высокую асимптотическую точность оценивания плотности, а в конечном итоге и точность клавиатурной аутентификации.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

3. *Брюхомицкий Ю.А., Казарин М.Н.* Параметрическое обучение биометрических систем контроля доступа / Вестник компьютерных и информационных технологий. – М.: Изд-во Машиностроение, 2006. – № 2 (20). – С. 6–13.

4. *Брюхомицкий Ю.А.* Классификация нестационарных вероятностных биометрических параметров личности // Известия ЮФУ. Технические науки. – 2008. – №8 (85) – С. 147 – 154.

5. *Фомин Я.А., Тарловский Г.Р.* Статистическая теория распознавания образов. – М.: Радио и связь, 1986. – 264 с.

Брюхомицкий Юрий Анатольевич
Технологический институт Федерального государственного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: bya@tsure.ru.
347928, г. Таганрог, ул. Чехова, 2.
Тел.: 8 (8634) 371-905.
Кафедра безопасности информационных технологий; доцент.

Bryukhomitsky Yuri Anatolyevich
Taganrog Institute of Technology – Federal State-Owned Educational Establishment of Higher Vocational Education “Southern Federal University”.
E-mail: bya@tsure.ru.
2, Chekhova st., Taganrog, 347928, Russia.
Phone: +7 (8634) 371-905.
Department of IT-Security; associate professor.

УДК 681.324

Г.Э. Абрамов

МОДЕЛЬ АНОМАЛЬНОГО ПОВЕДЕНИЯ СИСТЕМЫ НА ОСНОВЕ ВЕРОЯТНОСТНЫХ СУФФИКСНЫХ ДЕРЕВЬЕВ

Описывается метод применения вероятностных суффиксных деревьев для обнаружения аномального поведения программ. Используется «отпечаток» нормального поведения приложений с целью в дальнейшем обнаружить аномальное поведение как нечто, отклоняющееся от модели. В качестве основной модели используется вероятностные суффиксные деревья.

Вероятностное суффиксное дерево; PST, обнаружение аномального поведения.

G.E. Abramov

MODEL OF THE ANOMALOUS BEHAVIOR OF THE SYSTEM BASED ON PROBABILISTIC SUFFIX TREES

Described the method of probabilistic suffix trees for detecting anomalous behavior of programs. Use "fingerprint" of the normal behavior of applications in order to further detect anomalous behavior as something deviating from the model. As a basic model uses a probabilistic suffix trees.

Probabilistic suffix tree; PST; detection of abnormal behavior.

Классификация активности приложений на основе вероятностных суффиксных деревьев

Целью исследований является нейтрализация злонамеренного поведения программ. Будем стремиться получить «отпечаток» нормального поведения приложений с целью в дальнейшем обнаружить аномальное поведение как нечто, отклоняющееся от модели. В качестве основной модели будем использовать вероятностное суффиксное дерево (probabilistic suffix tree, PST), разработанное D. Ron [1]. Эта модель допускает эффективное использование информации высокого порядка, такой, как системные вызовы.

Вероятностное суффиксное дерево (probabilistic suffix tree, PST), описанное в [1], это n-арное дерево, в котором узлы организованы так, что корневой узел представляет безусловную вероятность каждого символа в алфавите, в то время как узлы на следующих уровнях представляют вероятности возникновения следующего символа при условии, что уже наблюдалась комбинация одного или большего числа символов (т.е. при наличии истории). Вероятности – это относительные оценки, вычисляющие частоту возникновения символа в обучающем примере или примерах. PST также имеют свойство порядка, соответствующее глубине дерева, так что PST порядка k содержит k+1 уровень.

Для иллюстрации, рис. 1 показывает PST третьего порядка, порожденное из образца «2 1 1 3 1 3 1 3 3 4», где символы «2» и «4» однозначно используются как начало и конец строки.

Начиная с корневого узла, который представляет пустую строку, каждый узел является суффиксом всех своих потомков, поэтому модель получила название суффиксного дерева. Корень, представляющий пустую строку, является непосредственным суффиксом всех единичных символов.

Числа в круглых скобках, расположенные рядом с узлами, являются условными вероятностями следующих символов, за исключением того, что распределение вероятностей следующих символов для корневого узла имеет вероятность, равную нулю, для конечного символа «4», поскольку этот символ сам не имеет следующих символов в обучающем примере. Необходимо также отметить, что переходы к следующему символу осуществляются от одной ветки к другой, а не от родительского узла к потомку, вследствие формата суффикса узла.

Ниже представлен рекурсивный алгоритм для построения дерева. Пример для построения сканируется для определения вероятности каждого символа. Каждый символ с ненулевой вероятностью становится потомком корневого символа, и для каждого такого узла пример пересканируется для определения распределения вероятностей следующих символов. Это может создать новые листья и добавить новый уровень к PST.

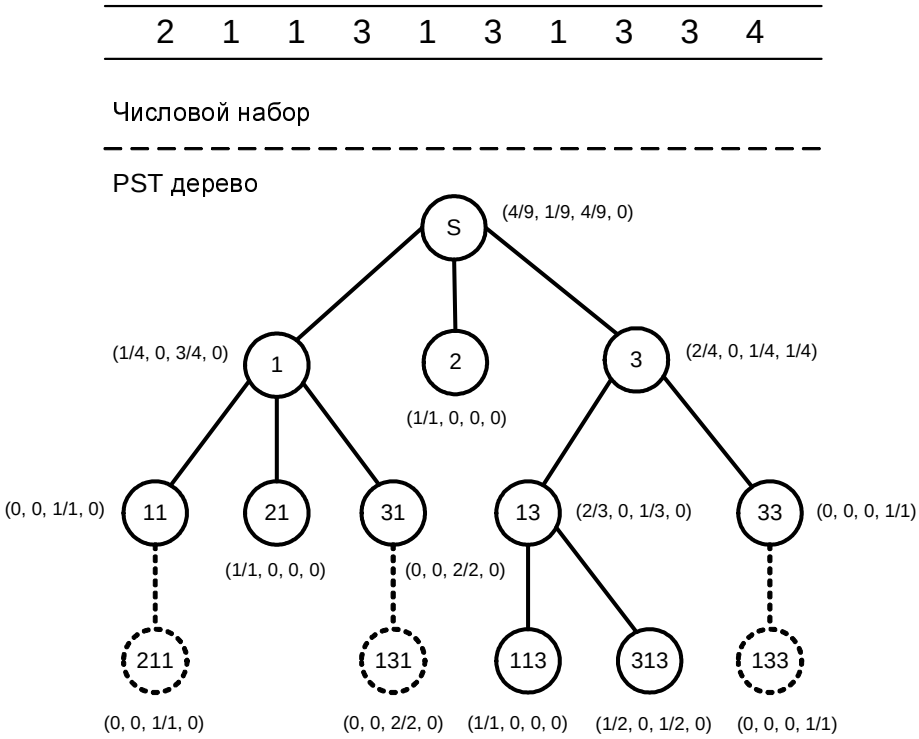

Рис. 1. Пример построения PST

Построение рекурсивно добавляет дополнительные уровни путем проверки каждого текущего листа для определения, могут ли быть созданы новые листья среди потомков данного узла. Информация о символах, предшествующих данному символу, сохраняется, поскольку эти символы могут встречаться в суффиксной информации потомков данного узла.

Узел добавляется, только если подстрока символов, которая соответствует метке узла, есть в обучающем примере, и для него существует ненулевое распределение вероятности следующего символа. Многие ветви отмирают прежде, чем распространяются до максимума глубины.

Глубина (то есть порядок) дерева может быть установлена в максимальное допустимое значение, или по умолчанию, это длина входной последовательности. Для каждого узла дерева память должна выделяться динамически.

Далее рекурсивный восходящий процесс усечения должен удалить ветви, которые предоставляют ту же информацию, что и родительские узлы. Удаление этих узлов «аналогичных родительским» снижает избыточность в структуре дерева и таким образом, создает экономичную модель. Пунктирные линии и фигуры на рис. 1 показывают отсеченные узлы и соответствующие ветви. Отсечение базируется на вероятностной информации.

Главная роль PST в нашем случае – это стохастическое представление обучающих данных в древовидной структуре. Несколько PST, потенциально представляющих разные или подобные обучающие примеры, могут быть слиты в одно дерево, которое может быть получено другим путем при использовании всех индивидуальных обучающих последовательностей. Эта операция позволяет прово-

дить прямые манипуляции с самими PST без потери дополнительных вычислительных мощностей для разборки большого дерева PST. В отношении реконструкции, как упомянуто выше, определенные узлы убираются из PST, если они предоставляют ту же стохастическую информацию, как и их предки.

Алгоритмы построения PST

Существует алгоритм для реконструкции вероятностной информации следующих символов для удаленных узлов и, таким образом, реконструкции или заполнения узлов в PST. Эта техника может использоваться для выполнения аналитических операций с PST, таких как сравнение двух деревьев. Есть несколько вариантов алгоритмов нахождения расстояния между двумя PST для нахождения степени похожести или непохожести между PST [2].

Основываясь на изложенной выше информации, приведём далее алгоритм построения PST.

1. Исходный образец сканируется для определения вероятности каждого символа.

2. Каждый символ с ненулевой вероятностью становится дочерним узлом корневого узла, и для каждого такого узла исходный образец сканируется заново для определения вероятности следующего символа.

3. Рекурсивно добавляются новые уровни, путём анализа каждого конкретного узла. Определяется, необходимо ли создавать новые узлы-потомки.

4. Информация о предшествующих символах запоминается, так как они могут появиться как дети потомков самих себя.

5. Узел добавляется только тогда, если подстрока символов, относящихся к узлу, находится в исследуемом исходном образце и если для него существует ненулевая вероятность.

6. Глубина (порядок) может быть изменена на максимально возможную или по умолчанию равна длине входного исходного образца.

7. После этого проходом снизу вверх удаляются узлы, содержащие такую же статистическую информацию (вероятности), как и их родители. Этим устраняется избыточность структуры дерева.

Можно записать алгоритм при помощи псевдоязыка:

Построение ПСТ

```
Если (тек.глубина=макс. глубина), то выход из рекурсии
Вызов Заполнение списка префиксов
Для I от 0 до Макс_число_знаков_алфавита
  Если (префикс(I)<>пустой_указатель)
    Конец_списка->правый_брат = префикс(I)
    T = префикс(I)
  Если нет
    Текущий узел->левый_сын = префикс(I)
    T = префикс(I)
```

Заполнение списка префиксов

```
Для I от 0 до Макс_число_знаков_алфавита
префикс(I) = пустой_указатель
для I от 1 до размер_строки-уровень_узла
Если текущий_уровень=0 или нашли вхождение текущей подстроки
```

Если узел, соответствующий данному символу_префиксу, не существует, создаём его

Если существует, узел->вероятность[символ_постфикс]+=1.0

```
Нормализация вероятностей (что означает нормализация)
Для I от 0 до Макс_число_знаков_алфавита
 Если узел существует
   Для J от 0 до Макс_число_знаков_алфавита
    Tsum += узел->вероятность(J)
   Для J от 0 до Макс_число_знаков_алфавита
    узел->вероятность(J) /= Tsum
Найти_ближайший_узел
Если узел=пустой указатель, вернуть пустой указатель
Если узел->глубина = 0
 T = найти_ближайший узел (узел->сын, суффикс, длина суффикса)
Иначе
 Пока узел <> пустой указатель
  Если узел->суффикс(0)=суффикс(размер суффикса – узел->уровень)
выход из цикла
   Узел = узел->брат
 Если узел = пустой указатель, вернуть пустой указатель
 Иначе если Узел->уровень = размер суффикса вернуть узел
 Иначе
 T = найти ближайший узел (узел->сын, суффикс, размер суффикса)
 Если t = пустой указатель вернуть узел иначе вернуть T
```

Алгоритмы сравнения PST

Для использования суффиксных деревьев в задачах обнаружения атак необходимо иметь механизм сравнения различных PST. В этом случае процедура сравнения просматривает образец с использованием PST, для того чтобы вычислить кумулятивную вероятность возникновения символов в образце.

Для сравнения образцов с использованием PST начинаем с первого символа и смотрим на корневой узел, чтобы определить вероятность возникновения данного символа. Затем продолжаем просматривать последовательность, добавляя следующий символ к предшествующему, получая таким образом новую метку.

Далее необходимо найти узел, основанный на вновь созданной метке, исключая последний символ. Например, если мы пытаемся найти "1 3 2", мы пытаемся найти узел "1 3", поскольку он содержит вероятность "2", возникающую после этого. Если узел не существует, мы удаляем один символ с внешней стороны узловой метки до тех пор, пока мы не столкнемся с узлом, который существует в дереве.

Этот процесс назван "backtracking", поскольку происходит перемещение по дереву к корню, пока мы не будет найден правильный узел для того, чтобы продолжать процесс сравнения. Отметим, что в наихудшем случае необходимо возвращаться в корень, поскольку он должен содержать вероятность следующего символа для того, чтобы приступать к алгоритму сравнения.

Суммарная вероятность сравнения получается путем умножения вероятности в каждой точке сравнения на текущую суммарную вероятность. Например, если мы хотим сравнить пример "1 3 2" с PST для последовательности «1 2 3 1 2 3 2 1 3», показанный на рис. 2, должны быть выполнены следующие шаги.

Шаг 1: Корневой узел дает вероятность 3/8 при обнаружении "1" .

Шаг 2: узел "1" дает условную вероятность 1/3 при обнаружении "3" , так что наша совокупная вероятность теперь 3/24.

Шаг 3: узел "1 3" не существует, поэтому мы удаляем символ с внешней стороны, которая дает "3". Узел "3" дает вероятность 1/2 при обнаружении "2". Совокупная вероятность сопоставления для образца - 3/48.

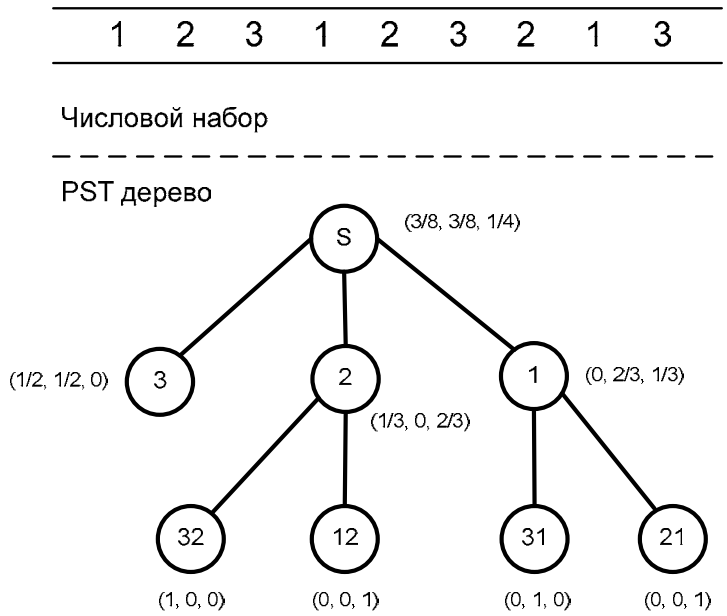

Рис. 2. Пример для иллюстрации сравнения PST

Чем выше вероятность сравнения, тем более вероятно, что данный образец был использован при построении суффиксного дерева.

Например, при сравнении "1 2" с PST на рис. 2, вероятность сравнения – 2/8. Это может быть проверено, поскольку из возможных восьми случаев "1 2" возникает в обучающих данных дважды.

При сравнении "1 2" – более вероятно использовано для того, чтобы построить суффиксное дерево, чем "1 3 2" с совокупной вероятностью 3/48.

Сравнение суффиксных деревьев может быть основано на расстоянии (т.е. значении отличия) между ними. Расстояние между деревьями определено как евклидово расстояние между узлами каждого дерева.

Расстояние между двумя узлами определяется следующим образом:

$$d(T_1, T_2) = \sqrt{\sum_i \sum_k \left(p_1(i)p_1(k|i) - p_2(i)p_2(k|i)\right)^2} \quad , \qquad (1)$$

где p(i) – сравнительная вероятность для узла i.

p(k|i) – условная вероятность появления символа k.

T_1 и T_2 – сравниваемые PST.

Расстояние равно 0 тогда и только тогда, когда два узла идентичны.

Есть два типа сравнения: семантическое и потоковое. Семантическое сравнение включает только те узлы, которые существуют в обоих деревьях. В зависимости от обучающих данных одно дерево может содержать больше узлов, чем вто-

рое, что делает процесс сравнения более сложным. Для того чтобы решить эту проблему, нужно учесть, что если данный узел не существует в дереве, то оно может иметь ту же вероятность, что и родитель.

Можно затопить дерево ненужными узлами для того, чтобы выполнить сравнение путем создания такого количества дочерних узлов, которое необходимо. При этом копируются распределения вероятностей, принадлежащие родительскому узлу, существующему в обоих деревьях.

Например, если «1_3» существует в первом дереве, но не существует во втором, мы можем «залить» общий суффиксный узел, «3», во втором дереве путем копирования распределения вероятностей в новый узел, маркированный «1_3». Теперь, когда оба дерева содержат «1_3», операция сравнения может быть успешно применена.

Запишем алгоритм на псевдоязыке:

Сравнение ПСТ

результат = Рекурсивное сравнение ПСТ (корень1, корень 2, точное_сравнение) +

Рекурсивное сравнение ПСТ (корень1, корень 2, неточное_сравнение) +

Рекурсивное сравнение ПСТ (корень2, корень 1, неточное_сравнение)

Рекурсивное Сравнение ПСТ

```
Если узел1=пустой_указатель, выход
Если (текущий_узел->глубина=0)
  Если точное_сравнение
    Для I от 0 до Макс_число_знаков_алфавита
      Результат+=(узел1->вероятность(i)- узел2->вероятность(i))^2
Иначе
T = найти_ближайший_узел
Если (T->уровень = узел1->уровень) искл. Или (неточное_сравнение)
    Для I от 0 до Макс_число_знаков_алфавита
      Результат+=(узел1->вероятность(i)- T->вероятность(i))^2
```

Результат += рек. Сравнение ПСТ (узел1->сын, узел2, точность сравнения) + рек. Сравнение ПСТ (узел1->брат, узел2, точность сравнения)

Вернуть результат.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *D. Ron, Y. Singer, and N. Tishby.* The power of amnesia: Learning probabilisitc automata with variable memory length. Machine Learning, 25:117–142, 1996.

2. *M. Schonlau, W. DuMouchel, W. Ju, A. Karr, M. Theus, and Y. Vardi.* Computer intrusion: Detecting masquerades. Statistical Science, 16:1–17, 2001.

Абрамов Георгий Эдуардович
Технологический институт Федерального государственного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: dashnak@rambler.ru.
347928, г. Таганрог, пер. Некрасовский, 44.
Тел. 8 (8634) 371-905.
Кафедра безопасности информационных технологий; аспирант.

Abramov George Eduardovich
Taganrog Institute of Technology – Federal State-Owned Educational Establishment of Higher Vocational Education “Southern Federal University”.
E-mail: dashnak@rambler.ru.

44, Nekrasovskiy, Taganrog, 347928, Russia.
Phone: 8 (8634) 371-905.
The Department of IT Security; post-graduate student.

УДК 681.324

Е.С. Абрамов, И.Д. Сидоров

МЕТОД ОБНАРУЖЕНИЯ РАСПРЕДЕЛЁННЫХ ИНФОРМАЦИОННЫХ ВОЗДЕЙСТВИЙ НА ОСНОВЕ ГИБРИДНОЙ НЕЙРОННОЙ СЕТИ

Описывается метод применения вероятностных суффиксных деревьев для обнаружения аномального поведения программ. Используется «отпечаток» нормального поведения приложений с целью в дальнейшем обнаружить аномальное поведение как нечто, отклоняющееся от модели. В качестве основной модели используется вероятностные суффиксные деревья.

Вероятностное суффиксное дерево; PST; обнаружение аномального поведения.

E.S. Abramov, I.D. Sidorov

METHOD OF DETECTION OF DISTRIBUTED INFORMATION IMPACTS ON THE BASIS OF HYBRID NEURAL NETWORK

Described the method of probabilistic suffix trees for detecting anomalous behavior of programs. Use "fingerprint" of the normal behavior of applications in order to further detect anomalous behavior as something deviating from the model. As a basic model uses a probabilistic suffix trees.

Probabilistic suffix tree; PST; detection of abnormal behavior.

Введение

Большинство современных систем обнаружения атак (СОА) осуществляют обнаружение атак путём контроля профилей поведения либо поиска специфических строковых сигнатур. Используя эти методы, практически невозможно создать полную базу данных, содержащую сигнатуры большинства атак. Существует три главные причины этого:

1. Новые сигнатуры необходимо создавать вручную. Сигнатуры известных атак, которые уже включены в БД, не могут гарантировать надёжной защиты без постоянных обновлений.

2. Теоретически существует бесконечное число методов и вариантов атак, и для их обнаружения понадобится БД бесконечного размера. Таким образом, имеется возможность того, что некая атака, не включённая в базу данных, может быть успешно осуществлена.

3. Современные методы обнаружения вызывают большое число ложных тревог. Таким образом, могут быть скомпрометированы легальные сетевые события.

Основное преимущество систем обнаружения атак, использующих нейронные сети, в том, что нейросеть не ограничена знаниями, которые заложил в неё программист. Они имеют возможность учиться на предшествующих событиях –