Поиск аномальных событий в журналах событий ОС Windows

В.С. Веденеев[1],[2] ingafen@gmail.com

И.В. Бычков[2],[3] bychkov@csu.ru

[1]ПАО «Челябинский цинковый завод», Челябинск, Россия
[2]Челябинский государственный университет, Челябинск, Россия
[3]Южно-уральский государственный университет, Челябинск, Россия

Аннотация

Поиск аномалий в журнале событий ОС Windows – это инструмент, помогающий выявлять нарушения (умышленные и неумышленные) информационной безопасности, неправильную работу программного обеспечения, ошибки в конфигурировании операционной системы. При этом перед исследователем возникает ряд задач, таких как выбор алгоритма поиска аномалий, настройка параметров такого алгоритма, проверка корректности и анализ полученных результатов. В настоящей работе описан способ решения таких задач с использованием самоорганизующихся нейронных сетей Кохонена.

Введение

Журнал событий ОС Windows (Event Log) является важным источником информации о работе пользователя за ПК, об ошибках и сбоях в программном обеспечении, о действиях системных администраторов, в том числе - скрытных для пользователя. Поиск событий, свидетельствующих об инциденте информационной безопасности (далее - ИБ), может происходить путем запросов к журналу или методами поиска аномалий. Первый способ предполагает наличие знаний о том, какие именно данные необходимо искать. Второй способ предполагает последующее изучение связанных, похожих событий для принятия решения о том, является ли событие инцидентом ИБ. Оба способа являются взаимосвязанными, так как события, признанные инцидентами ИБ и выявленные в результате поиска аномалий изучаются, и составляется запрос к журналу событий, позволяющих выявлять такие события.

1 Журнал событий ОС Windows

На момент написания работы наиболее распространенными являются две версии журнала событий. В операционных системах начиная с ОС Windows Vista и Windows Server 2008 используется третья версия журнала событий. В более ранних Windows XP и Windows Server 2003 использовалась вторая версия Event Log. Основными нововведениями в новой версии Event Log являются:

- Использование новых кодов событий.

  Новая группировка событий по кодам.

- Новый формат тела события.
- Возможность асинхронного чтения из журнала событий.

In: Sergey V. Belim, Nadezda F. Bogachenko (eds.): Proceedings of the Workshop on Data, Modeling and Security 2017 (DMS-2017), Omsk, Russia, October 2017, published at http://ceur-ws.org

Также следует отметить о возможности чтения событий старой версии Event Log из более новой версии журнала.

Рассмотрим структуры событий различных версий Event Log.

1.1 Описание структуры событий Event Log

Событие Event Log v.2 состоит из следующих полей:

- Дата и время.
- Тип журнала [1].
  - Application Log (Приложение).
  - Security Log (Безопасность).
  - System Log (Система).
  - Microsoft-Windows-Forwarding/Operational (Перенаправленные).
  - Custom.
- Тип события [2].
  - Error event (Ошибка).
  - Failure Audit event (Аудит отказа).
  - Success Audit event (Аудит успехов).
  - Information event (Информация).
  - Warning event (Предупреждение).
- Код события.
- Наименование учетной записи - инициатора события.
- Сетевое имя компьютера (NetBIOS).
- Тело события.

Тело события содержит детальное описание события, и в зависимости от кода события состоит из разных полей.

В обновленной версии системного журнала событие имеет следующие поля:

- Дата и время.
- Тип события.
- Тип журнала.
  - Такие же, как и в Event Log v.2.
  - Setup (Установка).
- Код события.
- Тело события.

2 Самоорганизующиеся сети Кохонена

Данный вид нейронных сетей функционирует по принципу «победитель получает всё». Сети Кохонена используют «обучение без учителя» (алгоритм может быть преобразован в «обучение с учителем»). Наиболее интересным свойством сетей Кохонена является самоорганизация, а именно – повторение в N-мерном пространстве расположение объектов. «Обычные» одномерные сети Кохонена используются для кластеризации данных, многомерные сети Кохонена могут использоваться для распознавания изображений.

Алгоритм обучения сети Кохонена описан в [3]. Обучение сети Кохонена содержит в себе некоторое число параметров, таких как функция скорости обучения, алгоритм инициализации весов нейронов, способы оптимизации, выбор которых существенно влияет на результат обучения. Далее опишем использованный в работе алгоритм обучения сети Кохонена.

2.1 Обучение сети

Шаг 1: Инициализация сети

Инициализация весов нейронов сети осуществлялась случайными значениями, расположенными на отрезке $[0; 1]$. Далее производится нормализация весов.

Шаг 2: Выбор нейрона-победителя

Из обучающей выборки выбирается некоторый вектор и подается на вход сети. Далее происходит выбор нейрона-победителя путем вычисления максимума из скалярных произведений между входящим вектором и всеми нейронами.

На шаге 1 обучения всем нейронам присваивается потенциал $p_i = 1/N$, где N – число нейронов в сети. После каждого круга обучения происходит корректировка потенциалов по следующей формуле:

$$p_i(k+1) = \begin{cases} p_i(k) + 1/N, i \neq j \\ p_i(k) - p_{min}, i = j \end{cases} \tag{1}$$

где i – номер нейрона, j – номер нейрона-победителя. При $p_i < p_{min}$ нейрон исключается из обучения, а именно, из шага 2 – т.е. он не участвует в выборе нейрона-победителя, и из шага 3 – его веса не корректируются.

Шаг 3: Корректировка весов

Корректировка весов нейрона-победителя на $k+1$ шаге осуществляется по следующей формуле

$$\mathbf{w}_i(k+1) := \mathbf{w}_i(k) + \eta(k)(\mathbf{x} - \mathbf{w}_i(k)), \tag{2}$$

где $\mathbf{w}_i(k+1)$ – вес i-го нейрона на следующем шаге, $\mathbf{x}$ – входной вектор, k – дискретное время, $\eta(k)$ – функция скорости обучения в момент времени k. Функция скорости обучения имела следующий вид:

$$\eta(k) = 0.2 \operatorname{arcctg}(7k/M), \tag{3}$$

где M – общее число шагов обучения. График функции скорости обучения представлен на рис. 1 (на оси абсцисс – дискретное время).

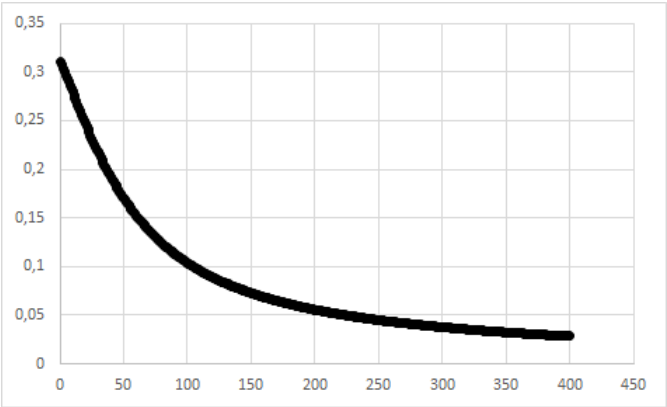

Рис. 1: График функции скорости обучения

По завершению шага 3 производится выбор следующего вектора из обучающей выборки и вновь повторяются шаги 2 и 3. После того, как все векторы из обучающей выборки были поданы на вход нейронной сети, счетчик дискретного времени k увеличивается на 1 и вновь на вход нейронной сети подаются векторы из обучающей выборки.

Обучение продолжается до тех пор, пока $k < M$.

Для ускорения процесса обучения применялось распараллеливание с использованием библиотеки CUDA, аналогичное работе [4].

2.2 Преобразование объектов из Event Log в числовой вектор

Для преобразования событий Event Log в числовой вектор применялись следующие подходы:

1. Для различных типов событий применялись отдельные сети, т.е. каждый вид логов обрабатывался по-отдельности.

2. Каждому пользователю и каждому ПК присваивался уникальный числовой идентификатор (Id), который в дальнейшем представлялся в виде двоичного поля фиксированной длины. Например, значение Id=90 при заданной длине векторов равной 10 преобразовывалось в вектор с координатами 0,0,1,0,1,1,0,1,0. Идентификаторы записывались в базу данных.

3. Временны́е значения преобразовывались в вектор длины 1 по следующей формуле:

$$(hour * 3600 + minutes * 60 + seconds)/3600. \tag{4}$$

Значения дней недели переводились в вектор длины 3.

$$\{\cdot,\cdot,\cdot\} = \begin{cases} monday = \{0;0;1\} \\ tuesday = \{0;1;0\} \\ wednesday = \{0;1;1\} \\ thursday = \{1;0;0\} \\ friday = \{1;0;1\} \\ saturday = \{1;1;0\} \\ sunday = \{1;1;1\} \end{cases} \tag{5}$$

2.3 Пример преобразования логов в числовой вектор

Далее покажем обработку логов об авторизации описываемыми алгоритмами. Для обработки события об авторизации объединялись в сессии, то есть в один объект собирались события о входе и выходе из системы. Преобразование объекта (сессии) в числовой вектор производилось следующим образом:

$$Obj \mapsto \mathbf{A}$$

$$\mathbf{A} = \langle a_1, \ldots, a_{15}, a_{16}, \ldots, a_{23} \rangle, \tag{6}$$

где $a_1, \ldots, a_{15}$ – идентификатор системы, $a_{16}, \ldots, a_{18}$ – день недели во время входа в систему (5), a_{19} – время входа в систему, полученное по формуле (4), $a_{20}, \ldots, a_{22}$ – день недели во время выхода из системы (5), a_{23} – время выхода из системы (4).

Для повышения точности вычисления вектор $\mathbb{A}$ может быть разделен на два вектора: $\langle a_1, \ldots, a_{15} \rangle$ и $\langle a_{16}, \ldots, a_{23} \rangle$, и обрабатываться двумя отдельными сетями.

Следует отметить, что в преобразовании не участвовал идентификатор пользователя. Это связано с тем, что для каждого пользователя и для каждого типа изучаемых объектов создается отдельная нейронная сеть и обработка логов осуществляется такими небольшими сетями. Такое разбиение на небольшие сети позволяет снизить нагрузку на графический адаптер (на котором производятся параллельные вычисления), упростить текущую работу с периодическим переобучением сети. При переобучении сети «старая» сеть инициализируется случайными значениями и проходит процесс обучения, только уже с более полным набором данных.

3 Поиск аномалий с использованием сети Кохонена

В работе применялись алгоритмы поиска аномалий, описанных в [5]. Основу этих методов составляет изучение полученных после обучения кластеров объектов. Кластер образуют объекты, для которых нейрон-победителем является один и тот же нейрон.

Алгоритм №1 (пороговый) выявляет малочисленные кластеры, которые свидетельствуют о том, что данная группа объектов отличается от остальных по неопределенному числу признаков и такая группа является меньшинством.

Алгоритм №2 основан на оценке расстояния от центра кластера до объектов из этого кластера. По сравнению с работой [5] было применено улучшение алгоритма, состоящее в том, что центр кластера и расстояние вычислялись отличным способом.

Пусть по результатам обучения сети мы имеем обученную сеть и разбиение обучающей выборки на кластеры. Для выявления аномалий среди полного множества объектов сделаем несколько предварительных вычислений. Рассмотрим один из полученных из обучающей выборки кластеров. Обозначим N – число векторов в кластере, $V = \{\mathbf{v}^i = \langle v_1^i, \ldots, v_k^i \rangle | i = 1 \ldots N\}$ – множество векторов из данного кластера. Вектора должны быть нормированными. Центром кластера будем считать матожидание от координат векторов этого кластера, а именно:

$$\mathbf{c} = \langle c_1, \ldots, c_K \rangle, \text{ где } c_i = \frac{1}{N} \sum_{l=1}^{N} v_i^l. \tag{7}$$

Вычислим радиус кластера:

$$r = \frac{1}{N} \sum_{i=1}^{N} d(\mathbf{c}, \mathbf{v}^i), \text{ где } d(\cdot, \cdot) - \text{Евклидово расстояние.}$$

Вычислим покоординатно среднеквадратическое отклонение в обучающей выборке:

$$\sigma = \langle \sigma_1, \ldots, \sigma_k \rangle, \text{ где } \sigma_i = \sqrt{\sum_{j=1}^{N} (v_j^i - c_j)^2}. \tag{8}$$

На этом предварительные вычисления заканчиваются. Значения $\mathbf{c}, r, \sigma$ сохраняются в базу данных и могут использоваться в дальнейшем. Для повышения точности выявления аномалий формулы (7) и (8) пересчитываются на полном множестве объектов. Все вектора, подаваемые на вход алгоритма должны быть нормированы. Выявление аномалий происходит следующим образом: если для некоторого нормированного вектора $\mathbf{v}$ из полного множества объектов выполняется следующее условие

$$d(\mathbf{v}, \mathbf{c}) > r + 2\|\sigma\|, \tag{9}$$

то объект, соответствующий вектору $\mathbf{v}$, является аномалией.

Алгоритм №3 основан на оценке расстояния от самого многочисленного кластера до остальных кластеров. Если расстояние превышает некоторое заранее заданное значение, то такой кластер считается аномальным и все вектора, входящие в этот кластер, считаются соответствующими аномалии.

Как правило вышеуказанные алгоритмы комбинируются и последовательно применяются, что позволяет выявлять различные виды аномалий в имеющемся наборе данных.

4 Проверка алгоритма

В качестве проверки алгоритма использовалось внедрение заведомо аномальных данных, таких как:

1. Авторизация в выходные дни при 5 дневном рабочем графике.
2. Авторизация во внерабочее время.
3. Оставление доступа к ПК во внерабочее время (т.е. события входа и выхода разнесены более чем на 1 сутки).
4. Вход на ПК или в систему, которой ранее пользователем не использовалась.
5. Опоздания и задержки на работе более чем на 30 минут.

Данные события включались в выгрузку из журнала событий с реальных ПК пользователей (см. табл. 1).

Заключение

Поиск аномалий является универсальным средством поиска инцидентов информационной безопасности, зарегистрированных в журнале событий Event Log. Нейронные сети Кохонена позволяют реализовывать поиск аномалий и являются отличным инструментом для изучения полученных результатов.

Статья выполнена при поддержке Правительства РФ (Постановление № 211 от 16.03.2013 г.), соглашение № 02.A03.21.0011.

Таблица 1: Проверка алгоритма

№ п/п	Кол-во журналированных сессий	Кол-во объектов в обучающей выборке	Тип аномального события	Есть ли аномалии в обучающей выборке	Кол-во ошибок, %	Примечание
1	57	57	1	Да	0,00 %	-
				Нет	0,00 %	-
			2	Да	0,00 %	-
				Нет	0,00 %	-
			3	Да	0,00 %	-
				Нет	0,00 %	-
			4	Да	0,00 %	-
				Нет	0,00 %	-
			5	Да	3,13 %	2 записи об опозданиях не были помечены как аномалия
				Нет	0,00 %	-
2	120	62	1	Да	0,00 %	-
				Нет	0,00 %	-
			2	Да	0,00 %	-
				Нет	0,00 %	-
			3	Да	1,67 %	-
				Нет	0,00 %	-
			4	Да	0,00 %	-
				Нет	0,00 %	-
			5	Да	1,67 %	-
				Нет	0,00 %	-

Список литературы

[1] URL: `https://technet.microsoft.com/en-us/library/cc722404(v=ws.11).aspx`.

[2] URL: `https://msdn.microsoft.com/en-us/library/windows/desktop/aa363646(v=vs.85).aspx`.

[3] T. Kohonen. *Samoorganizuyuschiesja karty.* M., Binom. Laboratoria znaniy, 2013 (In Russian).

[4] A.S. Batalov. Methods for increase Kohonen neural network training effectiveness. *Vestnik Permskogo universiteta. Ser.: Matematica. Mehanica. Informatika.* 3(11):86–93, 2012 (In Russian).

[5] V.S. Vedeneev, I.V. Bychkov. Anomaly detection with self-organization Kohonen neuron net. *Neyrokomyutery: razrabotka, primineniye*, 10:67–72, 2016 (In Russian).

Anomaly Detection in Event Log of OS Windows

Victor S. Vedeneev, Igor V. Bychkov

Windows event log anomaly detection is a tool, which allows detecting intentional and unintentional violations of information security, software malfunctions, and flaws of configuration of operating system. However, there is a few set of challenges for applying anomaly detection, which is choosing of anomaly detection algorithm, values of parameters of anomaly detection algorithms, checking correctness of results and analysis of results. This paper describes a way of solving such problem using Kohonen self-organizing neuron net.