо анализировать как можно больше различных типов событий. Вместе с тем увеличение количества типов рассматриваемых событий приводит к росту размерности пространства признаков, что затрудняет построение модели машинного обучения. Целесообразно выявить и обосновать небольшое множество типов событий, анализ которых позволит выявлять широкий спектр компьютерных атак на хосты.

В рамках данного исследования разрабатывается детектор аномалий для хоста с ОС Windows. Поведение хоста предлагается описывать ограниченным набором важных с точки зрения компьютерной безопасности типов событий, среди которых: запуск системных утилит (powershell.exe, cscript.exe, rundll32.exe), запуск процессов, интерактивный вход в систему, вход в систему с привилегиями, запись в файл etc/hosts, изменение настроек прокси сервера, изменение адреса сервера обновлений Windows, изменения в задачах планировщика.

Выбор данного набора событий обусловлен его значимостью с точки зрения мониторинга атак, которые могут обнаружить себя при появлении в системном журнале данных событий. Например, команды *powershell.exe* необходимо отслеживать, поскольку утилита является популярным инструментом безфайловых атак, характеризующихся отсутствием бинарных файлов с компилированным вредоносным кодом на жестком диске, что делает невозможным их обнаружение антивирусными решениями [7]. Атаки с помощью *powershell.exe* опасны тем, что могут привести к массовому заражению хостов корпоративной сети и инсталляции на них вредоносного программного обеспечения (ПО). Утилита *cscript.exe* используется для запуска сценариев из командной строки, поэтому её использование также следует контролировать. События, связанные с утилитой *rundll32.exe*, должны анализироваться, поскольку с её помощью могут запускаться некоторые команды-функции, заложенные в DLL-библиотеках. Кроме того, зачастую злонамеренные программы используют имя *rundll32.exe* для сокрытия своего присутствия в системе. Анализ событий запуска процессов, эвристический анализ функционирования запускаемых служб и сервисов позволяет обеспечить мониторинг действий ПО на защищаемых хостах.

События входа в систему необходимо контролировать для своевременного обнаружения изменений в поведении пользователя, например, когда вход выполнен в нетипичное время, или с нетипичной учётной записи, или с нетипичными правами и привилегиями. Как известно, с помощью записи в файл etc/hosts можно блокировать доступ к сайтам или перенаправлять пользователя на другие сайты, что и делают вредоносные программы в мошеннических целях [8]. Мониторинг событий данного типа позволит обнаружить аномальную активность, связанную с изменениями в файле *etc/hosts*. Аналогичная цель – перенаправление пользовательского сетевого трафика на сайты злоумышленников достигается изменением настройки прокси сервера. Анализ событий изменения адреса сервера обновлений Windows необходим во избежание ситуации изменения вредоносным программным обеспечением адреса сервера обновлений Windows и установке программного обеспечения из неизвестного и недоверенного источника. Планировщик задач осуществляет автоматическое выполнение одноразовых или повторяющихся задач по обслуживанию системы, что часто используется злонамеренными программами для запуска своих компонентов [9].

B. Построение пространства признаков

В пространство признаков включаются те признаки из содержимого тела события, которые позволяют однозначно отнести событие к классу нормальных или аномальных. На основе этой информации строится эталонная модель (модель нормы), описывающая нормальное поведение хоста. Поиск аномалий осуществляется в том же пространстве признаков на основе сравнения текущего поведения с эталонным. Поскольку модель нормы хоста будет представлена в виде нейронных сетей, важно понимать, что для успешного и быстрого обучения пространство признаков не должно содержать лишней информации. Исключение незначимых для принятия решения признаков позволит упростить архитектуру нейронной сети путем снижения размерности её слоёв, увеличить скорость и качество обучения.

В результате экспертного анализа системного

журнала Windows (событий запуска утилит *powershell.exe, cscript.exe, rundll32.exe*) в качестве признаков были выбраны командные строки запуска утилит.

Для других типов событий к наиболее важным признакам были отнесены: идентификатор события, имя компьютера, время события, имена учетных записей, список привилегий, имя задачи, имена новых и родительских процессов, тип входа.

Построение модели нормы хоста заключается в обучении автокодировщика на выбранных признаках, представленных в числовом виде.

C. Векторное представление признаков событий

Существует множество алгоритмов векторного представления текстов [10, 11], однако в силу специфики различных событий для их векторизации был применён следующий подход. Поскольку признаки, выделяемые из событий запуска утилит, не пересекаются с признаками остальных анализируемых событий, то для компактного представления событий первого и второго типа применялись два различных способа векторизации, что в дальнейшем повлекло синтез двух автокодировщиков разной архитектуры.

Для векторизации событий первого типа из тела события извлекается командная строка. Командная строка может включать в себя разнородную информацию: пути к файлам, параметры команды, доменные имена, URL, IP-адреса и т.д. Для того, чтобы преобразовать командную строку в вектор без потери информации о её составляющих, для каждой командной строки формируется числовой вектор фиксированной длины, элементы которого заполняются в соответствии с найденными в строке паттернами.

В ходе работы алгоритма векторизации исходная командная строка разбивается на подстроки или токены, среди которых осуществляется поиск известных паттернов. Каждая ячейка вектора, а в некоторых случаях две подряд идущие ячейки, соответствуют некоторому паттерну. В случае совпадения токена из векторизуемой строки с одним из паттернов (регулярным выражением), заполняется соответствующая этому паттерну одна или несколько ячеек вектора. В большинстве случаев в ячейку вектора заносится количество совпадений токенов с конкретным паттерном. Для описания некоторых токенов используются две ячейки, где первая соответствует количеству совпадений, а вторая – количеству символов токена.

В случае, если токен – путь, то в ячейку, соответствующую паттерну этого пути, заносится степень неизвестности пути относительно дерева известных путей, построенного на основе обучающих данных. Степень неизвестности пути определяется с помощью расстояния проверяемого пути от известных путей, которые использовались при нормальной работе хоста. Для известных путей степень неизвестности минимальна и равна единице. Чем больше путь отличается от известного, тем больше для него будет значение степени неизвестности. При векторизации событий обучающей выборки степень неизвестности путей всегда равна единице.

Всего было выявлено 37 паттернов токенов, которые чаще всего встречаются в командных строках запуска утилит. Поскольку для описания некоторых токенов используются две ячейки вектора, то размерность вектора составляет 41. По завершении векторизации событие запуска утилиты представляется в виде числового вектора, элементы которого несут в себе основную информацию о содержимом командной строки.

Разберём способ векторизации событий первого типа на примере события запуска утилиты cscript.exe. Пример процесса векторизации события запуска утилиты cscript.exe представлен на рис. 1.

Рис. 1. Процесс векторизации события запуска утилиты *cscript.exe*

На первом этапе алгоритма векторизации из тела события извлекается командная строка (CommandLine), затем осуществляется токенизация командной строки. На следующем этапе происходит классификация токенов, то есть каждый токен проверяется на соответствие известным паттернам (путь к утилите, путь к файлу, имя файла, команда, число, GUID и т.д.) и при совпадении заполняется ячейка вектора, соответствующая паттерну. Например, четвертой ячейке вектора соответствует паттерн «путь к утилите cscript.exe», поскольку путь стандартный, то мера неизвестности пути минимальна, и четвертая ячейка вектора заполняется «1». Так как в командной строке токены, совпавшие с паттернами - «команда», «Имя скрипта *.vbs», «Числовой параметр», «GUID», повторились единожды, то в соответствующие этим паттернам ячейки вектора заносится «1». Ячейки, соответствующие паттернам, не найденным в командной строке, остаются нулевыми. После классификации всех токенов командной строки и заполнения соответствующих ячеек вектора, исходная командная строка события представляется в векторном виде.

Для того, чтобы преобразовать событие второго типа в векторное представление, предлагается формировать числовой вектор фиксированной длины, элементы которого будут заполняться в соответствии с содержанием полей события (признаков), выбранных для анализа. Если какое-то поле в записи события не найдено, то соответствующая ячейка вектора остаётся нулевой. В противном случае ячейка заполняется определенным числовым кодом, зависящим от содержания поля.

Каждая ячейка вектора, а в некоторых случаях несколько подряд идущих ячеек, описывают содержание одного поля. Поскольку значения почти всех полей носят категориальный характер, они заменялись соответствующим числовым кодом. Например, поле, содержащее идентификатор события, может принимать одно из девяти значений (4624, 4657, 4672, 4688, 4698, 4699, 4700, 4701, 4702), каждому из которых в соответствие было поставлено число от 1 до 9.

Первая ячейка вектора содержит код идентификатора векторизуемого события.

Следующие две ячейки вектора отражают время события, в одну ячейку заносится час, в который произошло событие, а в другую - код из числового диапазона от 1 до 6, в зависимости от той части часа, когда произошло событие.

Далее, кодируются учетные записи в зависимости от формата их записи.

Имена задач планировщика, имена процессов представляют собой путь, поэтому в соответствующие ячейки вектора заносятся степени их неизвестности относительно дерева известных путей, заранее построенного на основе путей, найденных в обучающих примерах. Степень неизвестности каждого пути рассчитывалась аналогично способу, использованному при векторизации событий первого типа.

Разберём способ векторизации событий второго типа на примере события создания процесса. Пример процесса векторизации события создания процесса представлен на рис. 2.

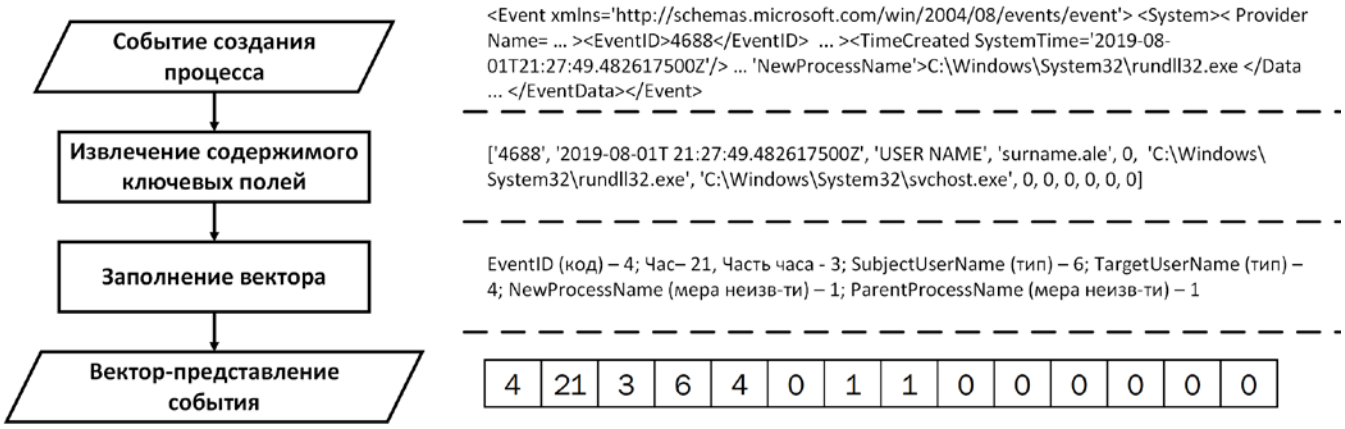

Рис. 2. Пример процесса векторизации события создания процесса

На первом этапе из тела события извлекается содержимое таких полей, как EventID, TimeCreated, SubjectUserName, TargetUserName, NewProcessName, ParentProcessName, и заносится в вектор признаков. Поскольку в данном событии найдены не все анализируемые признаки событий второго типа, то часть ячеек вектора остаются нулевыми.

Далее в каждую ячейку вектора заносится числовое значение, соответствующее определенному признаку. Например, идентификатор события создания процесса – 4688, поэтому в первую ячейку вектора заносится соответствующий этому идентификатору код - «4». Поскольку событие было создано в 21:27, то во вторую ячейку, соответствующую часу создания события, заносится «21», а в третью ячейку, соответствующую части часа, заносится «3». Четвертая и пятая ячейка заполняются кодом, соответствующим формату записи SubjectUserName и TargetUserName. В седьмую и восьмую ячейку вектора заносятся меры неизвестности путей из полей NewProcessName и ParentProcessName. Поскольку имена процессов известны, то в ячейках установлена «1».

Таким образом, по завершении процесса векторизации каждое событие второго типа преобразуется в числовой вектор, состоящий из 14 элементов.

D. Подготовка обучающих данных

Применение нейронных сетей требует наличия эталона – обучающих примеров, на которых нейронная сеть научится распознавать нормальное поведение хоста. Формирование массива обучающих примеров – важная задача, от которой во многом зависит успех обучения и дальнейшего применения нейронной сети. Обучающие примеры должны описывать максимально разнообразную активность хоста, чтобы обученная нейронная сеть могла представлять собой адекватную реальности модель поведения хоста. Чтобы иметь представление о том, какие события обычно происходят на хосте, как часто происходят те или иные события, какие параметры событий меняются, а какие нет, необходим мониторинг и сбор событий на хосте в течение длительного времени. Было принято решение собирать события для обучения как минимум в течение месяца. В этот период хост должен быть защищен от атак, чтобы обучающая выборка не содержала аномалий.

Совокупность числовых векторов, полученных после векторизации событий запуска утилит, будут использоваться для составления обучающих выборок, обучения нейронных сетей и построения модели нормы хоста с позиций событий двух типов.

В качестве обучающих данных формируется два массива обучающих векторов, описывающих нормальные события первого и второго типа.

После сбора данных и векторизации получим два массива векторов для событий первого и второго типа соответственно, которые далее будут использованы для обучения нейронных сетей.

E. Построение модели поведения хоста

В рамках данного исследования модель поведения хоста описывается с помощью двух автокодировщиков: один автокодировщик описывает поведение хоста с позиций событий запуска системных утилит, второй – с позиций событий входа, создания процессов и прочих.

Автокодировщик – это нейронная сеть специальной архитектуры, позволяющая применять обучение без учителя. Особенность архитектуры заключается в том, что выходной слой автокодировщика должен содержать столько же нейронов, сколько и входной слой. При этом в общем случае промежуточные слои должны содержать меньшее количество нейронов. Это – необходимое условие, поскольку автокодировщик обучается воспроизводить на выходе данные, наиболее близкие к входным. Особенности архитектуры автокодировщика позволяют искать обобщения и корреляцию в поступающих на вход данных, а также выполнять их сжатие в промежуточном слое.

Условно архитектура автокодировщика состоит из кодера и декодера. Кодер снижает размерность входного вектора для представления его в скрытом пространстве. При этом остаются лишь основные закономерности вектора, позволяющие декодеру из закодированного представления восстановить исходный вектор на выходе. Восстановление данных называется реконструкцией.

В ходе исследования эмпирическим путём были

выбраны оптимальные с точки зрения качества обучения архитектуры автокодировщиков (AE_41 и AE_14), которые представлены на рис. 3. Размерность входного и выходного слоя каждого автокодировщика соответствует длине обрабатываемого вектора.

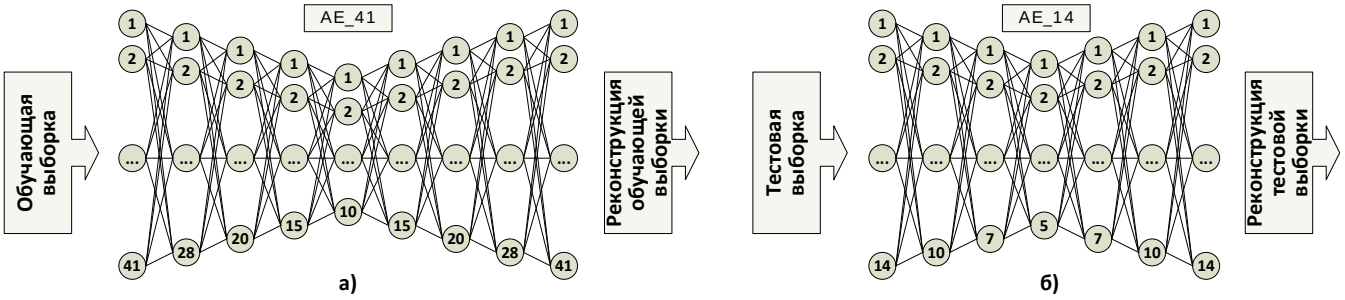

Рис. 3. Архитектуры автокодировщиков для событий первого (а) и второго типа (б)

Во входном и выходном слое автокодировщика для событий первого типа (рис. 3а) содержится 41 нейрон, в скрытом слое наименьшей размерности, где происходит сжатие или кодирование вектора, - 10 нейронов. Из сжатого представления автокодировщик восстанавливает на выходном слое реконструкцию вектора.

Архитектура автокодировщика для событий второго типа проще: во входном и выходном слое 14 нейронов, в скрытом слое наименьшей размерности – 5 нейронов.

Архитектуры автокодировщиков оставались неизменными для всех исследуемых хостов. Каждый автокодировщик обучался с помощью алгоритма адаптивной оценки моментов (ADAM) максимально точно воспроизводить вектора обучающей выборки. После того, как автокодировщик настроен воспроизводить вектора обучающей выборки с приемлемой точностью, которая определяется величиной среднеквадратичной ошибки (Mean Squared Error, MSE), его можно использовать для распознавания других векторов. Точность, с которой автокодировщик опознает входной вектор как известный, определяется величиной мгновенной ошибки реконструкции (Immediate Reconstruction Error, IRE). Мгновенная ошибка реконструкции входного вектора $X(x_1, x_2,\ldots,x_i,\ldots x_n)$ длины n рассчитывается по формуле (1), где $Y(y_1, y_2,\ldots,y_i,\ldots y_n)$ – вектор, восстановленный на выходе автокодировщика.

$$IRE_X = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

Малость величины мгновенной ошибки реконструкции означает близость тестируемого вектора к примерам обучающей выборки, а, значит, с большой долей вероятности можно утверждать, что данный вектор характеризует штатное и типичное для данного хоста поведение.

Завершающим этапом построения модели нормального поведения хоста является выбор порогового значения ошибки реконструкции (IRE_{th}) для каждого автокодировщика. IRE_{th} – это границы модели нормального поведения хоста. В общем случае пороговое значение выбирается как максимальное значение среди ошибок реконструкции для примеров из обучающего набора. Но поскольку нельзя полностью исключить возможность совершения атаки на хост во время сбора системных событий для обучения, необходимо предусмотреть способы борьбы с выбросами, которые могут попасть в обучающее множество примеров. В противном случае это приведёт к тому, что нейронная сеть, обучившись на данных с выбросами, в дальнейшем будет распознавать признаки атаки, как нормальное поведение.

В рамках данной работы поиск таких примеров осуществлялся за счет анализа значений ошибки реконструкции для примеров обучающей выборки ещё до этапа выбора порогового значения. Поскольку величина ошибки реконструкции отражает степень неизвестности примера для нейронной сети, то наличие значения ошибки, сильно превышающего основную массу значений, говорит о том, что соответствующий этому значению пример является выбросом в обучающей выборке. В случае обнаружения выбросов среди значений ошибок реконструкции, они исключались из рассмотрения и не учитывались на этапе выбора порогового значения.

Обученные автокодировщики и соответствующие им пороговые значения ошибок реконструкции представляют собой модель нормы хоста с позиций отслеживаемых событий. Стоит отметить, что пороговые значения ошибки реконструкции будут отличаться от хоста к хосту. Блок-схема создания модели нормального поведения хоста представлена на рис. 4, как первый этап способа обнаружения аномалий.

III. Обнаружение аномалий

После построения модели нормального поведения хоста можно приступать ко второму этапу – обнаружению аномалий. Обученная модель хоста и пороговые значения ошибки реконструкции используются для обнаружения на хосте событий, сильно отклоняющихся от нормальных (аномалий). Решение об обнаружении аномалии принимается на основе критерия обнаружения аномалий.

A. Критерий обнаружения аномалий

Критерием обнаружения аномалий является ошибка реконструкции. Тестовое событие считается аномальным, когда ошибка реконструкции тестового события превышает установленное пороговое значение. Такое превышение также может свидетельствовать о расширении функциональности хоста и появлении в связи с этим новых штатных событий. О степени аномальности события по сравнению с нормой можно судить по степени превышения порога IRE_{th}.

Информацию об обнаруженной аномалии необходимо передать эксперту в виде отчета. В случае, если эксперт посчитает такое событие нормальным для данного хоста, появляется необходимость в дообучении нейронной сети с целью обеспечения её дальнейшей корректной работы без ложных срабатываний.

B. Способ обнаружения аномалий

Предлагаемый способ обнаружения аномалий состоит из двух этапов: построение модели нормального поведения хоста с позиций отслеживаемых событий и обнаружения аномалий.

Для построения модели необходим сбор анализируемых событий хоста за месяц, их векторизация и построение обучающей выборки. После обучения автокодировщика рассчитывается ошибка реконструкции для каждого вектора обучающей выборки, затем осуществляется процедура выбора порогового значения ошибки реконструкции, служащего критерием обнаружения аномалий. Совокупность обученного автокодировщика и выбранного порогового значения является моделью нормального поведения хоста с позиций анализируемых событий.

Для обнаружения аномальных событий среди новых событий хоста необходимо провести векторизацию событий, построить тестовую выборку и подать её на вход обученного автокодировщика. Затем необходимо рассчитать ошибку реконструкции каждого нового события. Превышение порогового значения ошибки реконструкции означает, что проверяемое событие аномально.

Блок-схема предлагаемого способа обнаружения аномалий в общем виде представлена на рис. 4.

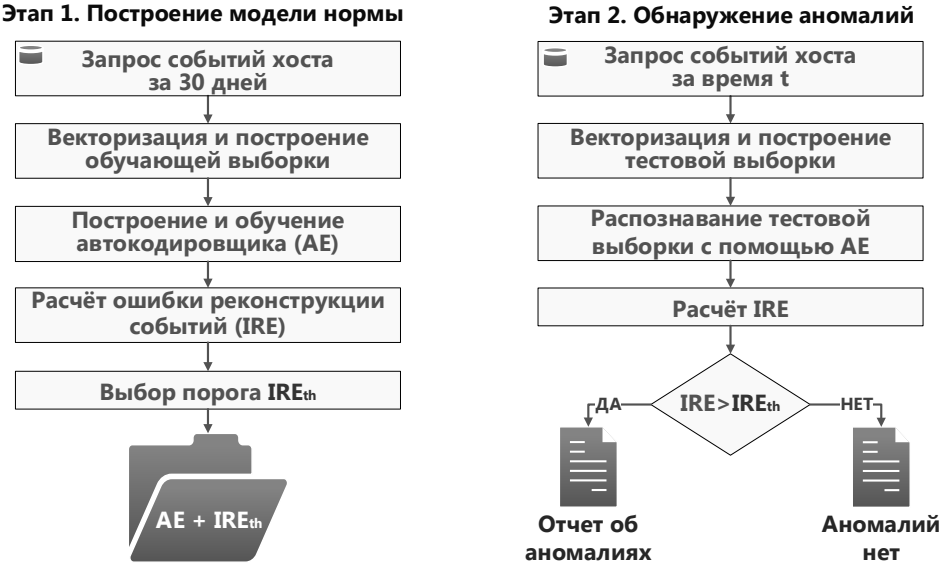

Рис. 4. Способ обнаружения аномальных событий

В рамках данной работы строятся две модели для обнаружения аномалий в событиях двух типов.

Для апробации предлагаемого способа был разработан прототип, который позволил протестировать способ на различных хостах корпоративной сети и получить практическое подтверждение его эффективности.

IV. Экспериментальные результаты

Прототип нейросетевого модуля обнаружения аномалий был реализован на языке Python 3 с применением библиотек keras, sklearn. Прототип включает в себя весь функционал обнаружения аномалий, описанный в теоретической части статьи, и является автономным. Для обнаружения аномалий на любом хосте под управлением Windows необходимо обеспечить прототипу доступ к системному журналу событий целевого хоста, в который собираются события нужных типов. При этом сам прототип может быть установлен на любой другой хост сети.

Для тестирования была выбрана рабочая станция обычного пользователя. Согласно первому этапу способа для каждой машины было осуществлено:

1) сбор событий двух типов за месяц;
2) векторизация событий двух типов;
3) формирование обучающих выборок для двух

автокодировщиков;
4) обучение автокодировщиков;
5) расчет ошибки реконструкции обучающих векторов отдельно для первого и второго типа согласно формуле (1);
6) установка пороговых значений ошибок реконструкции для каждого автокодировщика.

Пороговое значение выбиралось как максимальное значение среди ошибок реконструкции векторов обучающего набора за исключением выбросов, которые отсекались при помощи специально разработанного алгоритма. Обученные автокодировщики и соответствующие им пороговые значения ошибок реконструкции представляют собой модель нормального поведения каждого хоста с позиций отслеживаемых событий. После обучения и определения пороговых значений ошибки реконструкции автокодировщики настроены на обнаружение аномалий в событиях двух типов.

Для того, чтобы протестировать эффективность прототипа с точки зрения обнаружения не только новых событий, но и реального вредоносного использования системных утилит, в тестовые выборки с событиями первого типа были добавлены 110 событий, с вредоносными командными строками. Примеры вредоносных запусков системных утилит:

1) C:\Windows\system32\windowspowershell\v1.0\powershell.exe invoke-command -scriptblock {Write-Host "My voice is my passport, verify me."}
2) C:\Windows\system32\rundll32.exejavascript:"\..\mshtml,RunHTMLApplication ";document.write();GetObject("script:#{file_url}").Exec();«
3) C:\Windows\system32\cscript.exe //E:jscript \\webdavserver\folder\payload.txt.

В тестовую выборку с событиями второго типа вручную были добавлены 10 событий, являющихся подозрительной активностью. Примеры подозрительных событий:

1) установление новой неизвестной задачи планировщика;
2) создание нового процесса с неизвестным именем;
3) использование нетипичной учетной записи в нетипичный час при входе с нетипичными привилегиями;
4) нетипичное для хоста событие.

Согласно второму этапу способа для каждой машины было осуществлено:

1) сбор событий двух типов за месяц;
2) векторизация событий двух типов;
3) формирование тестовых выборок двух типов;
4) представление тестовых выборок на вход соответствующих автокодировщиков;
5) расчет ошибки реконструкции векторов отдельно для первого и второго типа согласно формуле (1);
6) проверка критерия обнаружения аномалий.

После подачи тестовой выборки с векторами событий запуска утилит (событий первого типа) на вход обученного автокодировщика были рассчитаны значения мгновенной ошибки реконструкции для каждого входного вектора. График значений ошибок реконструкции, рассчитанных для векторов тестовой выборки, включающей аномальные вектора, представлен на рисунке 5.

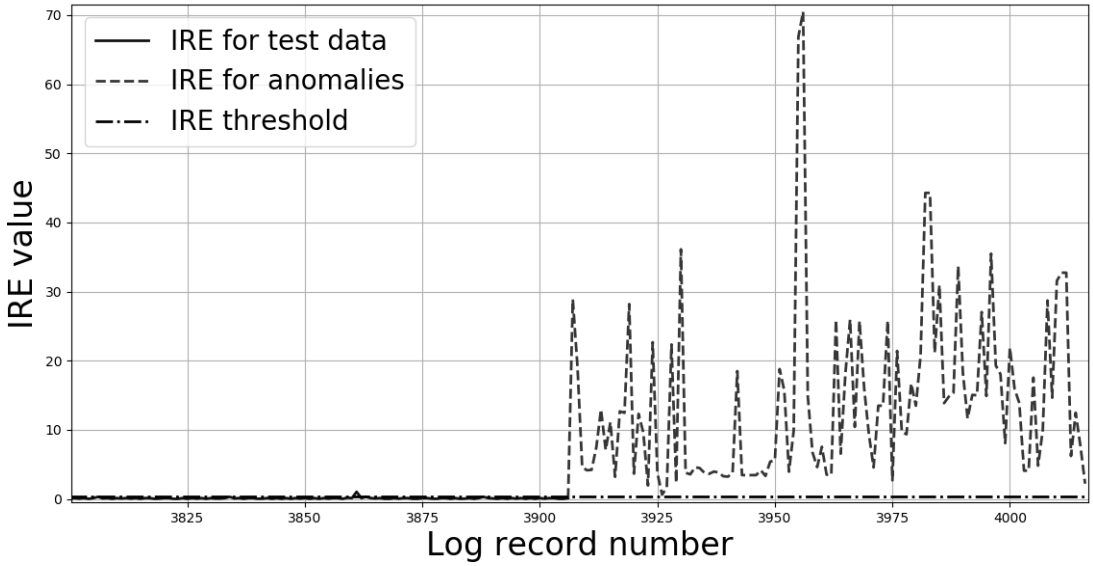

Рис. 5. График ошибки реконструкции для тестовой выборки событий первого типа

Тестовая выборка включала 3906 векторов, но для наглядности рисунка были отображены ошибки реконструкции только последних событий. Пороговое значение ошибки реконструкции отмечено на графике горизонтальной штрихпунктирной линией. Последние 110 значений тестовой выборки, отмеченные пунктирной линией, в значительной степени превышают порог, что говорит об успешном обнаружении вредоносных запусков утилит как аномалий. Кроме того, стоит отметить, что ошибка реконструкции нормальных

событий, полученных за пределами обучающей выборки, ниже порогового значения, что говорит об успешном распознавании автокодировщиком нормальных событий и вне обучающей выборки. Процент ложных срабатываний составил всего 0.07%.

Тестовая выборка с событиями второго типа также была подана на соответствующий автокодировщик. Рассчитанные значения ошибки реконструкции для тестовой выборки второго типа, включающей аномалии, а также порог ошибки реконструкции представлены на рисунке 6.

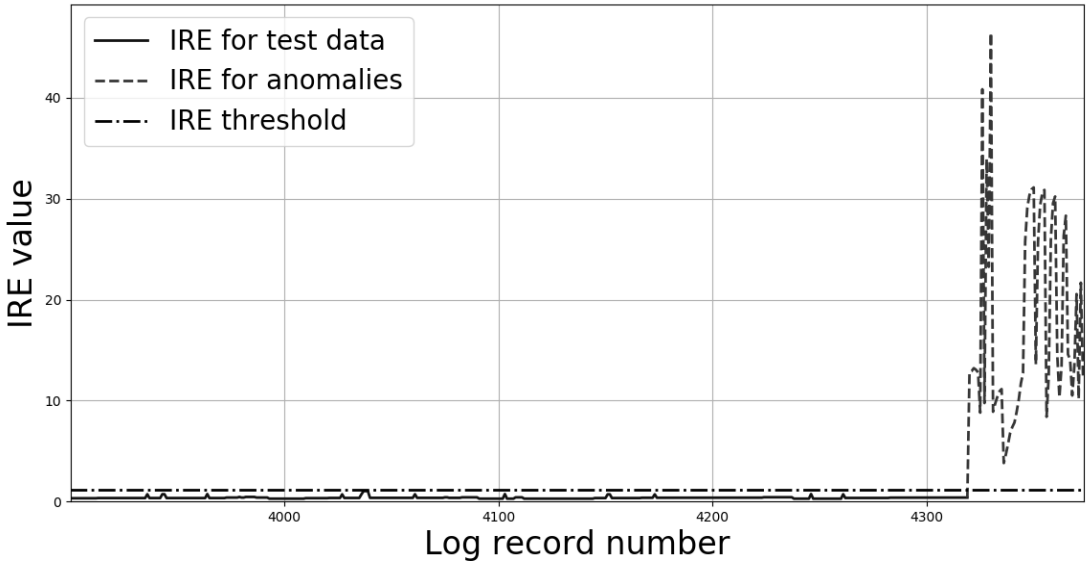

Рис. 6. График ошибки реконструкции для тестовой выборки второго типа

Все 4319 векторов тестовой выборки второго типа были верно распознаны как нормальные события, все аномальные вектора также были обнаружены.

Для оценки эффективности предлагаемого способа были рассчитаны традиционные метрики качества классификации, которые приведены в таблице 1.

Таблица 1. Метрики качества

Тип событий	Размер тестовой выборки	Метрики качества		
		Precision	Recall	F_score
1	3906	0.98	1	0.99
2	4319	1	1	1

Приведённые выше результаты позволяют сделать вывод, что прототип корректно распознает нормальные события и обнаруживает аномальные события на хосте, а предлагаемый способ работоспособен.

V. Достоинства и недостатки

Разработанный способ имеет как преимущества по сравнению с сигнатурными способами обнаружения атак, так недостатки, и ограничения.

К очевидным преимуществам предлагаемого способа относятся:

5) обнаружение новых и неизвестных угроз в реальном времени без необходимости загрузки сигнатур для них;
6) экономичность использования вычислительных ресурсов для обнаружения аномалий в реальном времени;
7) возможность управления точностью обнаружения аномалий за счет установки порогового значения ошибки реконструкции.

К недостаткам и ограничениям способа можно отнести:

1) расширение пространства анализируемых событий требует экспертной процедуры векторизации;
2) необходимость контроля безопасности целевого хоста на период сбора обучающей выборки;
3) события с различной структурой и непересекающимся набором признаков требуют использования различных архитектур нейронных сетей.

VI. Заключение

В работе предложен и детально описан способ обнаружения аномальной активности на хосте под управлением Windows на основе автокодировщиков. Результаты тестирования разработанного прототипа подтвердили успешность и эффективность предложенного способа.

Библиография

[1] Positive Technologies. Кибербезопасность 2019-2020. Тренды и прогнозы. [В Интернете] 19 Декабря 2019 г. [Цитировано: 13 Января 2020 г.] https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-2019-2020/#id3-3.

[2] Лаврентьев, Андрей. MLAD: обнаружение аномалий методами машинного обучения. Лаборатория Касперского. [В Интернете] 2018 г. [Цитировано: 13 Января 2020 г.] https://ics-cert.kaspersky.ru/reports/2018/01/16/mlad-machine-learning-for-anomaly-detection/.

[3] Гамаюнов, Денис Юрьевич. Обнаружение компьютерных атак на основе анализа поведения сетевых объектов. МГУ им. М.В. Ломоносова. 2007.

[4] Daboubi, Walid. Anomaly detection with autoencoder neural network applied on detecting malicious URLs. [В Интернете] 1 Июля 2018 г. [Цитировано: 13 Января 2020 г.] https://medium.com/@walid.daboubi/anomaly-detection-with-autoencoder-neural-network-applied-on-detecting-malicious-urls-7536abcb403f.

[5] Hieu, Mac, Dung, Truong и all, et. Detecting Attacks on Web Applications using Autoencoder. SoICT 2018. 2018 г.

[6] Gurina, Anastasia и Eliseev, Vladimir. Anomaly-Based Method for Detecting Multiple Classes of Network Attacks. 2019 г., Information, Т. 10(3):84.

[7] Panda Security. PowerShell – отличный вектор атаки для безфайловых угроз. [В Интернете] 28 Февраля 2019 г. [Цитировано: 13 Января 2020 г.] https://www.securitylab.ru/blog/company/PandaSecurityRus/345805.php.

[8] SecurityLab.ru. Владельцы NAS QNAP пожаловались на загадочный вредонос, отключающий обновление антивирусов. [В Интернете] 11 Февраля 2019 г. [Цитировано: 13 Января 2020 г.] https://www.securitylab.ru/news/497863.php.

[9] Давыдова, Анна. Хакеры начали использовать уязвимость нулевого дня в планировщике заданий Windows с помощью вредоносных программ. [В Интернете] 2 Октября 2018 г. [Цитировано: 13 Января 2020 г.] https://codeby.net/blogs/hakery-nachali-ispolzovat-uyazvimost-nulevogo-dnya-v-planirovshhike-zadanij-windows-s-pomoshhyu-vredonosnyh-programm/.

[10] Циось, А. И. Анализ тональности текстов с использованием классификаторов на основе машинного обучения. Санкт-Петербургский Политехнический Университет Петра Великого Институт Компьютерных Наук и Технологий. Санкт-Петербург : б.н., 2018. стр. 33-46, Диссертация.

[11] Нугуманова, А. Б., и др. Обогащение модели Bag-of-words семантическими связями для повышения качества классификации текстов предметной области. 2016 г., Программные продукты и системы, Т. 2(114).

Host Anomalies Detection using Autoencoders

A. Gurina, O. Guzev, V. Eliseev

***Abstract*—Traditional intrusion detection tools deal well with detecting known computer attacks but it is not enough to detect zero-day attacks. Improving computer security can be achieved by using a set of measures: signature analysis and anomaly detection. This article proposes a method for detecting abnormal Windows hosts events. The fundamental idea of the method is to build a model of normal host behavior using neural networks (autoencoders). The normal host behavior model is used to analyze the degree of abnormality of new host events. The method uses two autoencoders of different architectures to analyze events divided into two groups. The way of events digitization for groups is different. The anomaly criterion for a new event is the excess of Immediate Reconstruction Error (IRE) threshold. IRE threshold is calculated separately for each autoencoder at the training stage. The effectiveness of the method is confirmed by means of detecting malicious use of Windows system utilities and other atypical and suspicious Windows events. The article describes in detail: the algorithm for detecting abnormal Windows events, events digitization methods, neural networks architecture, the anomaly criterion and quality criteria of trained neural networks and the advantages and disadvantages of the proposed anomaly detection method.**

***Keywords*—**autoencoder, anomaly detection, malware detection, computer attack, immediate reconstruction error

REFERENCES

[1] Positive Technologies. Kiberbezopasnost' 2019-2020. Trendy i prognozy. [V Internete] 19 Dekabrja 2019 g. [Citirovano: 13 Janvarja 2020 g.] https://www.ptsecurity.com/ru-ru/research/analytics/cybersecurity-2019-2020/#id3-3.

[2] Lavrent'ev, Andrej. MLAD: obnaruzhenie anomalij metodami mashinnogo obuchenija. Laboratorija Kasperskogo. [V Internete] 2018 g. [Citirovano: 13 Janvarja 2020 g.] https://ics-cert.kaspersky.ru/reports/2018/01/16/mlad-machine-learning-for-anomaly-detection/.

[3] Gamajunov, Denis Jur'evich. Obnaruzhenie komp'juternyh atak na osnove analiza povedenija setevyh ob"ektov. MGU im. M.V. Lomonosova. 2007.

[4] Daboubi, Walid. Anomaly detection with autoencoder neural network applied on detecting malicious URLs. [V Internete] 1 Ijulja 2018 g. [Citirovano: 13 Janvarja 2020 g.] https://medium.com/@walid.daboubi/anomaly-detection-with-autoencoder-neural-network-applied-on-detecting-malicious-urls-7536abcb403f.

[5] Hieu, Mac, Dung, Truong i all, et. Detecting Attacks on Web Applications using Autoencoder. SoICT 2018. 2018 g.

[6] Gurina, Anastasia i Eliseev, Vladimir. Anomaly-Based Method for Detecting Multiple Classes of Network Attacks. 2019 g., Information, T. 10(3):84.

[7] Panda Security. PowerShell – otlichnyj vektor ataki dlja bezfajlovyh ugroz. [V Internete] 28 Fevralja 2019 g. [Citirovano: 13 Janvarja 2020 g.] https://www.securitylab.ru/blog/company/PandaSecurityRus/345805.php.

[8] SecurityLab.ru. Vladel'cy NAS QNAP pozhalovalis' na zagadochnyj vredonos, otkljuchajushhij obnovlenie antivirusov. [V Internete] 11 Fevralja 2019 g. [Citirovano: 13 Janvarja 2020 g.] https://www.securitylab.ru/news/497863.php.

[9] Davydova, Anna. Hakery nachali ispol'zovat' ujazvimost' nulevogo dnja v planirovshhike zadanij Windows s pomoshh'ju vredonosnyh programm. [V Internete] 2 Oktjabrja 2018 g. [Citirovano: 13 Janvarja 2020 g.] https://codeby.net/blogs/hakery-nachali-ispolzovat-uyazvimost-nulevogo-dnya-v-planirovshhike-zadanij-windows-s-pomoshhyu-vredonosnyh-programm/.

[10] Cios', A. I. Analiz tonal'nosti tekstov s ispol'zovaniem klassifikatorov na osnove mashinnogo obuchenija. Sankt-Peterburgskij Politehnicheskij Universitet Petra Velikogo Institut Komp'juternyh Nauk i Tehnologij. Sankt-Peterburg : b.n., 2018. str. 33-46, Dissertacija.

[11] Nugumanova, A. B., i dr. Obogashhenie modeli Bag-of-words semanticheskimi svjazjami dlja povyshenija kachestva klassifikacii tekstov predmetnoj oblasti. 2016 g., Programmnye produkty i sistemy, T. 2(114).