Методы оптимизации выборки данных для определения аномального трафика

Г.А. Зубриенко, О.Р. Лапонина

***Аннотация* — Рассмотрен способ проектирования и реализации простейшего и легко расширяемого программного комплекса системы обнаружения проникновений (IDS), совмещающий эффективность сигнатурного подхода с гибкостью и производительностью адаптивного обучения. Цель данного подхода состоит в снижении ресурсоемкости задач IDS, избавлении от необходимости хранения лишних данных благодаря оптимизации алгоритма отбора данных для периодического повторного обучения сигнатурных классификаторов. Для обнаружения аномального трафика предлагается использовать подход из систем DLP с применением существующих методов обнаружения аномалий с последующим обновлением тренировочных данных и повторным обучением сигнатурного классификатора. Такой подход позволяет не только оперативно обнаруживать проведение атаки на целевую систему, но и адаптировать сигнатурные классификаторы к новым атакам.**

***Ключевые слова* — информационная безопасность, IDS, адаптивное обучение, обнаружение аномалий, сигнатурный анализ.**

I. ВВЕДЕНИЕ

Одной из критически важных задач обеспечения информационной безопасности является использование не только сигнатурных IDS, но и IDS, которые могут определять аномальный трафик. Организации защиты корпоративных информационных систем с использованием IDS посвящено множество публикаций [1-4]. Все известные атаки определяются по их сигнатурам, использование машинного обучения позволяет прежде всего увеличить долю корректно определяемых сессий-атак (True Positive Rate) при использовании злоумышленником модификации знакомой системе атаки. Наиболее часто для подобных целей в литературе описываются метод Association Rules и его модификация Fuzzy-Rule-Based [5]. Также часто встречается применение метода опорных векторов (Support Vector Machines, SVM) и родственных ему методов [6-7]. Успешно применяются и более редкие алгоритмы, например генетическое программирование, в частности Linear Genetic Programming, Multi-Expression Programming (MEP) и Gene Expression Programming [9], при этом MEP оказался способен обнаруживать 5 разных видов атак в 99.75% случаев. Однако вычислительная сложность алгоритмов распознавания новых атак не всегда позволяет выполнить обнаружение этих атак в режиме реального времени. Комбинирование классического сигнатурного подхода и методов обнаружения аномалий в трафике, а также применение базовых принципов адаптивного обучения позволяет в некоторых случаях значительно ускорить реакцию на атаку, даже если используется ранее неизвестная уязвимость.

Для обнаружения неизвестных угроз требуется, чтобы существовало явное отличие трафика, приходящего с атакующего хоста, от трафика обычных хостов. Это свойство позволяет обучить классификатор обнаруживать атаки, для которых отсутствует на данный момент их сигнатура. Для этого достаточно выполнить кластеризацию трафика или получить гистограмму распределения значений каждого атрибута [8].

Комбинируя современные платформы обработки больших массивов данных с сигнатурным подходом, а также с методами обнаружения аномалий [10], можно обеспечить качество сигнатурного классификатора в течение длительного времени. Дополнительное использование подходов, применяющихся в рекомендательных системах, а именно адаптивное обучение, позволяет не только использовать загруженные заранее сигнатуры, но и автоматически обновлять их: даже абсолютно новый тип атаки через некоторое время будет распознан IDS и при повторном проведении аналогичной атаки она будет обнаружена.

Одна из важных особенностей IDS – использование данных в режиме реального времени, что делает их похожими на рекомендательные системы (recommendation engines). В отличие от классических алгоритмов машинного обучения, которые формируют модель исключительно на основании исторических данных, рекомендательные системы должны непрерывно обновлять полученную модель по мере поступления новых данных. Ситуация в защищаемой сети может меняться также быстро, как и потребительское, социальное поведение пользователей в интернете. В результате модели, полученные на исторических данных, быстро обесцениваются и перестают обеспечивать приемлемый уровень защиты – в первую очередь растет количество ложных срабатываний (false positive rate, FPR), а также снижается количество успешно обнаруженных атак / растет доля пропущенных (true positive rate, TPR / missed attack rate, MAR). Непрерывное обновление модели в случае классического подхода подразумевает несоразмерные со степенью улучшения FPR/TPR

Статья получена 28 августа 2016.
Работа выполнена при поддержке РФФИ, проект 16-07-00873 А.
Г.А. Зубриенко – МГУ имени М.В. Ломоносова (email: ops.xeon@gmail.com).
О.Р. Лапонина – МГУ имени М.В. Ломоносова (email: laponina@oit.cmc.msu.ru).

затраты вычислительных ресурсов. Рекомендательные системы обновляют модель, основываясь на результатах ее работы, что существенно снижает ресурсозатраты, а также избавляет от необходимости хранить исторические данные [11].

Эффективность классических методов можно существенно повысить за счет их комбинирования с методами анализа в режиме реального времени, однако для этого потребуется изменить процедуру отбора данных (data sampling), включив в нее этап проверки корректности модели. Таким образом, предлагается совместить сигнатурный подход с обнаружением аномалий следующим образом:

- Определение явных сессий-атак, т.е. таких сессий, чьи параметры сильно отличаются от текущего профиля трафика, возложить на алгоритм поиска аномалий, использовав описанные в литературе подходы к его реализации.
- Определение всех остальных сессий-атак возложить на сигнатурный классификатор, построенный по распространенным алгоритмам – дерево решений или случайный лес.
- Обеспечение обновления данных для обучения сигнатурного классификатора возложить на алгоритм поиска аномалий – для атак, а также на отдельный алгоритм адаптивного подбора нормальных сессий.

Упомянутый алгоритм адаптивного подбора должен обеспечить снижение количества данных, используемых при обучении, а также существенно повысить качество сигнатурного классификатора в долгосрочной перспективе.

II. ОБУЧЕНИЕ IDS В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

A. Используемое программное обеспечение

Для построения классификатора необходимо обеспечить сбор и хранение сетевого трафика, а также его быструю обработку с целью формирования наборов данных, предназначенных для обучения. В качестве базовой платформы может быть использован любой программный комплекс, работающий по технологии MapReduce. В данной работе была использована Hadoop-подобная платформа Splunk Enterprise версии 6.3 [12].

B. Обучение классификаторов

В качестве исходных данных для обучения использовались наборы DARPA [13] от 1998 года: они хранились в отдельной базе данных (индексе) Splunk. Задача обучения для модельной системы была упрощена до бинарной классификации (Attack – Not Attack), поэтому из листа сигнатур удалялась информация о типе атаки, вместо которой добавлялось булево поле IsAttack: значение 1 соответствовало сессии, классифицированной как попытка атаки на целевую машину, 0 – обычной сессии. Термин «сессия» здесь подразумевает не реальную TCP-сессию, а некий набор пакетов с определенными характеристиками, пересылаемый между целевым и потенциально атакующим хостом за гранулярный промежуток времени (минута, час и др.). Такой подход позволяет не привязывать классификатор к конкретным TCP-сессиям, а рассматривать потенциально атакующий хост как единую сущность, что существенно упрощает структуру классификатора (например, дерево решений) и, как следствие, снижает вероятность переобучения. Рассматривалась реакция системы на атаки сканирования по TCP различного типа, при этом базовые классификаторы практические не обладали возможностью обнаружить сканирование целевого хоста. Подготовка данных велась исключительно на сервере Splunk с помощью встроенного языка Search Processing Language (SPL). В результате выполнения SPL-скрипта сетевой трафик, используемый для обучения, приводился к виду

X_0	…	$X_n - attribute$	$Y - target \in \{0,1\}$
a_0	…	a_n	y_0
…	…	…	…
a_0	…	a_m	y_m

Далее полученный набор данных передается на цепочку пользовательских команд (Python 2.7). Важно отметить, что установленная версия Splunk (6.3) использует ядро Python версии 2.6 для исполнения всех скриптов поисковых команд. Эта особенность приводит к невозможности прямой интеграции с любыми библиотеками, использующими Python 2.7 и тем более 3.x. Однако проблему можно решить, используя скрипт-обертку.

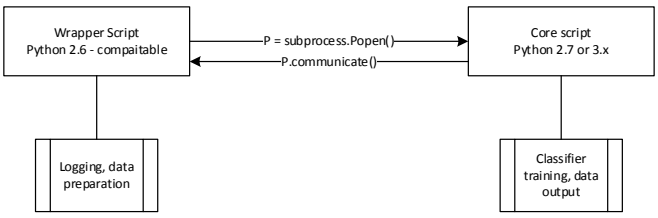

Рис. 1. Схема работы Custom Search Command для Splunk 6.3

Функция обертки сводится к передаче в скрипт-ядро информации от Splunk, в том числе параметров команды, например, процент данных для обучения и тип алгоритма, а также самих данных через механизм межпроцессного обмена: wrapper выполняет печать в stdout, core считывает данные из stdin, подключенного к stdout процесса wrapper. После отработки core обертка возвращает в Splunk результат, и он отображается в пользовательском интерфейсе. Для построения классификаторов можно использовать любые алгоритмы и библиотеки; в данной работе использовался популярный пакет scikit-learn [14] и три основных алгоритма: DecisionTreeClassifier, RandomForestClassifier и DecisionTreeClassifier + AdaboostClassifier. Тестовый запуск полного цикла построения классификатора на данных DARPA для протокола TCP с использованием следующих данных: 10.06-13.06 для 1998 года и IsAttack = 0 и без ограничений по времени для IsAttack = 1 с разбиением 70% train – 30% test и классификатором

DecisionTreeClassifier1 привел к следующим результатам:

Таблица 1. Результат классификации сессий набора DARPA с использованием Decision Tree Classifier, 70/30

Корректность определения атаки	Количество сессий
Пропуск атаки (False Negative)	28
Ложные срабатывания (False Positive)	7
Определение корректного трафика (True Negative)	9530
Определение атаки (True Positive)	476

Таким образом, из всего множества сессий ~0.28% были классифицированы неверно, при этом FPR (ложные срабатывания) также составил всего лишь ~0.07% от общего числа сессий. Сессии-атаки между тем определились не так хорошо, как ожидалось: ~5.6% атак системой были бы проигнорированы, однако даже с простейшим вектором атрибутов X, включающим только IP-адреса, флаги и объем переданной информации, модель уже работает намного лучше, чем случайный классификатор. Этот результат легко немного улучшить, всего лишь расширив вектор атрибутов X такими параметрами как число портов-источников и портов-цели на каждый флаг TCP.

Таблица 2. Результат классификации сессий набора DARPA с использованием Decision Tree Classifier, 70/30, с расширением вектора атрибутов сессии

Корректность определения атаки	Количество сессий
Пропуск атаки (False Negative)	20
Ложные срабатывания (False Positive)	6
Определение корректного трафика (True Negative)	9531
Определение атаки (True Positive)	484

В этом случае FPR составит те же 0.07%, а вот качество обнаружения атак возрастет – доля упущенных атак снизится до 3.9%. При этом данные классификаторы оказались не лучше чем random-классификаторы при проверке на тестовой выборке из того же набора DARPA.

C. Адаптивное обучение

Основой для адаптивного обучения использовался наиболее эффективный с вычислительной точки зрения метод обнаружения аномалий, основанный на анализе профиля сетевого трафика, представленного кластерами, сформированными алгоритмом k-Means. Для проведения экспериментов использовалась встроенная в Splunk реализация алгоритма.

Всего использовалось два периодических процесса кластеризации:

- Текущий профиль сетевого трафика, построенный по сетевым событиям за последние 14 дней, с частотой формирования раз в 5 минут. Условие независимости от предыдущего состояния системы, необходимое для применения адаптивных алгоритмов, в данном случае не нарушается, так как текущий профиль сети описывается сетевыми событиями за некоторый промежуток времени и может включать в том числе исторические данные;
- Текущий профиль трафика в тренировочном индексе. Частота формирования раз в 60 минут.

Указанные процессы управлялись Splunk. Скрипт анализа также запускался планировщиком Splunk, выполняя три основных задачи: выявление сильных отклонений от профиля трафика с копированием событий из базы данных сетевого потока в тренировочный индекс, а также модификацией справочника сигнатур; отбор событий из нормальных сессий для адаптации профиля тренировочного индекса к текущему профилю сети на основе стохастического подхода.

Предыдущий профиль – *oldcls*, текущий – *newcls*, профиль тренировочного индекса – *traincls.*

Набор меток времени для предыдущих аномалий – lts (используется для предотвращения записи дубликатов в лист сигнатур).

// все сессии конвертируются в тип Cluster

- *oldcls*, *newcls*, *traincls* = to_clusters([old_path, new_path, train_path])
- *lts* = load(lts_path), $anomalies = \emptyset$, $signatures = \emptyset$, $suspicious = \emptyset$, $normal = \emptyset$, $trainevents = \emptyset$
- $anomalies = anomalies \cup \{index, index \in \{deviation\ rates(newcls); deviation\ rates_i \geq threshold\}\}$ – аномальные кластеры
- $suspicious = suspicious \cup \{index, index \in \{deviation\ rates(newcls); deviation\ rates_i \in [threshold - 0.05, threshold]\}\}$ – подозрительные кластеры
- $normal = \{indexes, indexes_i \in [0..len(newcls)]\} \setminus anomalies \setminus suspicious$ – кластеры из нормальных сессий
- for *anomaly* in *anomalies*:
  - *allsessions = newcls[anomaly].get_coords()*
  - *sessions* = $\{session, session.StartTime > \arg\max lts, session \in allsessions\}$ – т.е. выполняется поиск новых аномалий
  - $lts = lts \cup \{session.StartTime, session \in sessions\}$
  - $trainevents = \{ trainevents \cup \{events, events_i \in getevents(sessions)\}\}$ – из обнаруженных сессий извлекаются соответствующие им события
  - $signatures = \{ signatures \cup sessions \}$
- update_signatures(*signatures*), update_lts(*lts*) - лист сигнатур дополняется аномальными сессиями (если найдены)
- update_suspicious_file(*suspicious*) – отдельно записываются подозрительные сессии
- if (cluster_error(*oldcls*) / cluster_error(*newcls*) - 1) >= THRESHOLD: - при необходимости в тренировочный индекс также отбираются события из нормальных сессий (trainevents)
- $policy = evalpolicy(normal, FPR, FNR) \mid deterministic$ // используется либо вычисляемая политика, либо дереминистская
- $trainevents = \emptyset$
- for *normal_id* in *normal*:
  - variances = {*tr_cluster*.cluster_error_variance(*newcls*[normal_id].*get_events*()) for *tr_cluster* in *traincls*} – рассчитывается изменение профиля при добавлении событий
  - if *variances* in *policy*:

[1] http://scikit-learn.org/stable/modules/tree.html

trainevents.add(newcls[normal_id].*get_events())*

- return to_splunk_input(*trainevents*)

Алгоритм 1. Поиск аномалий с последующим отбором событий для классификаторов.

В качестве метрики аномальности (deviation rate) использовалось среднее расстояние между центром анализируемого кластера (i) и центрами прочих кластеров [15]:

$$devr_i = \frac{\sum_{n=1,n\neq i}^{N} C_i.distance(C_n)}{N} \quad (1)$$

Аномальными считались кластеры, для которых $devr_i$ больше или равен 95-му перцентилю всего множества *devr*, определенному по алгоритму поиска непрерывного перцентиля. Профиль реального трафика и тренировочного трафика можно визуализировать, приведя многомерные координаты центра к двумерным по формуле

$$x = \sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N}} \quad (2)$$

$$y = \sqrt{\frac{\sum_{i=1}^{N} y_i^2}{N}} \quad (3)$$

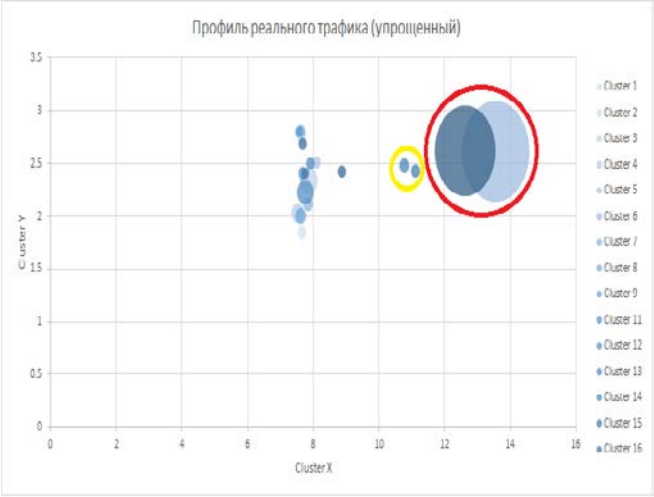

Рис. 2. Упрощенный профиль реального трафика. Красным выделены аномальные кластеры, желтым – подозрительные

Для определения обычных сессий-кандидатов на включение в тренировочный индекс использовалась величина ошибки кластеризации:

$$error(C) = \sum_{p\in C_i}(d_{p,center(C_i)})^2 \quad (4)$$

$$error(\{C_i, i \in [0..N]\}) = \frac{\sum_i error(C_i)*len(C_i)}{\sum_i len(C_i)} \quad (5)$$

Отбор событий в тренировочный индекс не носил обязательного характера и определялся системой на основе выбора из набора заранее определенных политик. Каждая из политик определяет принцип отбора событий в тренировочный индекс на основе единственной метрики – изменения ошибки кластеризации профиля индекса, то есть степени разброса точек внутри кластеров.

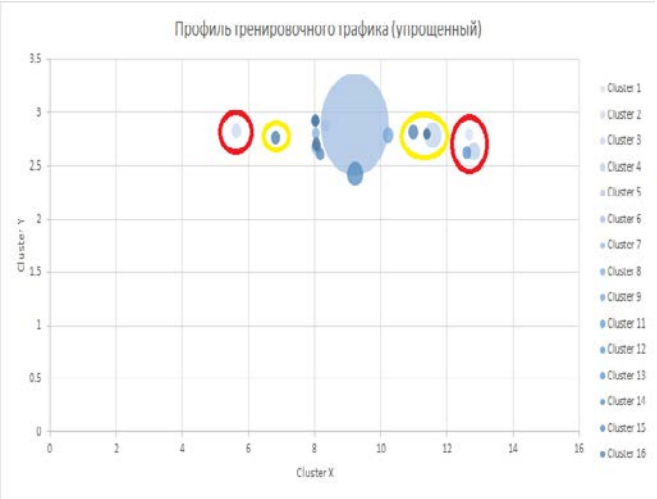

Рис. 3. Упрощенный профиль тренировочного трафика. Красным выделены аномальные кластеры, желтым – подозрительные

D. ВЫБОР ПОЛИТИКИ

Выбор политик осуществлялся на основании гипотезы о наличии зависимости FPR или MAR классификатора от профиля трафика, на котором он построен, то есть:

$$FPR, MAR \sim f(error(\{C_i, i \in [0..N]\})) \quad (6)$$

Данная гипотеза представляет интерес для последующих исследований, например в целях установления точной формы этой зависимости, что открывает возможность алгоритмически более эффективной подстройки любой IDS к окружению. В данной работе лишь ставится цель показать наличие подобной зависимости экспериментально на примере самообучающейся системы.

В теории адаптивного обучения существует понятие «награды», значение которой следует максимизировать, выполнив то или иной действие. Набор таких действий называется политикой системы (*policy*). Для управления качеством модели были использованы четыре политики. Во всех случаях «наградой» являлось сохранение или уменьшение FPR/MAR модели:

1. DECREASE_ERROR (DE): политика снижения средневзвешенной ошибки кластеризации (снижение энтропии профиля трафика).
2. STABILIZE_ERROR (SE): политика сохранения текущей средневзвешенной ошибки кластеризации около заданного уровня (сохранения текущего уровня энтропии).
3. INCREASE_ERROR (IE): политика увеличения средневзвешенной ошибки кластеризации (увеличение энтропии профиля).
4. BALANCED (B): политика увеличения объема данных в профиле с близким к нулю изменением средневзвешенной ошибки кластеризации (масштабирование профиля).

Перед непосредственным исполнением политики необходимо осуществить ее выбор. Использовался стохастический подход с вычислением вероятностей в процессе работы системы. В соответствии с теорией адаптивного обучения:

$$\sum_{i=1}^{4} p_i(\pi_i) = 1 \quad (7)$$

Можно определить вероятность выбора политики p_i как функцию от FPR модели, полученного в результате применения политики на предыдущем шаге (если необходимо сократить число ложных срабатываний до минимума):

$$p_i = \frac{1/FPR^{\pi_i}}{\sum_i 1/FPR^{\pi_i}} \quad (8)$$

Или от доли упущенных атак (MAR), если необходимо улучшить качество детектирования при возможном росте FPR:

$$p_i = \frac{1/MAR^{\pi_i}}{\sum_i 1/MAR^{\pi_i}} \quad (9)$$

После выбора политики также необходимо выполнить поиск тех сессий, которые удовлетворяют условиям политики, то есть изменяют ошибку кластеризации тренировочного индекса в нужном направлении. Таким образом, применяемая политика будет определяться лишь текущей ситуацией, а выбор политики будет основан на результатах применения предыдущих; выбор политики определяет набор событий, попадающих в тренировочный индекс для обеспечения адекватности профиля текущей ситуации.

III. Результаты

Примерно за месяц активной работы система добавила в лист сигнатур 430 модельных сессий (0.14% от всех модельных сессий в листе сигнатур), идентифицированных как атаки, при этом было охвачено 100% атак, проводившихся в рамках данной работы с помощью утилиты ZenMap с использованием различных типов сканирования. При этом размер тренировочного индекса вырос всего лишь на 4%, а из сетевого потока было отобрано 9.43% всех событий. За неделю работы был улучшен результат классификации тестовой выборки DARPA, содержащей атаки сканирования (AUC вырос с ~50% до 75.7%):

При проведении периодических атак полного сканирования TCP с разной интенсивностью и в разное время наблюдалось первоначальное небольшое ухудшение исходного классификатора с последующим восстановлением и улучшением качества анализа:

Рис. 4. Влияние адаптивного обучения на качество классификатора при проведении периодических атак сканирования

В приведенном примере первоначально система выбрала консервативный путь (политика BALANCED), сопровождавшийся увеличением размера выборки с последующей деградацией качества. Однако уже на следующей итерации была выбрана политика увеличения ошибки кластеризации (INCREASE ERROR), что разбавило обучающую выборку данными, характерными для проводимой атаки и позволило далее выйти на этап стабилизации и постепенного масштабирования. Поскольку данная стратегия оказалась успешной, система начала ее эксплуатировать и далее, выбрав IE даже после снижения MAR до уровня 2% вместо стартовых 3%. Однако при прекращении активных атак наблюдается очередной цикл адаптации: ухудшение качества классификатора даже при проверке на исторических данных – улучшение – стабилизация:

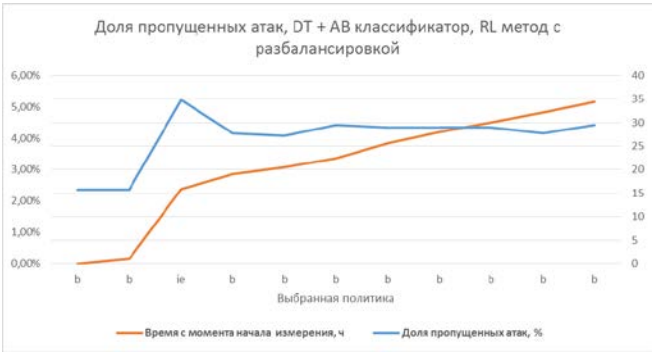

Рис. 5. Влияние адаптивного обучения на качество классификатора при резком прекращении атак сканирования

Аналогичным образом система ведет себя при использовании FPR в качестве reward-метрики:

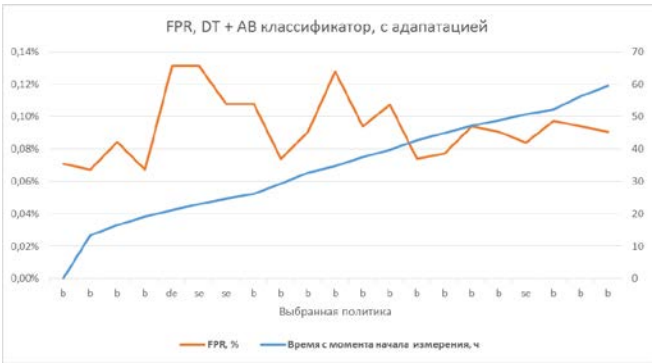

Рис. 6. Влияние адаптивного обучения на качество классификатора (FPR) при резком прекращении атак сканирования

Если окружение IDS и тип атак долгое время не меняется, она постепенно приходит в равновесное состояние с минимальным значением FPR/MAR, при этом минимум зависит в основном от возможностей самого сигнатурного классификатора. При этом для каждой политики установится значение вероятности быть выбранной равное $1/N$, где N – число используемых политик. Однако любое резкое изменение профиля, сопровождающееся ростом FPR или MAR, а также смена используемой метрики с FPR на MAR или обратно выведет систему из равновесия с временным ухудшением показателей классификатора. При условии подбора классификаторов и методов подготовки данных, адекватных анализируемому протоколу, качество анализа стабилизируется в течение нескольких часов. Стоит отметить, что данное время можно существенно снизить, увеличив вычислительную мощность IDS.

IV. ЗАКЛЮЧЕНИЕ

Предложенный алгоритм позволяет оптимизировать отбор данных для обучения сигнатурных классификаторов, что приводит к существенному снижению ресурсоемкости задач IDS, а также позволяет избавиться от необходимости хранения лишних данных.

Предложенная схема реализации поддержки качества классификатора с применением различных политик отбора событий в тренировочный индекс показала свою эффективность для атак сканирования по протоколу TCP. В дальнейшем большой интерес для исследований представляет применение созданных алгоритмов к более сложным видам атак, а также к протоколу HTTP для реализации активной защиты от SQL-инъекций и других атак на веб-приложения.

Библиография

[1] A. Shabtai, Y. Elovici, L. Rokach A Survey of Data Leakage Detection and Prevention Solutions // Springer Briefs In Computer Science. 2012

[2] R. Mogull Understanding and Selecting a Data Loss Prevention Solution // SANS Institute

[3] Y. Kim, N. Park, S.K. Un An Advanced Data Loss Prevention System Being Able to Respond Data-Leaking Incidents Using e-Discovery Primitives // WorldComp 2012

[4] C. Phua Protecting Organisations from Personal Data Breaches // Computer Fraud & Security. January 2009

[5] S. Dua, X. Du Data Mining and Machine Learning in Cybersecurity // Auerbach Publications P. 62-65

[6] D. Du, L. Yu, R.R. Brooks Sematic Similarity Detection For Data Leak Prevention // CISR'15 Proceedings of the 10th Annual Cyber and Information Security Research Conference. Article No. 4. ACM New York, NY, USA. 2015

[7] G. Xiang, J. Hong, C.P. Rose, L.Cranor CANTINA+: A Feature-Rich Machine Learning Framework for Detecting Phishing Web Sites // ACM Trans. Inf. Syst. Secur. 14, 2, Article 21. September 2011

[8] M. Goldstein, A. Dengel Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm // KI-2012: Poster and Demo Track. p. 59-63.

[9] A. Abraham, C. Grosan, C. Martin-Vide Evolutionary Design of Intrusion Detection Programs // International Journal of Network Security, Vol. 4, No.3, March 2007 PP. 328-339.

[10] S. Dua, X. Du Data Mining and Machine Learning in Cybersecurity // Auerbach Publications p. 57-61.

[11] A. Paprotny, M. Thess Realtime Data Mining. Self Learning Techniques for Recommendation Engines // Springer International Publishing Switzerland, 2013.

[12] Splunk Enterprise Documentation // http://docs.splunk.com/Documentation/Splunk.

[13] DARPA Intrusion Detection Evaluation Datasets (1998) // https://www.ll.mit.edu/ideval/data/.

[14] Pedregosa *et al.* Scikit-learn: Machine Learning in Python // Journal of Machine Learning Vol. 12. 2011 p. 2825-2830.

[15] L. Nakhleh Data Clustering // Coursera Algorithmic Thinking Part 1. Rice University, Department of Computer Science.

Приложение к алгоритму 1

Алгоритм записан в псевдокоде по следующим правилам:

- Любой набор значений одного типа рассматривается как множество, все операции над такими наборами описываются с использованием нотации операций над множествами.
- Для каждого метода описывается лишь результат действия.
- Операция присваивания обозначается как “=”.
- Операция *in* может означать как проверку включения в множество, так и соответствие некому набору правил.

Алгоритм использует операции над кластерами-объектами типа Cluster, который содержит:

1. Cluster_Id – номер кластера.
2. Coords – множество всех входящих в кластер модельных сессий.
3. Events – множество всех входящих в кластер сетевых событий.
4. Centroids – координаты центра кластера.

Объекты типа Cluster также должны поддерживать следующую логику:

- *get_events()* – возврат множества событий, входящих в кластер.
- *get_coords()* – возврат множества сессий, входящих в кластер.
- *get_center()* – возврат координат центра.
- *centroid_distance(other_cluster)* – возврат расстояния до другого объекта типа Cluster по метрике Евклида.
- *merge_clusters(other_cluster)* – слияние с другим объектом типа Cluster с пересчетом центра нового кластера; вызов метода должен изменять вызвавший его объект.
- *cluster_error()* – подсчет суммарного среднеквадратичного расстояния от входящих в кластер сессий до центра кластера. Метод можно изменять в зависимости от набора атрибутов, используемых для кластеризации.
- *cluster_error_variance(events)* – возврат относительного изменения ошибки кластеризации при добавлении в кластер событий из набора events.

В алгоритме использованы следующие методы:

- *to_clusters(string path)* – выполняет чтение сформированного набора модельных сессий по адресу *path* и конвертирует их в множество объектов типа Cluster.
- *load(string path)* – выполняет чтение меток времени последнего обновления сигнатур и тренировочного профиля. Данные метки позволяют избежать повторного включения аномальных сессий в лист сигнатур.
- *deviation_rates(Cluster[] clusters)* – выполняет расчет среднего расстояния между кластерами, с выдачей словаря вида { cluster_id : average_distance }. В описании алгоритма *index* - синоним *cluster_id.*
- *getevents(session)* – выполняет поиск событий, вошедших в модельную сессию путем запроса к API Splunk. Параметры поиска зависят от того, какие атрибуты события используются в формировании сессии.
- методы *update_signatures() update_lts(), update_suspicious()* – обновляют хранящуюся на диске информацию: лист сигнатур, метки времени, лист подозрительных сессий и др.
- *evalpolicy(PolicyType type)* – вычисляет политику отбора нормальных событий в тренировочный индекс для следующей итерации обучения. Может возвращать константу или неизменяемый объект, если политика детерминистская.
- *policy* – метод, определяющий текущую политику отбора событий в тренировочный индекс.

Data Sampling Techniques for Anomaly Detection in Network Traffic

G.A. Zubrienko, O.R. Laponina

***Abstract* — An architecture and a simple implementation of auto-scaling intrusion detection system is considered. A proposed concept combines efficiency of signature-based detection with flexibility and performance of adaptive learning. There are several goals founding the concept: reduction of computing resources required by IDS, reduction of the amount of data stored. These goals are achieved by optimizing a data sampling algorithm feeding the source data used in training signature-based classifier. Thus, anomalies are detected using classic anomaly detection methods with further update of training dataset and signature-based anomaly classifier as well. This approach allows not only real-time attack detection, but also rapid adaptation of the signature-based classifier to new attack types.**

***Keyword* — information security, IDS, adaptive learning, anomaly detection, signature-based detection.**