

# Идентификация сканеров в IP-сетях статистическим методом

Бредихин С. В., Костин В. И., Щербакова Н. Г.

Институт вычислительной математики и математической геофизики СО РАН

## 1. Введение

Известно, что сети, построенные на IP-протоколах, регулярно подвергаются различным процедурам зондирования с целью определения текущего состояния сетевых устройств. В результате выполнения этих процедур определяются адреса устройств, доступных в изучаемой зоне, протоколы или сервисы, которые они поддерживают. Зондирование проводится с разными целями. Например: поисковые системы изучают информационные серверы с целью обнаружения нового контента; р2р клиенты находят своих коллег; системные администраторы проводят ревизию сети с целью своевременного обнаружения уязвимых компьютеров и принятия превентивных мер защиты.

Ничего предосудительного в этих исследованиях нет до тех пор, пока информация, полученная в результате зондирования, не попадает к злоумышленникам. Они ищут незащищенные компьютеры для несанкционированного размещения на них своих, как правило, вредоносных, программ. Для поиска подходящих компьютеров используется «сетевая разведка», в результате проведения которой можно получить не афишируемую информацию о составе сетевых устройств и текущие характеристики их настроек.

Одним из видов сетевой разведки является сканирование доступных ip-адресов и (или) tcp/udp-портов [1]. Для этого используются специальные программы, которые, например, могут определять, по каким tcp/udp-портам устройства исследуемой зоны настроены на «слушание» (от англ. listening). Эти порты являются уязвимым точками сети, поскольку, как правило, через них осуществляется неконтролируемый доступ – вторжение. Такой тип сканирования далее будем называть tcp/udp сканированием.

В связи с этой угрозой, системным администраторам желательно знать, как действуют сканеры и каковы их сильные и слабые места, чтобы если не предотвратить, то зафиксировать попытки сканирования и, возможно, вторжение. Последствия вторжения могут быть весьма неприятными, например, заражение компьютера и приведение его в состояние «зомби» или атака сетевого устройства путем организации непрерывного потока запросов, что может блокировать его работоспособность.

Далее, под сканером мы будем понимать сетевой компьютер (хост), на котором функционирует какая-либо программа сканирования. Вопрос о том, как корректно отличить нормальное поведение активного хоста от хоста, осуществляющего вторже-

ние, является фундаментальной проблемой любой системы обнаружения сетевых аномалий.

На этапе проектирования алгоритма для обнаружения процесса сканирования возникает ряд трудностей. Во-первых, нет четкого определения активности. Например, считать ли сканером активное устройство, делающее попытки установить tcp-соединения с небольшим количеством сетевых устройств, если известно, что часть попыток unsuccessful. Во-вторых, не определено, какая степень детализации объекта сканирования требуется. Например, это множество портов одного локального адреса или это один порт множества смежных адресов. В-третьих, априори неясно, сколько времени требуется наблюдать за работой хоста, прежде чем принять решение, является ли он сканером. И, наконец, что считать враждебным сканированием. В конечном счете, считать ли процесс сканирования безопасным или нет, решать сетевым администраторам.

## 2. Базовые системы NIDS

В состав системы защиты сети от несанкционированного доступа (NIDS – аббревиатура от Network Intrusion Detection System), как правило, входят подсистемы, выявляющие процедуры враждебного сканирования. Программы этих подсистем анализируют поступающий в сеть трафик. Методы, заложенные в основу алгоритмов анализа поведения сетевых устройств, можно условно разделить на две основные группы. Первая группа использует метрику –  $N$  подозрительных событий за время  $T$ , так называемые базовые методы, вторая использует методы теории вероятностей и математической статистики.

Среди систем NIDS лидируют две – Snort (M. Roesch, 1998) и Bro (V. Paxson, 1998). В состав обеих входят подсистемы обнаружения сканеров. Исследуемым материалом является «сырой» трафик, собираемый с помощью функций библиотеки libpcap. Далее будем использовать следующие обозначения: протокол – *protocol*, адрес источника – *src\_ip*, адрес получателя – *dst\_ip*, порт источника – *src\_port*, порт получателя – *dst\_port*.

**2.1. Система Snort** [2] является de-facto стандартной системой для обнаружения несанкционированного вторжения. Snort может в реальном времени производить анализ трафика и протоколов, чтобы предотвратить атаки различных типов. Предварительно в Snort закладываются знания об активных устройствах исследуемой сети, т.е. заранее формируется множество адресов сетевых устройств, являющихся легальными, активными

источниками и получателями данных. Например, такими устройствами являются dns, проху и web серверы. Эти знания позволяют ускорить работу системы и сократить число ложных результатов. Snort использует специальный язык, определяющий набор правил для анализа трафика.

В основу подсистемы выявления сканирования заложено предположение, что сканер не имеет информации о хостах сканируемой сети, поэтому большинство его обращений к устройствам сети будет неуспешным. Далее мы будем рассматривать только tcp/udp сканирование. Выявление сканеров производится на уровне препроцессоров. Препроцессор flow, получая сырой трафик, собираемый с помощью функций библиотеки libpcap (Unix), отслеживает неправильно сформированные пакеты, используемые при так называемом «скрытом сканировании», и формирует потоки записей на основе пяти полей IP пакетов {protocol, src\_ip, src\_port, dst\_ip, dst\_port}. Препроцессор следующего уровня анализирует потоки на предмет выявления сканирования и вырабатывает сигналы тревоги. В первых версиях системы препроцессор portscan посылал сигнал тревоги, если

– хост с адресом *src\_ip* совершал *N* попыток установить соединение с хостами защищаемой сети, используя при этом *P* различных портов за *T* секунд в случае использования *tcp* протокола;

– хост с адресом *scr\_ip* совершил *N* попыток контактировать с хостами защищаемой сети, используя при этом *P* различных портов за *T* секунд в случае использования *udp* протокола.

В обоих случаях это могут быть попытки типа: один *src\_ip* -> одному *dst\_ip* или один *src\_ip* -> многим *dst\_ips*. Пороги *N*, *P* и *T* являются конфигурируемыми параметрами. В более поздних версиях системы (текущая версия 2.8.2) для обнаружения сканирования на втором этапе используется препроцессор *sfportscan*. Следует заметить, что в Snort заложена возможность исключать штатные и включать пользовательские препроцессоры.

Препроцессор *sfportscan* определяет следующие типы tcp/udp сканирования:

– *portscan one->one* – хост с адресом *src\_ip* активно обращается к разным портам хоста *dst\_ip*;

– *decoy one->one* – один хост активно обращается к разным портам хоста *dst\_ip*, используя при этом в качестве адреса источника собственный адрес *src\_ip* и вымышленный (*spoofed*);

– *distributed portscan* – хосты с разными адресами *src\_ips* обращаются к разным портам одного хоста с адресом *dst\_ip*;

– *portsweep* – хост с адресом *src\_ip* обращается к разным хостам с адресами *dst\_ips*, используя один и тот же порт назначения.

Пользователь может задать один из уровней защиты: «low», «medium» и «high». Если выбран уровень low, то для *src\_ip* рассматриваются только неудачные попытки (*N*) контактировать с хостами защищаемой сети за определенный интервал *T*. По прошествии интервала *T* посылаются сообще-

ние, и счетчик сбрасывается. Этот уровень почти не дает ложных результатов. Если выбран уровень medium, то рассматриваются все попытки контактировать с хостами защищаемой сети за время *T*. При этом по прошествии интервала *T* также посылается сообщение, и счетчик сбрасывается. Если выбран уровень high, то счетчики ведутся непрерывно, а сообщения посылаются по прошествии *T*. Таким способом выявляется медленное сканирование.

Во время работы препроцессор ведет счетчики, значения которых приводятся в сообщениях. *Priority Count* – сколько плохих ответов (отказ или нет ответа) получено данным *src\_ip*. *Connection Count* – счетчик активных соединений для данного хоста, ведется для адресов источников и адресов назначения. Эта мера информативна для tcp-соединений. *IP\_Count* – счетчик числа хостов с адресами *src\_ips*, которые пытались контактировать с данным *dst\_ip*; значение счетчика может быть небольшим для сканирования тира one->one и может быть большим для активных хостов и распределенного сканирования. *Port\_Count* – счетчик портов *dst\_ports*, по которым производились попытки контактировать с хостом *dst\_ip*; этот счетчик, совместно с *IP\_Count* используется для различения сканирования one->one и decoy one->one. *Scanned/Scanner IP Range* – в случае сканирования типа *portsweep* (один к многим) сохраняется диапазон сканируемых адресов, в случае *portscan* (один к одному) диапазон адресов сканеров.

Пользователи имеют возможность настроить механизм обнаружения сканеров применительно к своей сети (сетям) следующим образом:

- (а) определять множество адресов, за которыми следить; множество адресов, которые не считать сканерами; множество адресов, которые не считать сканируемыми. Это можно сделать как априори, так и в результате наблюдения за сигналами тревоги.
- (б) Следить за соотношениями *Connection Count / IP Count*, *Port Count / IP Count*, *Connection Count / Port Count* для выявления ложных сигналов тревоги. Соотношение *Connection Count / IP Count* должно быть как можно более высоким для сканирования типа *portscan* и низким для сканирования типа *portsweep*. Соотношение *Port Count / IP Count* должно быть высоким для *portscan* и низким для *portsweep*. Соотношение *Connection Count / Port Count* должно быть низким для *portscan* и высоким для *portsweep*. Большое значение счетчика *Priority Count* является хорошим показателем как для *portscan* так и для *portsweep*, если только не используется экранирование.
- (в) Если это не помогает, снизить уровень защиты.

Таким образом, метод выявления сканирования, используемый системой Snort, относится к базово-

му типу. При создании анализатора сканирования авторы ориентировались на выявление всех вариантов сканирования, определенных в классической утилите сканирования Nmap [3].

**2.2. Система Bro** [4] – это открытая автономная система пассивного мониторинга, предназначенная для выявления подозрительной активности. Данные о трафике собираются с помощью функций библиотеки libpcap. Концептуально Bro делится на два уровня.

На первом уровне, например, происходит фильтрация плохо сформированных пакетов, определение, для каких протоколов прикладного уровня передавать все пакеты на следующий уровень для проведения тщательного анализа, а для каких собирать только информацию о tcp/udp сессиях. Обработка tcp/udp пакетов происходит следующим образом. Если встречается запрос на установление tcp-соединения, то, если по прошествии таймера  $T$  не было получено ответа, соединение будет иметь статус *connection\_attempt*. Если в ответ получено подтверждение, то статус будет *connection\_established*, который может измениться на *connection\_finished*, если сессия закончится в рассматриваемый период. Если получен отказ, то статус будет *connection\_rejected*. Сохраняются времена начала и конца сессии, количество переданных и полученных байтов. В случае udp, если хост с адресом *src\_ip* посылает запрос устройству с адресом *dst\_ip*, используя при этом порт источника *src\_port* и порт назначения *dst\_port*, то считается, что устанавливается псевдо-соединение, которое будет иметь статус «udp\_reply», если от *dst\_ip* будет получен соответствующий ответ, иначе статус будет «udp\_request». Результаты работы первого уровня передаются на следующий уровень, в который закладывается политика обеспечения безопасности. Мы не будем останавливаться на способах взаимодействия уровней и языке задания правил безопасности.

Подсистема выявления сканирования, Scan Analyzer, функционирует на втором уровне. В подсистему, также как и в случае Snort, могут закладываться знания о легально активных хостах и адресах назначения, к которым происходит многочисленные обращения. Система Bro различает два типа сканирования: *сканирование адресов* и *сканирование портов*. Для каждого типа сканирования определяется свой набор порогов активности, и если величина активности превышает один из этих порогов, то система генерирует соответствующее этой величине сообщение, например, предупреждение или сигнал тревоги. Scan Analyzer выявляет кроме того, попытки взломать сервер с помощью подбора имени пользователя и пароля, на которых мы не будем останавливаться.

Сканирование адресов: удаленный хост с адресом *src\_ip* пытается установить tcp-соединения с определенным количеством адресов назначения или, в случае udp протокола, пытается контактировать с определенным количеством адресов на-

значения. В общем случае не прослеживается, чем закончилась попытка установить соединение; для выделенных протоколов, таких как ftp и telnet, определенных на первом уровне, внимание уделяется неудачным попыткам. Для выявления сканирования IP адресов ведутся две таблицы: *distinct\_peers [src\_ip, dst\_ip]*, где сохраняется информация обо всех различных парах взаимодействующих адресов, если появилась новая пара, то увеличивается счетчик в таблице *addr\_count[src\_ip]*, где ведется учет, со сколькими различными адресами взаимодействовало устройство с данным *src\_ip*. Если это количество превысило один из заданных порогов, посылается уведомление. По умолчанию используется шкала порогов {100, 1000, 10000}, т.е. уведомление посылается по превышению 100, затем по превышению 1000 и т.д. Информация собирается и для сканирования, инициированного локальными устройствами. Для них определяется своя шкала порогов, по умолчанию {1000, 10000}.

Сканирование портов: для каждого *src\_ip* ведется таблица *distinct\_ports[src\_ip, dst\_port]*, где сохраняется информация о том, по каким портам назначения *dst\_port* контактировал *src\_ip*. Вторая таблица *port\_count[src\_ip]* хранит информацию о количестве портов назначения для данного *src\_ip*. Если это количество превысило один из заданных порогов, посылается уведомление. По умолчанию порог равен 25. Заметим, что информация собирается безотносительно к адресу назначения. Если порог превышен, Scan Analyzer начинает собирать информацию, к какому адресу назначения относится порт. Эта информация позволяет уменьшить количество устройств, ложно принятых за сканеры, однако это дорогостоящая мера.

В подсистеме выявления сканирования намеренно не учитывается фактор времени, чтобы не пропустить «тонкие» сканеры, например такие, которые в течение короткого промежутка времени работают интенсивно, а затем подолгу не проявляют активность.

Заметим, что на текущий момент Scan Analyzer не способен выявлять распределенное сканирование. Пороги, адреса назначения и адреса источников, которые следует игнорировать при выявлении сканирования, задаются при настройке Scan Analyzer с помощью инициализации переменных, *scan variables*. *Scan functions* определяют, что следует предпринять, если обнаружено сканирование со стороны *src\_ip*. Например, если при настройке указано, что можно рвать коннективность и *src\_ip* не попал в число тех, с кем нельзя ее прерывать никогда, то она будет прервана. Система Bro позволяет принимать кардинальные меры борьбы с злоумышленниками.

Таким образом, методы выявления сканирования, используемые в Bro, также как и в Snort, можно отнести к базовым. Хотя Bro выявляет не все виды сканирования, является ресурсоемким и основывается на статических порогах, наличие шкалы порогов и отсутствие временных рамок де-

лают его достаточно привлекательным. Базовые методы показывают хорошие результаты в случае массированного, методичного сканирования. Однако выбор порога является камнем преткновения в базовых методах.

### 3. Расширение базовых систем и вероятностные подходы

Базовые методы имеют недостатки. Трудно правильно выбрать порог, а от этого зависит число устройств, ложно принятых за сканеры, и число не выявленных сканеров. Также трудно выявить сканеры, которые посылают запросы в случайном темпе или используют большие временные задержки в своей работе. Такое поведение невозможно выявить с помощью метрики  $N/T$ . Возникает необходимость в более изощренных методах выявления аномального поведения сетевых устройств.

В рассматриваемых нами примерах предполагается, что известно «нормальное» состояние сети, т.е. распределение вероятностей обращений к активным сетевым устройствам и сервисам. Например, вероятности обращений к адресу  $P(dst\_ip)$ , порту  $P(dst\_port)$  или совместная вероятность  $P(dst\_ip, dst\_port)$ . Тогда, если устройство с адресом  $src\_ip$  обращается к какому-либо устройству с адресом  $dst\_ip$  и использует в качестве порта назначения  $dst\_port$ , а  $P(dst\_ip, dst\_port)$  мала, то устройство попадает в число подозреваемых. Основная трудность состоит в том, какие именно вероятности, условные вероятности и совместные вероятности хранить, тем более что нужно собирать информацию в течение длительного периода времени. Рассмотрим работы, в которых предлагаются различные подходы к решению этой задачи.

**3.1.** Авторы [5] особое внимание уделяют изучению медленных сканеров, к которым не применимы базовые методы. Вводится основополагающее понятие *footprint* – количество различных комбинаций  $dst\_ip/dst\_port$ , т.е. след, который оставляет источник. Здесь и далее  $параметр_1/параметр_2$  означает, что  $параметр_2$  фиксирован, а  $параметр_1$  меняется. Авторы предполагают, что сканер ищет конкретные сервисы, предоставляемые хостами данной сети, т.е. сканируется большое количество адресов по небольшому количеству портов. Самый простой подход (первая мера) состоит в подсчете числа различных комбинаций, используемых источником.

Однако сканеры зачастую интересуются портами, открытыми на определенных хостах, по этим данным они, например, идентифицируют операционную систему исследуемого хоста. Поэтому необходимо следить и за комбинациями  $dsp\_port/dst\_ip$  (вторая мера), т.е. иметь полную информацию о деятельности источника. В качестве третьей меры авторы предлагают рассматривать количество обращений к закрытым портам, т.е. рассматривать комбинации  $dsp\_port/dst\_ip$ , где порты не поддерживаются  $dst\_ip$  или не доступны для  $src\_ip$ . И,

наконец, рассматривается метрика, которая обобщает идею наблюдения за закрытыми портами.

Предполагается, что априори известно распределение нормального трафика защищаемой сети по хостам и портам, т.е. если в пакете встречается данная комбинация  $dst\_port/dst\_ip$ , обозначим ее  $x$ , то известна вероятность ее появления  $P(x)$ . Определим индекс аномальности пакета (или события), содержащего пару  $x$  через отрицательный логарифм правдоподобия:  $A(x) = -\log(P(x))$ . Индекс аномальности множества  $X = x_1, x_2, \dots$  определим как  $A(X) = \sum_{x \in X} A(x)$ . Таким образом, чем больше необычных комбинаций использует сканер, тем быстрее он будет обнаружен. Заметим, что здесь игнорируется тот факт, что распределение вероятностей зависит от времени суток.

Понятие индекса аномальности события используется в модели Spice [5], состоящей из двух компонент: сенсора и коррелятора. В задачу сенсора входит определение индекса аномальности наблюдаемых событий. События с высоким значением этого индекса передаются коррелятору, который группирует их и генерирует отчеты об обнаруженных аномалиях.

Индекс аномальности события определяется на основе таблиц вероятности отдельных элементов, условных и совместных вероятностей. Например, для каждой четверки  $(dst\_port, dst\_ip, src\_port, src\_ip)$  вычисляется вероятность ее появления и сохраняется в таблице. Однако такая таблица будет слишком объемной, кроме того, если процент нормального трафика высок, то будет много ложных тревог. Поэтому можно рассматривать вероятности различных комбинаций этих параметров, например,  $P(dst\_port)$ ,  $P(src\_port, dst\_port)$ ,  $P(dst\_ip, src\_ip, src\_port)$ . Для определения независимости между случайными величинами предлагается использовать графическую модель – байесовскую сеть [6].

Коррелятор объединяет события в группы. Функция, определяющая насколько связаны между собой события  $e_1$  и  $e_2$  имеет вид:

$$f(e_1, e_2) = c_1 h_1(e_1, e_2) + c_2 h_2(e_1, e_2) + \dots + c_k h_k(e_1, e_2),$$

где  $c_1, \dots, c_k$  – константы, а  $h_1, \dots, h_k$  – эвристические функции, отражающие, каким образом события объединяются между собой при сканировании, они строятся на основании знаний о поведении сканеров. Например, проверяется, используется ли один и тот же адрес источника, порт назначения, сеть назначения; как соотносятся времена событий, адреса назначения, порты назначения; как соотносятся ближайшие события; как растет значение порта источника по сравнению с ростом адреса назначения и т.д. Сильно связанные между собой события (значение функции  $f$ ) превосходит заданный порог) объединяются в подгруппы. Слабо связанные между собой группы не рассматриваются.

Работу коррелятора можно представить как построение графа, в вершинах которого находятся события, а ребра отображают связи между ними.

Для того, чтобы добавить новое событие (вершину) к графу, требуется просчитать, насколько связано это событие с каждым из уже имеющихся узлов. Поскольку число узлов может оказаться большим, то для выполнения этой операции используется метод «имитации отжига» [7]. Индекс аномальности группы событий – это сумма индексов событий. Если сумма превысила некоторый порог, то посылается отчет о группе аномальных событий.

Таким образом, сенсор принимает решение на уровне пакетов, а коррелятор объединяет пакеты в группы. Частично это можно сделать с помощью генерации потоков (*netflow*), в которых пакеты агрегируются в группы на основе содержимого полей пакета. Однако не все связи между событиями можно определить с помощью потоков, например, распределенное сканирование.

Модель *Spice* [5] была частично реализована в системе SPADE (*Statistical Packet Anomaly Detection Engine*). Рассматривались только tcp пакеты с флагом SYN, как считающиеся наиболее информативными. Препроцессор *flow*, входящий в *Snort* и включенный в эту систему, преобразует сырой трафик и формирует структуры данных, содержащие нужные поля. SPADE обеспечивает возможность выбора, на основании каких данных будет определяться аномальность пакетов. Это либо одна из совместных вероятностей  $P(src\_ip, src\_port, dst\_ip, dst\_port)$ ,  $P(src\_ip, dst\_ip, dst\_port)$ ,  $P(dst\_ip, dst\_port)$ , либо байесовская сеть, определяющая, как связаны между собой параметры  $src\_ip$ ,  $src\_port$ ,  $dst\_ip$ ,  $dst\_port$ . Поскольку выбор порога является узким местом методики, SPADE предоставляет возможность регулирования порога в зависимости от нагрузки сети. Кроме того, отчеты об аномальности могут поступать как на основе периодичности, так и на основе некоторых статистических характеристик.

Для определения качества используемой методики введены две метрики: производительность (*efficiency*) и эффективность (*effectiveness*). Производительность – это отношение количества выявленных эталонных сканеров (*true positives*), к количеству адресов источников, определенных алгоритмом как сканеры (*all positives*). Эффективность – отношение количества выявленных эталонных сканеров (*true positives*) к количеству эталонных сканеров (*true positives*). Отметим, что определенные здесь метрики качества алгоритма используются в работах других авторов, например, в [8].

Для проверки работы SPADE использовался специально подготовленный блок данных трафика, в котором были выявлены эталонные сканеры. Затем результаты SPADE были проанализированы с использованием различных значений параметров: порогов и вероятностных мер. Было выявлено, что между производительностью и эффективностью существует обратная связь: если повышается эффективность, то понижается производительность. Результаты показали, что методика дает хорошие результаты при правильно выбранных параметрах.

Следует заметить, что низкая производительность приемлема для SPADE, при условии, что пакеты будут переданы на следующий уровень, который выявит часть ложных тревог.

Использование двухступенчатой структуры в модели *Spice* представляется интересным подходом, позволяющим производить тщательный анализ данных. Однако *Spice* предполагает достаточно большой объем предварительной работы по объединению событий в группы.

3.2. В работе [9] предлагается совместить возможности базовых методов и метода определения аномальности, предложенного в [5]. В базовом варианте для выявления сканирования учитывается весь трафик, инициированный устройством с адресом  $src\_ip$  за определенный период времени, а в [5] аномальность определяется на основе обращения к редко используемым комбинациям  $dst\_ip/dst\_port$ . Однако если сканер действительно зачастую использует редкие пары, то обратное неверно. В результате, во-первых, небольшое количество обращений к редко используемой паре приводит к тому, что источник определяется как сканер, во вторых, обращение к часто используемой паре будет иметь большую степень правдоподобия, и сканер будет пропущен.

Рассматриваются две новые характеристики, отличающие поведение сканера от поведения обычного хоста. Во-первых, если рассмотреть взаимодействие сканера с устройством с адресом  $dst\_ip$  и использованием порта назначения  $dst\_port$ , то для сканеров, в отличие от обычных хостов, при этом зачастую не используется более трех пакетов, так как сканеры либо запрашивают неактивный сервис, либо в случае активного намеренно не завершают процедуру установления соединения. Поэтому предлагается рассматривать только входящие потоки, содержащие менее 4 пакетов. Во вторых, большинство сканеров последовательно перебирает адреса одной и той же подсети, в отличие от нормальных хостов, обычно использующих случайный набор адресов назначения при поиске. Поэтому можно рассматривать сканирование не отдельных адресов, а подсетей. Предлагается вести счетчик для адреса источника  $src\_ip$ , использующего комбинацию  $dst\_ip/dst\_port$  в рамках подсети (сканирование подсетей).

Кроме использования эвристических предположений предлагается использование так называемой «usage based» схемы, подобной используемой в [5]. Отслеживается история контактов для адреса источника  $src\_ip$  в рамках временного интервала, но оценивается каждое взаимодействие с уникальным адресом  $dst\_ip$  по порту  $dst\_port$  по-разному. Для устройства с адресом  $src\_ip$  рассматриваем число обращений к некоторому порту  $dst\_port$ . Если устройство с адресом  $src\_ip$  впервые контактирует с устройством с адресом назначения  $dst\_ip$  с использованием данного порта назначения, то к индексу аномальности  $ScanScore_{src\_ip}$  прибавляется не единица, как в базовом методе, а

$1/1 + \lg(\text{count}_{dst\_ip/dst\_port})$ , где  $\text{count}$  – число различных источников, обращавшихся к этой паре. Таким образом, учитывается, часто ли используется такая пара, например, обращение к веб-сайтам, которые используются многими, будет иметь меньший вес, чем редко используемая случайная комбинация. Кроме того, в данном методе, наряду с понятием *time window* (время, в течение которого ведется наблюдение), используется понятие *connection window*, т.е. сохраняется информация об  $N$  контактах для каждого источника, это позволяет правильно оценивать сканеры с низкой скоростью сканирования. Хост с адресом  $src\_ip$  объявляется сканером, если суммарный индекс аномальности  $\text{ScanScore}_{src\_ip}$  за время *time window* или *connection window* превосходит заданный порог.

Проведение экспериментов выявило, что предлагаемые подходы позволяют улучшить характеристики по сравнению с классическим базовым подходом. Обозначим через  $B_k$  базовый подход, через  $H_k$  – подход с использованием обоих эвристических предположений ( $k$  – порог, пороги одинаковые при двух подходах) и через  $U$  – «usage based» (здесь порог свой). Если классический базовый метод  $Y$  событий за время *time-window* расширился возможностью следить за  $N = \text{connection-window}$  адресами назначения, то базовый метод показывал достаточно хорошие результаты. Обозначим через  $N\_trues$  количество выявленных сканеров, а через  $N\_false$  количество ложно принятых за сканеры. Использование эвристического метода  $H_k$  несколько уменьшило  $N\_trues$ , но значительно сократило  $N\_false$ . Использование метода  $U$  значительно увеличило  $N\_trues$ , оно самое высокое, при этом  $N\_false$  стало немного выше, чем при использовании  $H_k$ , но ниже, чем при базовом методе. Таким образом, расширение базового метода позволило значительно снизить количество ложных тревог.

**3.3.** В работе [10] для выявления сканирования рассматривается поток пакетов между защищаемой сетью и внешним миром. В процессе анализа трафика строятся две таблицы. В одной таблице собираются все пары  $(src\_ip, dst\_ip)$ , в другой – все пары  $(src\_ip, dst\_port)$ . Эти таблицы используются для определения необычности распределения использования адресов или портов назначения. С вероятностной точки зрения ставится цель выявить, у какого источника распределение  $P(dst\_ip|src\_ip)$  значительно отличается от  $P(dst\_ip)$  для всех источников. Аналогично, распределение  $P(dst\_port|src\_ip)$  отличается от общей вероятности  $P(dst\_port)$ .

Пусть  $D$  – множество адресов назначения защищаемой сети, с которыми был контакт извне, а  $D_i \in D$  – множество адресов, с которыми контактировал  $src\_ip_i$ . Соответственно, рассматриваются все множество портов  $P$  и множество портов  $P_i$ , используемых  $src\_ip_i$ . Пусть  $n_s(dst\_ip_j)$  – количество источников, которые контактировали с  $dst\_ip_j$  в рассматриваемой выборке. Определим априорную вероятность нормального контакта с  $dst\_ip_j$ :

$$P_{norm}(dst\_ip_j) = n_s(dst\_ip_j) / \sum_{di \in D} n_s(dst\_ip_i),$$

здесь суммирование производится по всем адресам назначения, с которыми был контакт.

В противоположность этому для атакующего устройства, которое с равной вероятностью может обращаться к любому адресу назначения, определяем вероятность:

$$P_{att}(dst\_ip_j) = 1/|D|.$$

Аналогично определяются вероятности  $P_{norm}(dst\_port_j)$  и  $P_{att}(dst\_port_j)$ .

В предположении, что достижение каждого адреса является независимым событием, определяются вероятности контакта с множеством адресов назначения  $D_i$  для нормального хоста и атакующего  $P_{norm}(D_i)$  и  $P_{att}(D_i)$ . В общем виде формулы выглядят так:

$$P(D_i) = P(D_i = \{dst\_ip_1, dst\_ip_2, \dots\}) * P(|D_i|).$$

Здесь вероятность контактировать с конкретным множеством адресов умножается на вероятность  $P(|D_i|)$  контактировать с множеством адресов данной мощности. Это делается для того, чтобы внести дополнительную характеристику, в предположении, что атакующий обращается к большому количеству адресов/портов в отличие от нормального хоста. В свою очередь,

$$P(D_i = \{dst\_ip_1, dst\_ip_2, \dots\})$$

определяется через произведение вероятностей контакта с каждым отдельным адресом. Мы не рассматриваем вопрос, насколько достоверны априорные вероятности и не будем приводить точные формулы.

Если для некоторого адреса источника  $src\_ip$  зафиксировано множество адресов назначения  $D_i$ , такое что  $P_{att}(D_i) > P_{norm}(D_i)$ , адрес объявляется сканером. На практике вычисляются отрицательные логарифмы правдоподобия:

$$L_{norm}(D_i) = -\ln(P_{norm}(D_i)); L_{att}(D_i) = -\ln(P_{att}(D_i)).$$

И, если для  $src\_ip$  верно неравенство  $L_{norm}(D_i) - L_{att}(D_i) > 0$ , то устройство с адресом  $src\_ip$  объявляется сканером.

В предложенном методе учитывается как нестандартное использование адресов или портов, так и обращения к большому числу адресов и портов. Остается невыясненным, как определять вероятность доступа к устройствам, к которым еще никто не обращался. Более того, нет оценки, достаточно ли различаются, например, вероятности контакта с множеством адресов назначения для нормального и атакующего хостов, чтобы на их основании сделать предположение о том, является ли хост сканером.

#### 4. Метод последовательного анализа

Модели, о которых пойдет речь ниже, основаны на «Последовательном анализе» (ПА) А. Вальда (1902-1950) [11] – способе проверки статистических гипотез, при котором необходимое число наблю-

дений не фиксируется заранее, а определяется в процессе самой проверки. Метод ПА позволяет ограничиться значительно меньшим числом наблюдений, чем при способах, в которых число наблюдений фиксировано заранее. Одно из главных утверждений [11] состоит в том, что процедура ПА в случае двух простых альтернативных гипотез и наличии однородной независимой выборки оптимальна по числу шагов.

Оставляя в силе вышесказанное предположение, сформулируем суть метода ПА. Пусть  $X$  – случайная величина, а  $x_1, \dots, x_n, \dots$  – последовательность независимых и одинаково распределенных наблюдений за  $X$ . Допустим, что относительно этого распределения имеется два предположения. Гипотеза  $H_0$  – наблюдения распределены с плотностью  $p_0(x)$ , а гипотеза  $H_1$  – наблюдения распределены с плотностью  $p_1(x)$ . После каждого наблюдения предоставляется выбор из трех возможных решений: принять  $H_0$  и закончить наблюдения, принять  $H_1$  и закончить наблюдения, не принимать ни одну из гипотез и продолжить наблюдения.

Решающая процедура  $\delta^*$  определяется следующим образом. Фиксируются два порога: верхний  $A$  и нижний  $B$ , такие что  $0 < B < 1 < A$ . Пусть выполнено  $n$  наблюдений ( $n=1,2,\dots$ ). Обозначим через  $L_n(x_1, \dots, x_n)$  отношение правдоподобия:

$$L_n(x_1, \dots, x_n) = \prod_{i=1}^n \frac{p_1(x_i)}{p_0(x_i)}.$$

Процедура  $\delta^*$  на шаге  $n$  такова: если  $L_n(x_1, \dots, x_n) \geq A$ , то принимается гипотеза  $H_1$  и процесс наблюдения заканчивается; если  $L_n(x_1, \dots, x_n) \leq B$ , то принимается гипотеза  $H_0$  и процесс наблюдения заканчивается; если  $B < L_n(x_1, \dots, x_n) < A$ , то выполняется еще одно наблюдение, вычисляется новое отношение правдоподобия  $L_{n+1}(x_1, \dots, x_{n+1})$ , для которого вновь применяется  $\delta^*$ .

Эта процедура характеризуется вероятностями ошибок и средними числами наблюдений:

$$\alpha = \alpha(A, B) = P\{\text{принята } H_1 / \text{верна } H_0\} - \text{ошибка первого рода,}$$

$$\beta = \beta(A, B) = P\{\text{принята } H_0 / \text{верна } H_1\} - \text{ошибка второго рода;}$$

$n_0 = n_0(A, B) = E(n / H_0)$ ,  $n_1 = n_1(A, B) = E(n / H_1)$ , где  $n$  – число наблюдений (случайная величина) до принятия окончательного решения, а  $E$  – математическое ожидание. Зададим желаемые вероятности ошибок первого и второго рода  $\alpha_0$  и  $\beta_0$  и воспользуемся тремя результатами из [11].

Во-первых, фактом, что среди всех решающих правил  $\delta'$ , обладающих свойством

$$\alpha(\delta') \leq \alpha_0, \quad \beta(\delta') \leq \beta_0,$$

последовательный критерий отношения вероятностей  $\delta^*$  имеет минимальные средние числа наблюдений:

$$n_0(\delta^*) \leq n_0(\delta'), \quad n_1(\delta^*) \leq n_1(\delta').$$

Во-вторых, вместо  $A$  и  $B$  можно использовать приближенные значения:

$$A' = \frac{1-\beta}{\alpha}, \quad B' = \frac{\beta}{1-\alpha}.$$

В третьих, формулами для вычисления среднего числа наблюдений, которые зависят от параметров  $p_0(x)$ ,  $p_1(x)$ ,  $\alpha$  и  $\beta$ .

Таким образом, задав параметры  $\alpha$ ,  $\beta$ ,  $p_0$  и  $p_1$  можно построить алгоритм принятия той или иной гипотезы за конечное число шагов.

## 5. Модели на базе метода последовательного анализа

Известно несколько моделей (и алгоритмов на их основе), базирующихся на методе ПА, для выявления сканеров в ip-сетях [8, 12]. Исходными данными этих алгоритмов является информация о трафике, зафиксированная в точках наблюдения. На основании анализа этих данных необходимо определить множество ip-адресов сетевых устройств, которые предположительно являются сканерами. Поскольку единой модели поведения сканеров не существует, в каждом случае исследуется своя модель поведения.

Наше внимание привлек подход, представленный в работе [8]. Рассматриваемые в ней модель и алгоритм TRW не только используют эвристические предположения по поводу поведения сканеров, но и применяют метод проверки предположений на основе оптимального метода последовательного анализа [11], связывающего пороги с ошибками первого и второго рода.

### 5.1. Модель TRW

В основе модели TRW (Threshold Random Walk) [8] лежит предположение о том, что сканеры чаще, чем не сканеры, предпринимают попытки установить tcp-соединения с несуществующими сетевыми устройствами (ip-адрес назначения не используется) или попытки запросить несуществующие сервисы (tcp-порт назначения не используется).

Рассматривается фрагмент сети; ip-адреса устройств, входящих во фрагмент, считаются локальными адресами. Исследуются tcp-соединения, проходящие через контрольные точки, отделяющие фрагмент от глобальной сети. При проверке работы соответствующего алгоритма исходными данными являлась предварительно собранная информация о tcp-сессиях, полученная с помощью системы Bro [4], установленной в точках наблюдения. Если рассматривать трафик в реальном времени, то при поступлении первого пакета на установление tcp-соединения нельзя сразу определить, будет ли соединение успешным. Однако

в случае небольшого размера исследуемого фрагмента сети можно хранить данные о том, какие сетевые устройства активны и какие сервисы они обслуживают.

Будем считать, что произошло событие, если удаленный хост с адресом  $src\_ip$  произвел первую с начала времени наблюдения попытку установить tcp-соединение с локальным хостом с адресом  $dst\_ip$ . Попытка считается успешной, если соединение установлено, и неуспешной, если получен отказ или не было получено ответа.

Для каждого удаленного адреса  $src\_ip$  рассматривается последовательность наблюдений за событиями. Результат наблюдения на шаге  $i$  характеризует случайная величина  $Y_i$ , определяемая следующим образом:  $Y_i = 0$ , если попытка соединения успешна, и  $Y_i = 1$ , если попытка соединения неуспешна. Итак, для каждого  $src\_ip$  строится последовательность наблюдений  $Y_1, Y_2, \dots$

Определяются две гипотезы: гипотеза  $H_1$  состоит в том, что устройство с адресом  $src\_ip$  является сканером, а гипотеза  $H_0$  – что устройство с адресом  $src\_ip$  не является сканером.

Анализируя последовательность наблюдений  $Y_1, Y_2, \dots$  необходимо за конечное число шагов сделать заключение, какая из гипотез верна для устройства с адресом  $src\_ip$ .

Предположим, что если гипотеза  $H_j$  верна, то случайные величины  $Y_i | H_j$ ,  $i = 1, 2, \dots$ ,  $j = 0, 1$ , независимы и одинаково распределены. Обозначим через  $P[Y_i | H_j]$  условную вероятность случайной величины  $Y_i$  при верности гипотезы  $H_j$ . Выразим распределение Бернулли для  $Y_i$ :

$$P[Y_i = 0 | H_0] = \theta_0, \quad P[Y_i = 1 | H_0] = 1 - \theta_0, \\ P[Y_i = 0 | H_1] = \theta_1, \quad P[Y_i = 1 | H_1] = 1 - \theta_1.$$

Из предположения о том, что попытка установить соединение более успешна для не сканеров, чем для сканеров, ясно, что  $\theta_0 > \theta_1$ .

Следуя методике ПА, определим отношение правдоподобия для вектора событий  $Y = (Y_1, \dots, Y_n)$ :

$$\Lambda(Y) = \prod_{i=1}^n \frac{P[Y_i | H_1]}{P[Y_i | H_0]}. \quad (1)$$

Механизм принятия решения основан на пошаговом обновлении значения  $\Lambda(Y)$  и сравнении полученного значения  $\Lambda(Y)$  с верхним и нижним порогами  $\eta_1$  и  $\eta_0$  соответственно. Если  $\Lambda(Y) \geq \eta_1$ , то принимается гипотеза  $H_1$ . Если  $\Lambda(Y) \leq \eta_0$ , то принимается гипотеза  $H_0$ . Если  $\eta_0 < \Lambda(Y) < \eta_1$ , то продолжаем наблюдение, т.е. переходим к рассмотрению следующего события. Величина  $\Lambda(Y)$  осуществляет так называемое случайное блуждание между величинами  $\eta_0$  и  $\eta_1$ .

В работе [8] через  $\beta$  обозначена вероятность обнаружения сканера:

$$\beta = P\{\text{принята } H_1 / H_1\}.$$

Соответственно, вероятность ошибки второго рода  $P\{\text{принята } H_0 / H_1\} = 1 - \beta$ .

Тогда неравенства для порогов выглядят следующим образом [11]:

$$\eta_1 \leq \frac{\beta}{\alpha}, \quad \eta_0 \geq \frac{1-\beta}{1-\alpha}.$$

Приблизительную оценку для числа наблюдений  $N$ , требуемых для принятия решения, можно выразить в виде формул:

$$E[N | H_0] = \frac{\alpha \ln \frac{\beta}{\alpha} + (1-\alpha) \ln \frac{1-\beta}{1-\alpha}}{\theta_0 \ln \frac{\theta_1}{\theta_0} + (1-\theta_0) \ln \frac{1-\theta_1}{1-\theta_0}}, \\ E[N | H_1] = \frac{\beta \ln \frac{\beta}{\alpha} + (1-\beta) \ln \frac{1-\beta}{1-\alpha}}{\theta_1 \ln \frac{\theta_1}{\theta_0} + (1-\theta_1) \ln \frac{1-\theta_1}{1-\theta_0}}. \quad (2)$$

Видно, что число шагов зависит от четырех параметров:  $\alpha$ ,  $\beta$ ,  $\theta_0$ ,  $\theta_1$ . Таким образом, определив эти параметры и проводя пошаговое вычисление отношения правдоподобия для каждого удаленного  $src\_ip$  за конечное число шагов можно принять решение, какая из гипотез справедлива.

В работе приведены результаты исследования, позволяющие рационально задать значения  $\alpha$ ,  $\beta$ ,  $\theta_0$ ,  $\theta_1$ . В рассматриваемом блоке данных предварительно было выделено множество эталонных сканеров, обозначим его  $TRUESCAN$ . Это множество ip-адресов сканеров, которые желательно выявить с помощью предлагаемого алгоритма. Для оценки алгоритма используются приведенные в п. 3.1 характеристики: производительность и эффективность. Сформулируем их в терминах гипотез.

Производительность  $P$  – отношение количества ip-адресов, правильно распознанных как сканеры (принадлежат множеству  $TRUESCAN$ ) к общему количеству ip-адресов, которые алгоритм определил как сканеры:

$$\frac{\text{принята гипотеза } H_1 / \text{верна гипотеза } H_1}{\text{принята гипотеза } H_1}$$

Эффективность  $E$  – отношение количества ip-адресов, правильно распознанных как сканеры (принадлежат множеству  $TRUESCAN$ ) к общему количеству эталонных сканеров:

$$\frac{\text{принята гипотеза } H_1 / \text{верна гипотеза } H_1}{|TRUESCAN|}$$

По этим параметрам алгоритм сравнивается с Bro и Snort. Выясняется, что TRW показывает наивысшую эффективность при производительности выше, чем 0.96.

## 5.2. Модель TRWSYN

Эта модель [12] является модификацией модели TRW и предназначена для обнаружения сканеров в транзитной сети. Для его работы: во-первых, не требуется априорных знаний о реакции хостов на запрос на установление tcp-соединения; во-

вторых, он пригоден для сети с асимметричной маршрутизацией.

В данной модели, также как и в TRW, рассматриваются попытки устройства с адресом  $src\_ip$  установить tcp-соединение с устройством с адресом  $dst\_ip$ . Но статус tcp-соединения определяется не на основе одного пакета с флагом SYN, а на основе последовательности пакетов.

В качестве исходных данных рассматривается последовательность потоков. Сырые данные группируются в потоки (*flows*) на основании следующих правил: пакет принадлежит потоку, если поля его tcp/ip заголовка соответствуют фиксированной пятерке  $\{src\_ip, dst\_ip, src\_port, dst\_port, protocol\}$ ; время, в которое появился первый пакет, соответствующий пятерке, считается временем начала потока; поток считается завершенным, если истек установленный таймер; временем завершения потока является время появления последнего пакета. Заметим, что при проверке работы алгоритма исходные данные, также как и в случае TRW, были собраны предварительно. Это делалось намеренно, для того, чтобы иметь возможность определить множество «реальных» сканеров, с которым можно сравнивать результаты работы алгоритма.

Обозначим через  $flow.src\_ip$ ,  $flow.dst\_ip$ ,  $flow.pkts$  и  $flow.flag$  адреса источника и назначения, количество пакетов в потоке и tcp-флаги соответственно. В модели TRWSYN считается, что произошло событие для потока  $flow.src\_ip$ , если поток закончился. Если  $flow.src\_ip$  содержит один tcp-пакет с флагом SYN, то соединение считается неуспешным ( $Y_i=1$ ). Если  $flow.src\_ip$  содержит несколько пакетов или один пакет без флага SYN, то соединение считается успешным ( $Y_i=0$ ). На самом деле, реальное соединение может быть и неуспешным, но без этой погрешности алгоритм не применим к транзитной сети.

В алгоритме TRWSYN основной цикл для вычисления значения  $\Lambda(Y)$  для каждого  $flow.src\_ip$  выглядит следующим образом:

если

$(flow.pkts > 1)$  или

$(flow.pkts == 1 \text{ и } flow.flag \neq SYN)$ ,

$$\text{то } \Lambda(Y) = \Lambda(Y) * \frac{P[Y_i = 0 | H_1]}{P[Y_i = 0 | H_0]},$$

иначе, если  $(flow.pkts == 1)$  и  $(flow.flag == SYN)$ ,

$$\text{то } \Lambda(Y) = \Lambda(Y) * \frac{P[Y_i = 1 | H_1]}{P[Y_i = 1 | H_0]}.$$

Механизм принятия решения относительно каждого  $scr\_ip$  выглядит так: если  $\Lambda(Y) \geq \eta_1$ , то хост с данным  $scr\_ip$  считается сканером, а если  $\Lambda(Y) \leq \eta_0$ , то хост с данным  $scr\_ip$  не считается сканером.

### 5.3. Модель TAPS

В основе модели TAPS (Time-based Access Pattern Sequential Hypothesis Testing) [12] лежит предположение о том, что сканеры имеют боль-

шую пропорцию соотношения ip-адресов к портам или, наоборот, портов к ip-адресам в сравнении с не сканерами. В качестве механизма порождения событий используются временные интервалы. Мотивация выбора истечения заданного интервала времени как события базируется на наблюдении, что атакующим устройствам необходимо поддерживать определенную скорость сканирования.

Задаются два параметра  $T$  – временной интервал и  $k$  – параметр соотношения. В качестве исходных данных рассматривается последовательность потоков. Каждые  $T$  секунд происходит событие. Рассмотрим некоторый  $src\_ip$  и все *flows*, попавшие в интервал, для которых  $flow.src\_ip = src\_ip$ . Подсчитываем количество различных адресов назначения  $flow.dst\_ip$ , обозначим его  $\#ip$ , и количество различных портов назначения  $flow.dst\_port$ , обозначим его  $\#port$ .

$$\text{Если } \frac{\#ip}{\#port} > k \text{ или } \frac{\#port}{\#ip} > k,$$

то событие успешное ( $Y_i=1$ ); (3)

$$\text{если } \frac{\#ip}{\#port} \leq k \text{ и } \frac{\#port}{\#ip} \leq k,$$

то событие неуспешное ( $Y_i=0$ ). (4)

На основании полученных данных вычисляется отношение правдоподобия (1) и сравнивается с заданными порогами.

Алгоритмы TAPS и Snort зависят от скорости канала передачи данных. Этот фактор учитывается с помощью коэффициентов  $k/T$  ( $\#ip/\#port$  за  $T$  секунд) для TAPS и  $N/T$  ( $\#ip$  за  $T$  секунд) для Snort. Для высокоскоростных каналов фактор выше, чем для низкоскоростных. Для Snort приходится выбирать разные коэффициенты в зависимости от скорости канала, чтобы получить хорошие результаты. В случае TAPS, в основе которого лежит метод последовательного анализа, эта зависимость гораздо ниже. Фактически можно пользоваться одной и той же величиной  $k/T$  для каналов с разными скоростями передачи

Для оценки алгоритма определяются три характеристики:

– коэффициент успеха  $R_s$  – отношение количества обнаруженных реальных сканеров к общему количеству реальных сканеров (то же, что эффективность  $E$ );

– коэффициент ложного обнаружения  $R_f^+$  – отношение количества не сканеров, объявленных как сканеры, к общему количеству реальных сканеров;

– коэффициент потерь  $R_f^-$  – отношение количества пропущенных сканеров к общему количеству реальных сканеров.

По этим коэффициентам сравниваются алгоритмы, реализующие модели Snort, TRWSYN и TAPS. Выясняется, что TAPS имеет наивысший коэффициент  $R_s$  при самых низких значениях

$R_f^+$  и  $R_f^-$ . Далее следует TRWSYN, Snort оказался самым неэффективным. Следует заметить, что рассматривалась ранняя версия Snort, где для выявления сканирования вычислялось только соотношение  $N/T$ .

## 6. Модификация алгоритма TAPS

Для обнаружения сканирующих устройств мы предлагаем модификацию алгоритма, предложенного в модели TAPS [12], который интересен по трем причинам. Во-первых, он не предполагает априорных знаний об исследуемом участке сети. Во-вторых, он расширяет базовый подход, делая выбор метрики  $N/T$  не столь критичным. В-третьих, мы обладаем опытом использования такой метрики для выявления «зомбированных» (инфицированных) локальных хостов [13]. Помимо сканеров, обнаруживаемых алгоритмом TAPS, наш алгоритм выявляет «тонкие» сканеры, те которые интенсивно работают в коротком временном интервале, затем делают значительный перерыв и вновь возобновляют сканирование.

### 6.1 Постановка задачи

Формат исследуемых данных соответствует понятию *flow*, предложенному в Cisco IOS NetFlow [14]. Данные о трафике непрерывно поступают от маршрутизаторов и накапливаются на рабочей станции в виде блоков значительного объема. В роли коллектора потоков выступает рабочая станция, оснащенная пакетом утилит *flow-tools* [15] для работы с потоками формата [14] и дополнительными утилитами. Коллектор настроен на агрегирование информации за фиксированный интервал времени  $M$ . Результатом работы коллектора являются файлы, состоящие из записей, содержащих нужные для работы алгоритма поля.

Анализируя множество таких записей (блок данных), накопленных за время  $M$ , необходимо определить адреса устройств, которые ведут себя «неадекватно». Время для проведения анализа ограничено, не более  $M$ , т.е. до получения следующего блока данных, в противном случае результаты устаревают и теряют значимость. Рассматривается только tcp/udp трафик.

### 6.2 Алгоритм

Наш алгоритм базируется на методе последовательного анализа [11]. Аналогично TAPS событием является истечение временного интервала  $T$ , где  $T < M$ . Для каждого адреса источника, проявившего активность во время текущего интервала, проверяются соотношения (3) и (4) и пересчитывается отношение правдоподобия (1). В пошаговой нотации наш алгоритм выглядит так:

**Входные данные:** Блок данных со статистикой, полученной за интервал  $M$ . При работе алгоритма анализируются следующие поля записей:

$\{t1, protocol, src\_ip, src\_port, dst\_ip, dst\_port\}$ , где  $t1$  – время начала сбора потока.

**Шаг 0.** Упорядочиваем блок данных по полю  $t1$  в порядке не убывания.

**Шаг 1.** Инициализируем  $S$  – множество подозреваемых ip-адресов источников,  $S\_TEMP$  – множество ip-адресов источников, встретившихся за промежуток времени  $T$ , множество сканеров  $SCAN$  (верна гипотеза  $H_1$ ) и множество не сканеров –  $NOTSCAN$  (верна гипотеза  $H_0$ ). Задаем значения параметров алгоритма  $T, k, \eta_0, \eta_1, \theta_0$  и  $\theta_1$ .

**Шаг 2.** Рассматриваем первую строку блока данных. Она содержит следующие поля:  $t1\_1, src\_ip\_1, dst\_ip\_1, dst\_port\_1$ . Задаем значение начальной временной отметки  $T_0 := t1\_1$ . Зафиксируем  $src\_ip\_1$ . Помещаем  $src\_ip\_1$  в множество  $S$  и в множество  $S\_TEMP$ .

Инициализируем  $Da_i$  – множество различных значений поля  $dst\_ip$ , встречающихся в записях, где адресом источника выступает  $src\_ip\_1$  и  $Dp_i$  – множество различных значений поля  $dst\_port$ , встречающихся в записях, где адресом источника выступает  $src\_ip\_1$ . Каждому адресу источника  $src\_ip\_j$  будем ставить в соответствие множества  $Da_j$  и  $Dp_j$ , при этом  $\forall i, j (src\_ip\_j = src\_ip\_i \Leftrightarrow i = j)$ .

Помещаем  $dst\_ip\_1$  в множество  $Da_i$ , а  $dst\_port\_1$  в множество  $Dp_i$ .

**Шаг 3.** Если файл закончился, остановка, иначе рассматриваем очередную строку  $j$ .

Если  $src\_ip\_j \in SCAN$  или  $src\_ip\_j \in NOTSCAN$ , то перейти на Шаг 8 – пропускаем строку, иначе: Если  $src\_ip\_j \notin S$ , то помещаем  $src\_ip\_j$  в множество  $S$  и в множество  $S\_TEMP$  и инициализируем соответствующее отношение правдоподобия  $\Lambda_j := 1$ .

Если  $dst\_ip\_j \notin Da_j$ , то помещаем  $dst\_ip\_j$  в множество  $Da_j$ .

Если  $dst\_port\_j \notin Dp_j$ , то помещаем  $dst\_port\_j$  в множество  $Dp_j$ .

**Шаг 4.** Проверяем, истек ли таймер  $T$ . Таймер считается истекшим, если выполняется неравенство:  $(t1\_j - T_0) > T$ . Если таймер истек, то обновляем значение  $T_0, T_0 := t1\_j$ , и переходим на Шаг 5, иначе переходим на Шаг 3.

**Шаг 5.**  $\forall src\_ip\_i \in S\_TEMP (i=1..|S\_TEMP|)$  производим следующие действия.

(а) Вычисляем мощность соответствующих множеств  $Da_i$  и  $Dp_i$ . Обозначим  $|Da_i| \equiv \#ip, |Dp_i| \equiv \#port$ .

(б) Обновляем отношение правдоподобия  $\Lambda_i$  (1), соответствующее адресу  $src\_ip\_i$ , следующим образом:

$$\text{Если } \frac{\#ip}{\#port} > k \text{ или } \frac{\#port}{\#ip} > k, \text{ то } \Lambda_i := \Lambda_i \cdot \frac{1-\theta_1}{1-\theta_0}$$

(событие успешно)

иначе,

если  $\frac{\#ip}{\#port} \leq k$  и  $\frac{\#port}{\#ip} \leq k$ , то  $\Lambda_i := \Lambda_i \cdot \frac{\theta_1}{\theta_0}$

(событие неуспешно)

(в) Сохраняем текущее значение  $\Lambda_i$ .

**Шаг 6.**  $\forall src\_ip\_i \in S\_TEMP$  сверяем соответствующее значение  $\Lambda_i$  с установленными порогами. Если  $\Lambda_i \geq \eta_1$ , то добавляем  $src\_ip\_i$  в множество сканеров  $SCAN$ , исключаем его из множества  $S$  и переходим на Шаг 7. Если  $\Lambda_i \leq \eta_0$ , то добавляем  $src\_ip\_i$  в множество  $NOTSCAN$ , исключаем его из множества  $S$  и переходим на Шаг 7. Иначе переходим на Шаг 7, не производя никаких действий.

**Шаг 7.**  $\forall i (src\_ip\_i \in S\_TEMP)$  инициализируем множества  $Dp_i := \emptyset$ ,  $Da_i := \emptyset$ . Инициализируем множество адресов источников, исследованных за текущий период,  $S\_TEMP := \emptyset$ .

**Шаг 8.** Переходим на Шаг 3.

После завершения чтения всех записей блока данных, накопленных за интервал времени  $M_i$ , программа, реализующая алгоритм, приостанавливает работу до получения следующего блока  $M_{i+1}$ . В результате работы программы над блоком  $M_i$ , из множества всех адресов источников выделяются два основных множества  $SCAN$  и  $NOTSCAN$ . Множество  $SCAN$ , представляющее основной интерес, содержит ip-адреса сканеров. Для каждого такого ip-адреса доступна дополнительная информация, например, число шагов, затраченных до принятия решения, и тип сканирования (порты/адреса).

### 6.3 Особенности

В отличие от алгоритма TAPS, анализирующего потоки, построенные в ходе работы, в нашем случае формат исследуемых данных заранее фиксирован и определяется идеологией, предложенной в Cisco IOS NetFlow. Так же как и TAPS, разработанный алгоритм использует механизм временных интервалов, т.е., производит обновление отношения правдоподобия каждые  $T$  секунд. При этом в алгоритме TAPS прекращение сетевой активности наблюдаемого устройства с адресом  $src\_ip$  в рассматриваемый период является показателем того, что устройство ведет себя как безопасный хост, т.е. отношение правдоподобия пересчитывается с  $Y_i=0$ . В предлагаемом алгоритме отношение правдоподобия для ip-адресов, не проявляющих активность за данный промежуток времени, не обновляется. Это позволяет отследить сканеры, которые действуют интенсивно в течение короткого промежутка времени и делают большие перерывы между попытками сканирования. Однако, в свою очередь, это может увеличить количество адресов, для которых вообще не принято решение.

Заметим, что ip-адреса, попавшие в множество не сканеров ( $NOTSCAN$ ) исключаются из рассмотрения до конца рассматриваемого блока данных. Это создает опасность того, что могут быть пропущены устройства, которые сначала не проявляют аномальной активности, а затем начинают интенсивно сканировать. Поначалу кажется целе-

сообразным не формировать множество  $NOTSCAN$ , т.е. фактически перейти к рассмотрению алгоритма с одной гипотезой  $H_1$  – «сканер». Однако, как показала практика, применение классического варианта предпочтительнее. Рассмотрение лишь одной гипотезы увеличивает количество шагов и количество ложных сканеров, но практически не расширяет множество обнаруживаемых сканеров. Кроме того, для анализа очередного блока статистики программа перезапускается с пустым множеством  $NOTSCAN$ , т.е. ip-адреса, попавшие в это множество, исключаются из рассмотрения на небольшой промежуток времени, поэтому риск пропустить активный сканер очень мал.

## 7. Эксперимент и оценки

### 7.1 Формат данных

Рассмотрим формат анализируемых данных. Поток (network flow) – это однонаправленная последовательность пакетов между определенным источником и получателем. Подробнее, каждый пакет, проходящий через сетевое устройство, исследуется на наличие ip-атрибутов. Традиционно Cisco IOS NetFlow исследует следующие атрибуты: ip-адрес источника, ip-адрес назначения, порт источника, порт назначения, протокол следующего за ip протоколом уровня, тип обслуживания (TOS), входной интерфейс устройства, с которого поступают пакеты потока и др.

Правила сбора и отсылки потоков для используемой версии 5 Cisco IOS NetFlow таковы. Потоки собираются устройством и считаются завершенными, когда произойдет одно из следующих событий: транспортный протокол указывает на завершение взаимодействия (флаг TCP FIN); трафик не активен в течение 15 с; для потоков, остающихся постоянно активными, поток принудительно завершается через 30 мин. Потоки, генерируемые сетевыми устройствами, инкапсулируются в UDP-дейтаграммы. Пакеты посылаются на рабочую станцию, задающуюся в конфигурации экспортера, либо когда количество потоков превзойдет определенный максимум, либо каждую секунду (выбирается то событие, которое произошло первым).

В нашем случае коллектор настроен на агрегирование информации в течение 30 минут. Результатом работы коллектора являются файлы, состоящие из записей следующего формата:

```
|t1|t2|src_ip|dst_ip|src_org|dst_org|src_int|dst_int|src_port|dst_port|protocol|packets|octets|TCP-flags|,
```

где  $t1$  – время начала потока (время появления первого пакета/мс);

$t2$  – время окончания потока (время появления последнего пакета/мс);

$src\_ip$  – ip-адрес источника;

$dst\_ip$  – ip-адрес назначения;

*src\_org* – индекс организации из некоторого фиксированного множества организаций, к которой принадлежит хост с *ip*-адресом источника (0 – организация неизвестна);

*dst\_org* – индекс организации – получателя (0 – организация неизвестна);

*src\_int* – SNMP-индекс интерфейса сетевого устройства, по которому поступил поток данных от источника;

*dst\_int* – SNMP-индекс интерфейса сетевого устройства, на который отправляется поток данных;

*src\_port* – tcp/udp порт источника;

*dst\_port* – tcp/udp порт назначения;

*protocol* – протокол (поле *protocol* ip-пакета).

*packets* – количество пакетов в потоке;

*octets* – количество байтов в потоке;

*flags* – аккумулярованные флаги (FIN, SYN и RST) tcp-пакетов потока. Если установленных флагов нет или протокол отличен от tcp, в поле *flags* стоит 0.

## 7.2 Эксперимент и оценка эффективности работы алгоритма

Для того чтобы оценить работу алгоритма, была создана программа на языке Perl, реализующая этот алгоритм. Анализировался «эталонный» блок данных, собранных за 30 мин наблюдения и содержащий  $4,6 \cdot 10^6$  записей. Этому блоку соответствует 29650 Мбайт реального трафика сети Интернет СО РАН [16].

В отсутствие оракула, способного определить правильность принятия той или иной гипотезы, записи были предварительно исследованы, в результате чего было создано эталонное множество сканеров *TRUESCAN*. В него мы поместили те и только те *ip*-адреса устройств, которые проявили себя как сканеры. Множество *TRUESCAN* строилось в два этапа. Исследовались *ip*-адреса источников. Пусть *#port* – количество портов назначения, *#ip* – количество *ip*-адресов назначения, зафиксированное в блоке для некоторого *src\_ip*. Если выполняется

одно из неравенств  $\frac{\#ip}{\#port} \geq k_1$  или  $\frac{\#port}{\#ip} \geq k_1$  (при этом  $k_1$  выбирается достаточно большим, например,

$k_1 = 100$ ), то помещаем этот *ip*-адрес в *TRUESCAN*. Затем, после дополнительного анализа оставляем только те *ip*-адреса, которые мы признали сканерами. На втором этапе анализируется список *ip*-адресов сканеров, который получен в результате работы программы, реализующей алгоритм. Те *ip*-адреса, которые не вошли в эталонное множество на первом этапе, но были признаны сканерами после анализа, добавляются к *TRUESCAN*. В исследуемом блоке содержалось 267044 различных адресов источников, из них 79 принадлежало множеству *TRUESCAN*. Следует заметить, что процесс построения эталонного множества требует внимания и значительных временных затрат.

Поскольку не существует общего подхода к определению характеристик алгоритмов распознавания сканирования, были рассмотрены две меры, оценивающие работу алгоритма: эффективность *E* и производительность *P*, предложенные в [5].

При параметрах  $k = 10$ ,  $T = 10$  с,  $\alpha = 0,01$ ,  $\beta = 0,01$ ,  $\theta_0 = 0,9$  и  $\theta_1 = 0,1$  за время работы программы, равное ~4,5 мин, был получен следующий результат. Выявлено 40 сканеров ( $|SCAN|$ ), из них 38 входят в *TRUESCAN*. Устройства, входящие в *SCAN*, создали 43 Мбайт трафика, что составляет 0,15% от общего трафика блока. Сканеры не генерируют большие объемы трафика, так как в их задачу входит разведка, а не обмен данными. Безопасными были признаны ( $|NOTSCAN|$ ) 72303 адреса. Для остальных адресов не было принято решение, так как соответствующие устройства не проявили достаточную активность. Количество пересчетов отношения правдоподобия для *src\_ip* до принятия одной из гипотез в среднем выполнялось за три итерации основного цикла алгоритма.

Полученные характеристики алгоритма:  $P=0,95$ ,  $E=0,48$  свидетельствуют о том, что производительность алгоритма (отношение числа выявленных сканеров к  $|SCAN|$ ) достаточно высока, т.е. количество ложных тревог невелико. Невысокая эффективность (отношение числа выявленных сканеров к  $|TRUESCAN|$ ) вызвана тем, что среди не выявленных сканеров основной процент составляют сканеры, которые за короткий промежуток времени сканируют большое количество адресов, а затем не проявляют активность в течение всего рассматриваемого периода. В [5] было замечено, что между производительностью и эффективностью существует обратная связь. Если рассматривать задачу выявления локальных инфицированных хостов, то предпочтение следует отдать высокой производительности. Небольшое время работы алгоритма на блоке данных, собранных за 30 мин, позволяет встроить его в систему анализа «сетевой погоды» [17], функционирующую в режиме реального времени.

## 7.3 Графическое представление результатов

Здесь представлена графическая интерпретация результатов работы программы, реализующей наш алгоритм, при различных значениях параметров.

Рис. 1 отображает зависимость коэффициента ложного обнаружения  $R_f^+$  (см. п. 5.3) от выбора параметра  $k$  при различных значениях параметров  $\theta_0$  и  $\theta_1$ . Таймер во всех случаях выбирался равным 10 с ( $T=10$ ). Приводятся два графика: для  $\theta_0 = 0,9$ ;  $\theta_1 = 0,1$  и для  $\theta_0 = 0,8$ ;  $\theta_1 = 0,2$ . По оси абсцисс откладывается значение  $k$ , по оси ординат откладывается  $R_f^+$ .

Из рис. 1 видно, что  $R_f^+$ , при значениях  $\theta_0 = 0,9$  и  $\theta_1 = 0,1$  ниже, чем при значениях  $\theta_0 = 0,8$  и  $\theta_1 = 0,2$ . Для обоих случаев оптимальное значение параметра

тра  $k \approx 10$ . Рис. 2 отражает зависимость эффективности работы алгоритма  $E$  от выбора параметра  $k$  при различных значениях  $\theta_0$  и  $\theta_1$ . Приводятся два графика: для  $\theta_0 = 0,9, \theta_1 = 0,1$  и  $\theta_0 = 0,8, \theta_1 = 0,2$ . По оси абсцисс откладывается значение  $k$ , по оси ординат откладывается  $E$ . Видно, что при значениях  $\theta_0 = 0,9$  и  $\theta_1 = 0,1$  эффективность выше, чем при значениях  $\theta_0 = 0,8$  и  $\theta_1 = 0,2$ .

На Рис. 3 отражена зависимость количества распознанных эталонных сканеров от выбора параметра  $k$  при различных  $\theta_0$  и  $\theta_1$ . Приводятся два графика: для значений  $\theta_0 = 0,9, \theta_1 = 0,1$  и  $\theta_0 = 0,8, \theta_1 = 0,2$ . По оси абсцисс откладывается значение  $k$ , по оси ординат – количество распознанных эталонных сканеров,  $S_c$ .

Из Рис. 3 видно, что количество распознанных сканеров при  $\theta_0 = 0,9$  и  $\theta_1 = 0,1$  выше, чем при  $\theta_0 = 0,8$  и  $\theta_1 = 0,2$ . Таким образом, можно сделать вывод, что значения параметров  $\theta_0 = 0,9$  и  $\theta_1 = 0,1$  предпочтительнее, чем  $\theta_0 = 0,8$  и  $\theta_1 = 0,2$ . Это можно интерпретировать так: чем больше различают-

ся между собой гипотезы, тем лучше результаты работы алгоритма.

Рассмотрим некоторый ip-адрес, принадлежащий множеству *TRUESCAN*, для которого проводится процедура принятия решения при различных значениях параметров  $\theta_0, \theta_1, \alpha$  и  $\beta$ . На рис. 4 приведено графическое представление процедуры принятия решения при  $\theta_0 = 0,9, \theta_1 = 0,1$ , ошибка первого рода  $\alpha = 0,01$ , ошибка второго рода  $\beta = 0,01$ . По оси абсцисс откладывается количество шагов алгоритма  $n$ , по оси ординат –  $Q = \sum_{i=1}^n Y_i$ .

Увеличим значения ошибок первого и второго рода и посмотрим, как изменится процедура принятия решения. На рис. 5 приведено графическое представление процедуры принятия решения при  $\theta_0 = 0,9, \theta_1 = 0,1$ , ошибка первого рода  $\alpha = 0,05$ , ошибка второго рода  $\beta = 0,05$ . По оси абсцисс откладывается количество шагов алгоритма –  $n$ , по оси ординат –  $Q$ .

Сравнив рис. 4 и рис. 5, можно сделать вывод, что при увеличении значений ошибок первого и второго рода расстояние между прямым  $H_0$  и  $H_1$

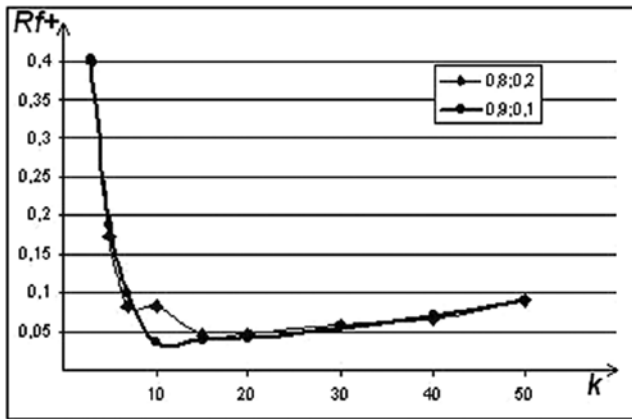


Рис.1. Зависимость  $R_f$  от выбора параметра  $k$

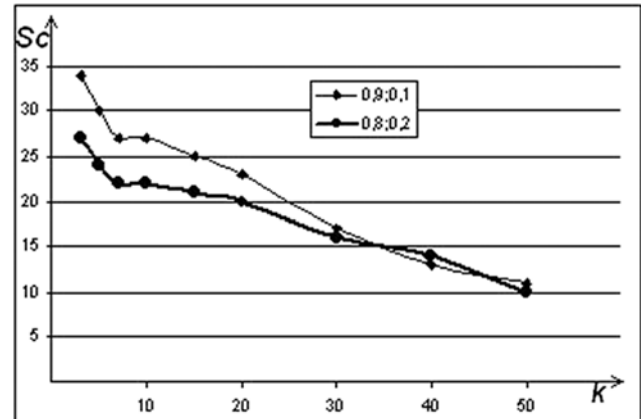


Рис.3. Зависимость количества распознанных сканеров  $S_c$  от выбора параметра  $k$

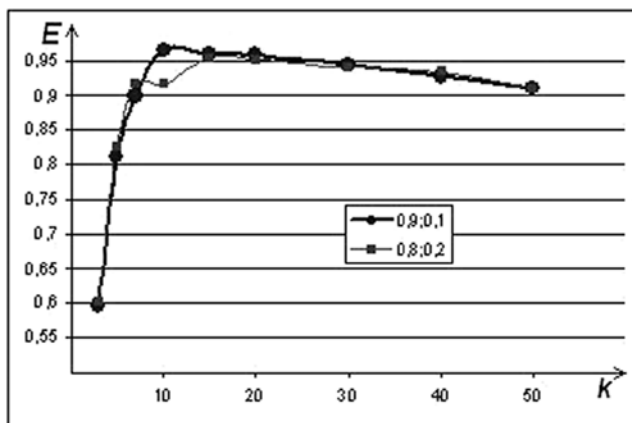


Рис.2. Зависимость эффективности алгоритма  $E$  от выбора параметра  $k$

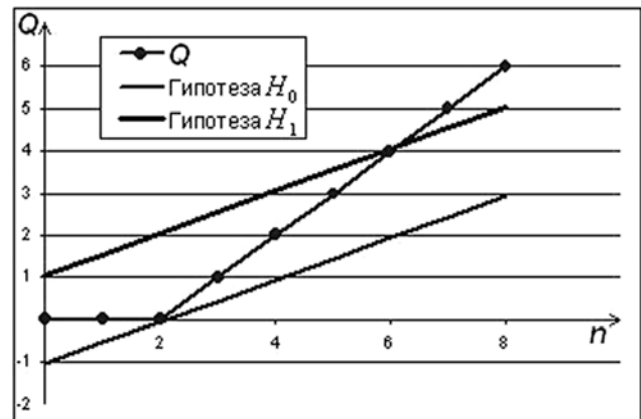


Рис.4. Процедура принятия решения для данного ip-адреса при ошибках первого и второго рода, равных 0,01

уменьшается, что логично, так как уменьшается точность алгоритма, что может привести к принятию неверной гипотезы, как видно из рис. 5.

Теперь посмотрим, как будет выглядеть диаграмма принятия решения при других значениях  $\theta_0$  и  $\theta_1$ . Пусть  $\theta_0 = 0,8$ ,  $\theta_1 = 0,2$ ,  $\alpha = 0,01$ ,  $\beta = 0,01$ .

Сравнение рис. 4 и рис. 6 иллюстрирует тот факт, что для принятия решения при значениях  $\theta_0 = 0,8$  и  $\theta_1 = 0,2$  требуется большее число шагов, чем при значениях  $\theta_0 = 0,9$  и  $\theta_1 = 0,1$ . То есть уменьшение  $\theta_0$  и увеличение  $\theta_1$  ведет к росту числа шагов. Аналогичные выводы можно сделать, проанализировав формулы (2), оценивающие среднее число шагов.

### Заключение

Безусловно, метод определения сканеров, предложенный в модели TRW [8], является перспективным, поскольку он базируется на математической теории, позволяющей определить ожидаемую эффективность соответствующего алгоритма, и не требует длительного первоначального изучения трафика. Подтверждением этого является появление моделей TRWSYN [12], TAPS [12] и модифицированной модели TAPS, в которые заложены эвристические предположения, отличные от [8].

Существуют пути улучшения характеристик модели TRW. Можно расширить эвристические предположения о поведении сканеров. Во-первых, определять параметры более консервативно для некоторых сервисов, например, http, так как http сканер трудно отличить от http проху. Во-вторых, делать различие между попытками соединения, на которые получен отказ, и попытками, оставшимися без ответа, так как последние больше похожи на «выстрел в темноте». В третьих, внести компоненты корреляции: считать две последовательных неудачных попытки соединения более подозрительным поведением, чем перемежающиеся попытки. Однако это приведет к усложнению ал-

горитма и может поставить под сомнение его эффективность.

Главным ограничением, заложенным в рассматриваемую методику, является предположение о независимости наблюдений. Например, количество устройств, ложно принятых за сканеры, было бы меньше, если учитывать факт, что сканеры чаще всего последовательно перебирают адреса подсетей, в отличие от активных хостов. В случае распределенного сканирования также нужно учитывать совместное распределение, что также значительно усложняет алгоритм.

В ряде работ предлагается решать задачу выявления сканеров с применением технологии Data Mining. Например, в [18] утверждается, что известные алгоритмы, в том числе и TRW, который имеет статус «state-of-the-art», страдают от большого числа ложных тревог, так как не отличают от сканирования p2p трафик, трафик веб-поисковиков, а также «backscatter traffic», когда, например, атакуемый сервер отвечает на DoS атаки по вымышленным адресам. Предлагается использовать классификатор для разделения потоков данных на четыре категории: *SCAN*, *P2P*, *NORMAL*, *NOISE*. Разделение делается на основании информации, полученной во время достаточно длительного наблюдения за трафиком, и ряда экспертных правил. Достоинства этой методики оцениваются с помощью метрик, эквивалентных производительности и эффективности. На экспериментальной основе в [18] утверждается, что характеристики алгоритма на базе Data Mining выше, чем у TRW.

Существуют подходы к созданию систем IDS, основанные на моделировании сетевого поведения, например, с помощью нейронных сетей. Примером может служить работа [19], где предлагаются методы обучения сети для выявления атак типа зондирование, DoS, u2r (user to root), r2l (remote to local). Применяются также и генетические алгоритмы. В работе [20] рассматривается применение такого алгоритма для дифференциации нормального и аномального сетевого соединения. Этот ал-

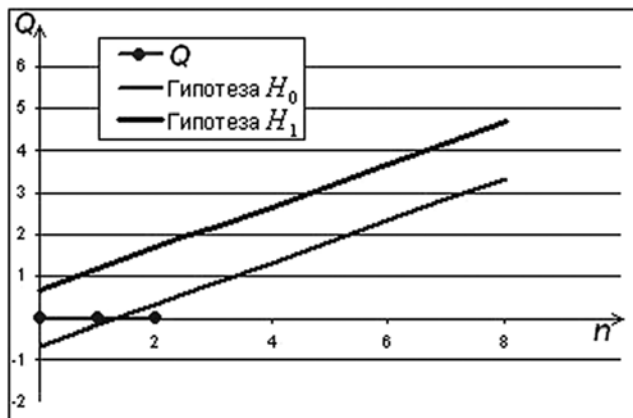


Рис. 5. Процедура принятия решения для данного ip-адреса при ошибках первого и второго рода, равных 0,05

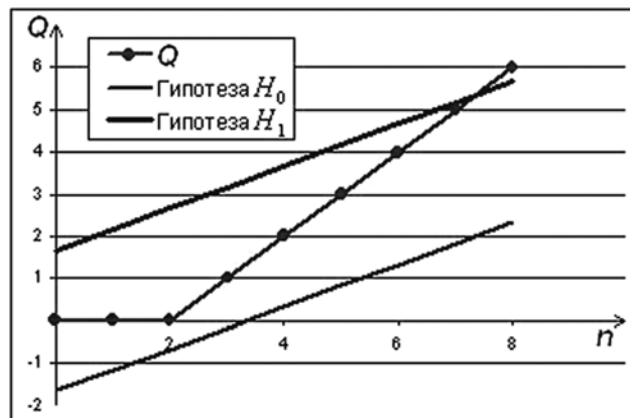


Рис. 6. Процедура принятия решения для данного ip-адреса при ошибках первого и второго рода, равных 0,01;  $\theta_0 = 0,8$ ,  $\theta_1 = 0,2$

горитм использует предварительную информацию для построения множества правил, учитывающих различные сетевые параметры и применяемых системой обнаружения вторжения. Для защиты компьютерной системы используются также средства, имитирующие поведение биологической иммунной системы, так называемые иммунные алгоритмы, например [21]. Иммунные процессы, способные к самоорганизации, изучают, распознают, пропускают или отвергают запросы к компьютерной системе.

Следует заметить, проблема интерпретации полученного ответа (адресов устройств из множества TRUESCAN) остается открытой. Можно блокировать эти адреса, например, как в системе Bro [4]. Это применимо для небольших сетей, и в Bro имеется возможность указать, какие адреса никогда не следует блокировать. Отметим, что значительная часть этих адресов может оказаться фальшивой. Существует методика определения фальшивых адресов с помощью «ловушки» Honeypot [22], которая в ответ на попытку соединения предлагает источнику осуществить полную процедуру handshake, которая обычно не заканчивается, если адреса фальшивые. С помощью этой методики можно выявить неиспользуемые ip-адреса, однако это приводит к дополнительной нагрузке и, соответственно, к значительному увеличению времени работы алгоритма, что ставит под сомнение возможность его работы в режиме on-line. Видимо, следует классифицировать сканеры по степени их опасности и определить множество соответствующих порогов, о преодолении которых сообщать системному администратору сети, от которого зависят дальнейшие действия в рамках принятой политики сетевой безопасности.

### Литература

1. <http://www.iana.org/assignments/port-numbers/>
2. <http://www.snort.org/>
3. <http://nmap.org/>
4. <http://www.bro-ids.org/>
5. Staniford S., Hoagland J.A., McAlerney J.M. Practical Automated Detection of Stealthy Portscans. // Journal of Computer Security, Volume 10, Issue 1-2 (2002). P. 105-136.
6. Niedermayer D. An Introduction to Bayesian Networks and their Contemporary Application. <http://www.niedermayer.ca/papers/bayesian/bayes.html/>
7. Kirkpatrick S., Gelatt C. D., Vecchi M. P. Optimization by Simulated Annealing // Science, Vol. 220, No. 4598. (May 13, 1983), P. 671-680.
8. Jung J., Paxton V., Berger A.W., Balakrishnan H. Fast Portscan Detection Using Sequential Hypothesis Testing. // Proceedings IEEE Symposium on Security and Privacy, 2004. P. 211-225
9. Ertöz L., Eilertson E., Dokas P., Kumar V., Long K. Scan Detection - Revisited // Technical report AHPARC 127, University of Minnesota - Twin Cities, 2004. P. 127.

10. Lekie C., Kotagiri R. A Probabilistic Approach to Detecting Network Scans // IEEE/IFIP Network Operations and Management Symposium (NOMS) 2002. P. 359-372.
11. Вальд А. Последовательный анализ. Москва: Физмат, 1960. 328 с.
12. Sridharan A., Ye T., Bhattacharyya S. Connectionless Port Scan Detection on the Backbone // IEEE International Performance, Computing and Communications Conference (IPCCC), 2006. P. 10-20.
13. Бредихин С.В., Щербакова Н.Г. Две компоненты анализа сетевого трафика // Вестник Новосибирского государственного университета. Серия: информационные технологии. Новосибирск, 2008. Т. 6, вып. 1. С. 10-14.
14. <http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.pdf>
15. <http://www.splintered.net/sw/flow-tools/>
16. СПД СО РАН. Сеть передачи данных Сибирского отделения РАН. Информационные материалы научно-координационного совета целевой программы «Информационно-телекоммуникационные ресурсы СО РАН». Институт вычислительных технологий СО РАН. Новосибирск, 2005. 79 с.
17. Бредихин С.В., Ляпунов В.М., Щербакова Н.Г. Анализатор «сетевой погоды» // Вестник Новосибирского государственного университета. Серия: информационные технологии. Новосибирск, 2005. Т. 2, вып. 1. С. 62-67.
18. Simon G.J., Hui Xiong, Eilerton E., Kumar V. Scan Detection: A Data Mining Approach // Sixth SIAM International Conference on Data Mining, 2006. P. 118-129.
19. Ray-I Chang, Liang-Bin Lai, Wen-De Su, Jen-Chieh Wang, Jen-Shiang Kouh. Intrusion Detection by Backpropagation Neural Networks with Sample-Query and Attribute-Query // International Journal of Computational Intelligence Research. Vol.3, No. 1 (2007). P. 6-10.
20. Wei Li. Using Genetic Algorithm for Network Intrusion Detection // Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference, Kansas City, Kansas, May 24-27, 2004.
21. Visconti A., Fusi N, Tahayori H. Intrusion Detection via Artificial Immune System: a Performance-based Approach, 2008 // IFIP International Federation for Information Processing, Volume 268; Biologically-Inspired Collaborative Computing; Mike Hinchey, Anastasia Pagnoni, Franz J. Rammig, Hartmut Schmeck; (Boston: Springer). P. 125-135.
22. [http://www.dmoz.org/Computers/Security/Honeypots\\_and\\_Honeynets/](http://www.dmoz.org/Computers/Security/Honeypots_and_Honeynets/)