А. Ю. Михайлов

Средства эффективного учета и анализа сетевого трафика

Научный руководитель: к.т.н Ю. В. Шевчук

Аннотация. В данной работе предложена архитектура системы для анализа сетевого трафика, основанная на использовании IPFIX. Рассмотрено приложение данной системы к задаче автоматического определения сетевых аномалий.

1. Введение

Необходимым инструментом эксплуатации современной компьютерной сети являются средства учета и анализа сетевого трафика. С их помощью решаются следующие задачи:

- расчет стоимости услуг сети;
- анализ «узких мест» и планирование расширения сети;
- обнаружение сетевых атак и аномалий.

В условиях быстрого роста пропускной способности сетевых каналов и объемов трафика к средствам учета и анализа трафика должно предъявляться требование эффективности. Система учета должна обрабатывать трафик в реальном времени, не предъявляя высоких требований к аппаратной платформе.

Данная работа описывает систему учета и анализа трафика, разработанную автором для системы телекоммуникаций «Ботик». В системе телекоммуникаций «Ботик» используются маршрутизаторы на базе операционной системы Linux, не имеющей штатных средств экспорта информации о трафике. В задачу автора входила разработка эффективных средств экспорта информации о трафике из ядра ОС Linux, а также использование этой информации для обнаружения сетевых аномалий.

2. Выбор способа представления информации о трафике

В работе автора [1] были проанализированы существующие методы учета сетевого трафика. В том числе был сделан вывод, что сетевые потоки предоставляют собой очень удобный уровень абстракции для анализа сетевого трафика.

Напомню, что сетевой поток определяется как однонаправленная последовательность сетевых пакетов, проходящих через сетевое устройство в определенный промежуток времени, а также имеющих набор совпадающих атрибутов. Например, компания Cisco определяет этот набор атрибутов семеркой: IP–адрес источника, IP–адрес получателя, порт источника, порт получателя, номер протокола, бит ToS, индекс входящего интерфейса.

Существенный момент данного определения заключается в том, что поток определяется как однонаправленная последовательность. Естественным образом можно расширить понятие потока, понимая под ним набор пакетов, идущих в обоих направлениях соединения. Будем далее называть данное расширение определения двунаправленным потоком.

Наиболее распространенным протоколом для экспорта информации о потоках является Netflow. Одной записи Netflow соответствует набор полей с информацией об однонаправленном потоке. Существуют две широко используемые версии протокола: 5 и 9. В версии 5 используется фиксированный формат пакета с заранее определенным набором полей: IP–адреса, порты, счетчик байтов и пакетов и т.д. Основным изменение в 9-ой [2] версии протокола стало то, что пользователь получил возможность определять формат записи, используя механизм шаблонов. Шаблон состоит из пар $(type, length)$, где $type$ — это тип поля (определяет его семантику), а $length$ — это длина. Каждому экспортируемому потоку соответствует шаблон, указывающий каким образом будут интерпретироваться сообщение. Так как определения шаблонов должны посылаться очень редко в сравнении с весомой информацией (записи потоков), этот механизм гораздо более эффективен, чем любой другой, добавляющий описательную информацию к каждой записи (например, XML).

Инициатива IETF привела к объединению и унификации протоколов и приложений, использующих сетевые потоки (в свою очередь, протокол Cisco Netflow является проприетарным). Стандарт

RFC 3917 [3] определяет требования для экспорта информации о потоках для последующей обработки на удаленном устройстве. За основу протокола был выбран Netflow V9. Стандарт RFC 5101 [4] был выпущен в январе 2008 года и определил протокол IP Flow Information Export (IPFIX), который служит для передачи информации о потоках в сетевом окружении. Последовавший стандарт RFC 5102 [5] определил информационную модель для протокола IPFIX, которая используется для кодирования информации как о трафике, так и обо всем процессе.

Благодаря гибкости протокола IPFIX, стало возможным появление стандарта RFC 5103 [6], в котором вводится понятие двунаправленного потока, а также описывается эффективный метод экспорта подобных потоков. Суть идеи заключается во введении понятия обратного поля — поля, характеризующего величину, относящуюся к однонаправленному потоку, текущему в обратном направлении (под обратным направлением понимается движение к инициатору соединения).

В работе автора [1] был предложен эффективный метод получения информации о потоках в формате Netflow V5. Одной из целей данной работы является его расширение для поддержки протокола экспорта IPFIX, что дает следующие преимущества:

- гибкость экспорта и глубокая информационная модель;

  Мы можем формировать шаблон, используя глубокую информационную модель, что может предоставить разнородный материал для анализа.
- возможность развития;

  К сожалению, формат Cisco Netflow v5 не дает возможности для развития. Например, фиксированный формат записи потока делает невозможным экспорт информации о IPv6–потоках. В то время как протокол IPFIX находится в стадии динамичного развития, впереди — множество нововведений, которые можно будет пытаться использовать на практике. Также протокол IPFIX является открытым, что ведет к совместимости всех средств ПО, которые его реализуют.
- двунаправленные потоки.

  Использование двунаправленных потоков позволяет не только уменьшить размер передаваемой информации, но и дает много дополнительной информации, которую можно

использовать для анализа и дальнейшей обработки. Небольшие примеры:

(1) разделение трафика на «отвеченный» и нет;
(2) возможность определить направление трафика, а также инициирующую сторону;
(3) возможность полной реконструкции TCP–сессии;
(4) получение значения RTT для любого потока тривиально (разница между временем начала оригинального потока и обратного ему).

3. Архитектура системы анализа трафика

3.1. Постановка задачи

Архитектура системы, основанной на IPFIX, обычно строится следующим образом: сенсор (Metering Proccess) собирает пакеты с данными в точке наблюдения (Observation Point), возможно фильтруя и агрегируя информацию о них. Далее экспортер (Exporter) передает данные коллектору (Collector) согласно протоколу IPFIX. Обычно коллектор помещает данные на диск для последующей обработки. Расширенную модель, обоснование которой будет дано позже, можно увидеть на рис. 1. Проанализируем составные части данной системы для того, чтобы сформировать требований к будущей системе.

В первую очередь, необходимо расширить ранее разработанный сенсор для поддержки информационной модели IPFIX, а также продумать механизм передачи данных экспортеру. Специфика задачи состоит в том, что сенсор — это программный код, выполняющийся в пространстве ядра Linux. В том числе, это означает, что реализация протокола IPFIX должна удовлетворять всем критериям ядерного кода: небольшой фиксированный размер стека, отсутствие плавающей арифметики, отсутствие зависимости от других библиотек (в том числе и от стандартной библиотеки C), высокое быстродействие. Также необходимо получить эффективное решение проблемы производителя/потребителя в контексте ядра и пользовательского процесса. В задачи процесса–экспортера входит не только получение информации о потоках от сенсора, но также их доставка на удаленное устройство. Конечно, должны быть выполнены требования безопасности (невозможность фальсификации или перехвата) и надежности (гарантированная доставка). Стандартные методы доставки IPFIX, основанные

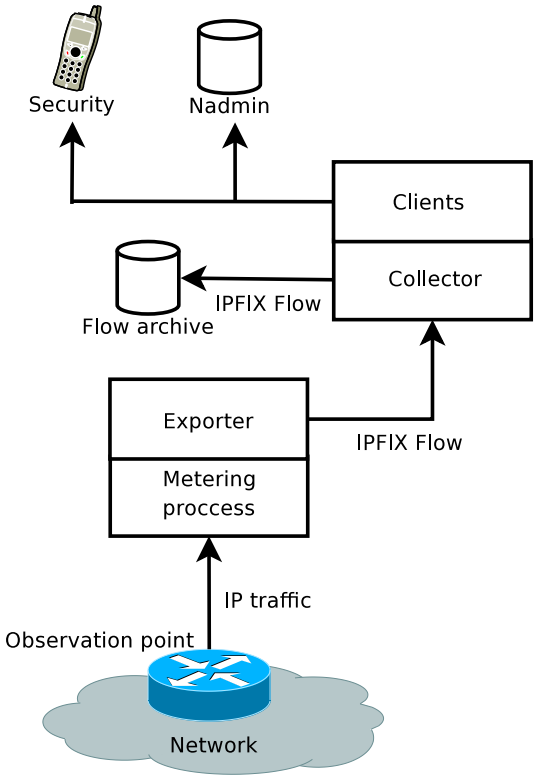

Рис. 1. Архитектура системы

на использовании протоколов UDP/TCP/SCTP без шифрования и аутентификации, данным требованиям не удовлетворяют.

Далее, в очевидные задачи коллектора могут входить как получение IPFIX сообщений из сетевого окружения, так и архивация для долговременного хранения и ретроспективного анализа. С другой стороны, некоторые процессы требуют немедленной обработки получаемых потоков (например, биллинг или модуль определения сетевых атак), поэтому хотелось бы иметь возможность эффективным

образом передавать получаемые потоки другим процессам для обработки в реальном времени. Более того, нужно учитывать что данные процессы могут быть достаточно разнородными.

В силу того, что IPFIX является еще очень молодым протоколом, существует достаточно ограниченное количество его реализаций. Были исследованы готовые решения на предмет возможности их использования в данной работе.

В первую очередь, было исследовано ПО, разработанное в CERT. Эта реализация известна как наиболее полная и соответствующая всем изданным стандартам. В данный набор ПО входит: libfixbuf — библиотека для работы с сообщениями IPFIX, YAF — IPFIX–сенсор, и SiLK — система для сбора (коллектор) и анализа потоков. К сожалению, libfixbuf является достаточно громоздкой и сложной библиотекой, поэтому использовать ее как эталонную реализацию для сенсора достаточно сложно. В свою очередь, система SiLK оказалась достаточно мощной и интересной. Ее возможности для анализа потоков и практические применения обсуждались в [7]. Однако, SiLK не поддерживает в полной мере двунаправленных потоков, а при получении раскладывает их на два однонаправленных, что делает невозможным архивацию потоков в данном формате. Тем не менее, возможность использования SiLK в системе есть: достаточно лишь при необходимости конвертировать информацию в подходящее представление.

Далее, была рассмотрена библиотека libipfix, в задачи которой входит поддержка формата сообщений IPFIX. К сожалению, данная библиотека не поддерживает использование двунаправленных потоков. Однако, реализация библиотеки достаточно компактна и эффективна, поэтому она была выбрана как базис для реализации поддержки IPFIX в ядре.

Также были рассмотрены системы VERMONT и OpenIMP, однако, по ряду причин они не были использованы: неэффективное сообщение между модулями систем, отсутствие поддержки двунаправленных потоков, недостающий функционал для детального анализа.

3.2. Сенсор

Как уже было сказано, одна из первых задач при доработке сенсора — это его расширение поддержкой IPFIX. Ранее упомянутая

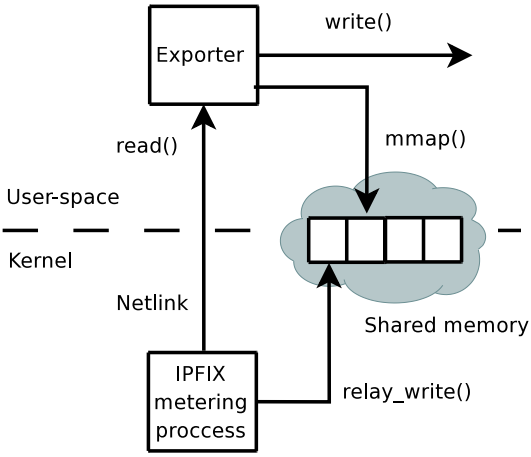

Рис. 2. Обмен информацией между сенсором и экспортером

библиотека libipfix была существенным образом изменена и портирована для работы в ядре Linux, а также расширена поддержкой двунаправленных потоков. API библиотеки также остался простым:

```
Инициализация шаблона templ с количеством элементов nele:

ipfix_new_data_template(struct ipfix_template **templ, int nele)

Добавление поля типа type к шаблону templ длины len;
если reverse > 0, тогда поле обратимо:

ipfix_add_field(struct ipfix_template *templ, int reverse,
                          uint16_t type, uint16_t len );

Экспорт согласно шаблону templ:

ipfix_export(struct ipfix_template *templ, ...)
ipfix_export_array(struct ipfix_template *templ, int nfields,
                   void **fields, u_int16_t *len)
```

Следующий вопрос — это передача информации из пространства ядра в пространство пользователя. Через большинство маршрутизаторов проходит огромное количество трафика, которое порождает множество потоков, поэтому здесь важен вопрос производительности. Точнее говоря, передача каждого IPFIX–сообщения отдельно — это достаточно высокие накладные расходы, которые, как правило,

сопровождаются дополнительной нагрузкой, связанной с лишним копированием информации из пространства ядра в пространство пользователя. Предлагается использовать область общей памяти, которая разделена на буферы. Возможно объединить эти буферы в кольцо, тогда в случае заполнения буфера, можно просто продолжать писать далее, не дожидаясь отстающего потребителя.

Эта схема была реализована с использованием подсистемы Relay ядра Linux. Однако, при использовании данной подсистемы, пользовательскому процессу необходимо было использовать поллинг во время ожидания готовности данных. Использование поллинга зачастую не самый эффективный метод ожидания готовности, поэтому был реализован протокол общения при помощи подсистемы Netlink ядра Linux. Пользовательский процесс засыпает в ожидании поступления данных, а в момент готовности одного из буферов получает сообщение через Netlink–сокет (см. рис. 2).

3.3. Экспортер

Экспортер был реализован как простое консольное приложение, которое не имеет входа, а всю получаемую из ядра информацию пересылает в стандартный поток вывода. Таким образом, пользователь свободен выбирать, что делать с получаемой информацией. В тривиальном случае можно просто перенаправить вывод приложения в файл для накопления. Данный файл будет содержать последовательность IPFIX–сообщений и сможет быть обработан любыми инструментами, соответствующими стандарту. Для решения проблемы с удаленной доставкой был использован протокол SSH, который предоставляет возможность обмена между двумя сетевыми устройствами по защищенному каналу. Для этого вывод приложения перенаправляется к SSH–клиенту, который устанавливает соединение с удаленным устройством и пересылает всю входную информацию по защищенному каналу.

3.4. Коллектор

В задачи коллектора входит получение IPFIX–сообщений, их архивирование для долгосрочного хранения и анализа, а также передача другим клиентам — различным приложениям, которые производят обработку поступающих потоков в реальном времени. Получаемые IPFIX–сообщения последовательно сохраняются в файл. Для

того чтобы была возможность проводить последующий анализ с помощью SiLK, используется утилита rwipfix2silk, которая дает возможность конвертировать формат IPFIX в представление SiLK.

Для реализации работы с клиентами была использована следующая схема: полученные IPFIX–сообщения коллектор сохраняет в двух буферах, расположенных в общей памяти. Коллектор последовательно переходит с одного буфера на другой при записи сообщений, а клиенты могут читать только тот буфер, в который данный момент ничего не записывается. Используются два семафора, по одному на буфер, для решения проблемы синхронизации. Смена буфера происходит в момент достижения конца буфера (сделан полный круг), либо по истечению некоторого промежутка (по умолчанию, 5 секунд).

Причина использования подобной схемы — производительность. Клиенты могут засыпать при чтении информации на неопределенный срок, что может стать «узким местом» системы. Также стоит отметить, что любой клиент может восстановить пропущенную информацию из архива.

4. Обнаружение сетевых аномалий

4.1. Актуальность

Сетевые аномалии и атаки уже давно стали обыденными явлениями в современных сетях, их четкая и быстрая идентификация важна для любой крупной сети. С очень быстрым ростом пропускной способности каналов и еще более быстрым появлением новых типов атак и вредоносных программ, большинство существующих систем определения атак (Intrusion Detection System, IDS) не могут более справляться со своей задачей по следующим причинам:

- многие IDS основаны на определение атак, направленных на некоторое выделенное устройство или их ограниченный набор; такие системы не масштабируются к большим сетям;
- определение атак во многих IDS основано на сигнатурах — типичном поведении известных вредоносных программ, что делает обнаружение неизвестных видов атак невозможным.

Поэтому является логичным пытаться определять атаки не на конечных и промежуточных устройствах, а на граничных маршрутизаторах, вне зависимости от их нагрузки. Также предлагается использовать IDS, основанные не на сигнатурном алгоритме работы,

а статистическом. Под статистическим алгоритмом работы понимается следующий принцип: через постоянные промежутки времени проводятся замеры некоторых параметров сетевого трафика (например, количество замеченных TCP или UDP пакетов). Предполагается, что полученный ряд отражает «нормальное» поведение. Далее можно пытаться искать отклонения текущего поведения от «нормального». Для этого разработано достаточно много различных подходов: [8],[9],[10]. Для использования был выбран метод Holt–Winters в силу того, что, во-первых, имеется его открытая реализация в пакете RRDtool, и, во-вторых, этот метод уже давно зарекомендовал себя на практике.

Стоит отметить, что попытка использования метода Holt–Winters для анализа трафика на базе NetFlow уже была [11]. Однако, эта система имеет следующие недостатки:

- Строится на базе коллектора nfcapd и графического интерфейса nfsen. Это означает отсутствие возможности использовать протокол IPFIX и, как следствие, двунаправленные потоки.
- Используется слишком грубая гранулярность. Анализ производится над слишком общими параметрами трафика, такими как количество замеченных байтов и пакетов с возможностью разделения по протоколам.
- Отсутствует система оповещения и анализа инцидента: известен лишь промежуток времени, в который произошел сбой.

4.2. Обзор алгоритма Holt–Winters

Многие временные ряды наделены следующими особенностями, которые должны учитываться при моделировании:

(1) изменение во времени (например, увеличение нагрузки на сервер, связанное с увеличением числа клиентов);
(2) наличие сезонных циклов (например, загрузка на сетевое оборудование утром возрастает, достигает пиковой нагрузки в 4–8 часов дня и спадает к полуночи);
(3) наличие сезонной изменчивости (например, количество запросов к серверу существенно меняется каждую минуту в пиковые часы, но в ночное время подобной динамики не наблюдается);

(4) постепенная эволюция величины за счет (1) — (3) на протяжение времени.

Пусть $y_1 \dots y_{t-1}, y_t, y_{t+1} \dots$ — это последовательность значений некоторой переменной величины (через фиксированные промежутки времени), а m — это период сезонной изменчивости (например, число замеров в день). Метод прогнозирования Holt–Winters основан на том, что наблюдаемый временной ряд может быть разложен на три составляющие: нормальная составляющая, линейное отклонение и сезонное отклонение. Предполагается, что каждая из этих компонент изменяется со временем и ее изменение можно найти за счет последовательного применения экспоненциального сглаживания. Прогноз значения величины — это сумма трех компонент: $\hat{y}_{t+1} = a_t + b_t + c_{t+1-m}$. Компоненты обновляются согласно формулам:

- $a_t = \alpha(y_t - c_{t-m}) + (1-\alpha)(a_{t-1} + b_{t-1})$ (нормальная компонента);
- $b_t = \beta(a_t - a_{t-1}) + (1-\beta)b_{t-1}$ (линейной отклонение);
- $c_t = \gamma(y_t - a_t) + (1-\gamma)c_{t-m}$ (сезонное отклонение).

Новая оценка для нормальной компоненты — это значение наблюдаемой величины с учетом наилучшей доступной оценки сезонного отклонения (c_{t-m}). Так как новая величина должна учитывать линейной отклонение, предполагаемая величина отклонения добавляется к оценке. Новое значение линейного отклонения — это разница между старой и новой оценкой для нормальной компоненты. Новая оценка для сезонного отклонения — это разница между наблюдаемым значением и соответствующей ему нормальной компонентой. α, β, γ — это адаптационные параметры алгоритма такие, что $0 < \alpha, \beta, \gamma < 1$. Большие значения приводят к более быстрой адаптации алгоритма, и на прогноз сильнее влияют недавние значения величины; при маленьких значениях возрастает значение прошлого.

4.3. Реализация

Описываемых подход основан на проведении последовательных замеров некоторых величин сетевого трафика с последующим помещение этой информации в базу RRDtool, системы для высокопроизводительной обработки и визуализации временных рядов. Далее, мы используем уже реализованный в пакете алгоритм Holt–Winters для обнаружения аномалий. Архитектуру системы можно видеть на рис. 3.

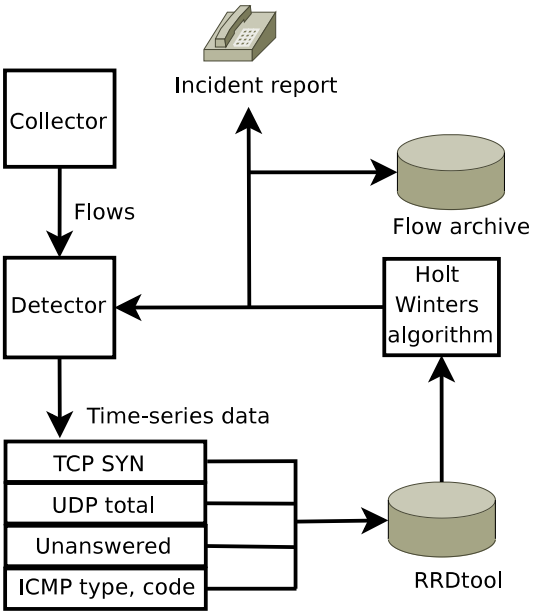

Рис. 3. Архитектура детектора

Модуль обнаружения в течении определенного промежутка времени принимает потоки. Для каждого потока мы проводим анализ: в первую очередь, мы определяем протокол и был ли поток отвеченным (за счет использования двунаправленных потоков). Таким образом, мы, например, может найти значение количества переданной информации с использованием конкретного протокола. Далее мы можем разделить это значение на две составляющие: «отвеченный» трафик и нет, и прогнозировать по обоим из этих величин для каждого протокола. В случае, если используется протокол ICMP, потоки разделяются согласно типу и коду ICMP–сообщения. Так, например, большое количество ICMP–сообщений типа “ICMP destination unreachable” или “ICMP port unreachable” могут свидетельствовать о попытке сканирования. Для протокола TCP проводится анализ, используя значение TCP–флагов соединения. Например, отдельно классифицируются потоки с единственным битом TCP SYN; это используется для определения типичной активности сетевых червей

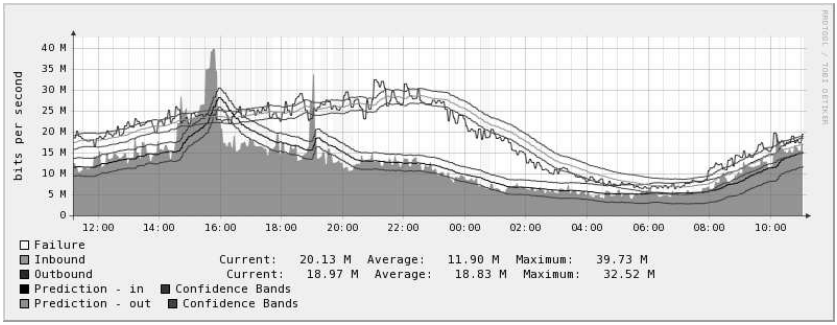

Рис. 4. Пример прогноза

(SYN–сканирования). Далее, каждую из величин мы можем разделить согласно направления трафика: внутренний (из внутренней сети во внутреннюю сеть), входящий (из внешней сети во внутреннюю), выходящий (из внутренней сети во внешнюю). Потоки принимаются в течении определенного времени (в данном случае, 5 минут), затем полученные итоговые значения для каждой из характеристик мы передаем для хранения в базу RRDtool.

Для определения аномального поведения используется скользящее окно, состоящее из 5 замеров: если обнаруживается ошибка в трех из пяти последовательных замеров, поведение величины считается аномальным. Пример получаемого прогноза можно видеть на рис. 4. На данном графике показана входная/выходная пропускная способность канала, занимаемая протоколом TCP. Выделены промежутки, в которых значение величины считается аномальным (ошибка в замере — это выход графика за «коридор», задаваемый алгоритмом прогнозирования).

В случае срабатывания модуля для одной из компонент автоматически генерируется отчет. Для этого проводится общий статистический анализ указанного промежутка с помощью информации, содержащейся в архиве потоков. Конкретная методика анализа зависит от компоненты, на которую пришлось срабатывание механизма. Например, при выявлении сбоев в TCP SYN–компоненте, соответствующие потоки агрегируются по IP–адресам получателя и отправителя с целью выяснения наиболее активных инициаторов соединений. Для проведения подобного анализа можно использовать систему

SiLK, которая дает большие возможности для получения произвольной статистической информации [7]. Отчет с полученными результатами может быть далее послан системному администратору.

Стоит отметить, что данный подход является очень гибким: пользователь свободен сам определить действия при срабатывании механизма; также есть возможность реализовать собственный алгоритм прогнозирования. В данном случае использовался алгоритм Holt–Winters, но одной из основных причин этого является его готовая реализация в пакете RRDtool. В будущем, возможна реализация и использование произвольных алгоритмов.

Задача выбора значений адаптационных параметров алгоритма α, β, γ не является простой. Не существует оптимального набора данных параметров, даже если речь идет об анализе одной переменной [12]. Обсуждение данного вопроса можно найти в [13]. Важен вопрос о периоде обучения системы: нельзя обучать систему на трафике, содержащем аномалии, так как это может привести к пропуску срабатываний в будущем. Как правило, недельного периода обучения достаточно для получения удовлетворительных результатов.

5. Выводы

В данной работе была представлена архитектура и реализация системы для учета и анализа трафика. Данная система удовлетворяет требованиям эффективности. В качестве примера использования системы было предложено высокопроизводительное решение для обнаружения сетевых аномалий. На данный момент система проходит тестовую эксплуатацию в сети телекоммуникации «Ботик». В планах на будущее стоит:

- расширение функциональности системы за счет реализации новых клиентских модулей;
- поиски узких мест и повышение эффективности системы;
- исследование новых методов обнаружения сетевых аномалий: использовать другие алгоритмы прогнозирования и новые параметры, возможно применить информационную модель IPFIX для реализации новых подходов.

Список литературы

[1] Михайлов А. Ю. Эффективный метод учета трафика в ОС Linux // Материалы XII научной студенческой конференции университета города Переславля им. А. К. Айламазяна. — г. Переславль-Залесский, 2008. ↑2

[2] Claise B. RFC 3954: Cisco Systems NetFlow Services Export Version 9, 2004, http://www.ietf.org/rfc/rfc3954.txt. ↑2

[3] Quittek J., Zseby T., Claise B., Zander S. RFC 3917: Requirements for IP Flow Information Export (IPFIX), 2004, http://www.ietf.org/rfc/rfc3917.txt. ↑2

[4] Claise B. RFC 5101: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information, 2008, http://www.ietf.org/rfc/rfc5101.txt. ↑2

[5] Claise B., Quittek J., Bryant S., Aitken P., Meyer J. RFC 5102: Information Model for IP Flow Information Export, http://www.ietf.org/rfc/rfc5102.txt. ↑2

[6] Trammell B., Boschi E. RFC 5103: RFC 5103: Bidirectional Flow Export Using IP Flow Information Export (IPFIX), 2008, http://www.ietf.org/rfc/rfc5103.txt. ↑2

[7] Gates C., Collins M., Duggan M., Kompanek A., Thomas M. More NetFlow Tools: For Performance and Security // In Proceedings of the 18th Large Installation Systems Administration Conference (LISA 2004). — Georgia, USA: USENIX Organization, 2004. — 121–132 с. ↑3.1, 4.3

[8] Rehak M. A. P. (M. and Bartos), Pechoucek M., Bartos K., Grill M., Celeda P., Krmicek V. CAMNEP: An intrusion detection system for high-speed networks, 2008, http://www.nii.ac.jp/pi/n5/5_65.pdf. ↑4.1

[9] Ertoz L., Eilertson E., Lazareciv A., Tan P., Kumar V., Sristava J., Dokas P. The MINDS — Minnesota Intrusion Detection System, 2004, http://www-users.cs.umn.edu/~kumar/papers/minds_chapter.pdf. ↑4.1

[10] Xu K., Zhang Z., Bhattacharyya S. Profiling Internet BackboneTraffic: behaviour Models and Applications, 2005. ↑4.1

[11] Mohacsi J., Kiss G. Anomaly detection for NFSen/nfdump netflow engine — with Holt-Winters algorithm, http://bakacsin.ki.iif.hu/~kissg/project/nfsen-hw/JRA2-meeting-at-Espoo_slides.pdf. ↑4.1

[12] Brutlag J. Abberant behaviour Detection in Time Series for Network Monitoring, 2000, http://www.usenix.org/events/lisa00/full_papers/brutlag/. ↑4.3

[13] Chatfield C., Yar M., Holt-Winters Forecasting: Some Practical Issues, 1988. ↑4.3

A. Y. Mikhailov. *Effective traffic measurement and accounting* // Proceedings of Junior research and development conference of Ailamazyan Pereslavl university. — Pereslavl, 2009. — p. 132–146. (*in Russian*).

Abstract. In this paper paper architecture and implementation of a network traffic measurement system proposed. Solution is based on IPFIX protocol. Application of the system to anomaly-based detection is given.