на правах рукописи

Маркин Юрий Витальевич

МЕТОДЫ И СРЕДСТВА УГЛУБЛЕННОГО АНАЛИЗА СЕТЕВОГО ТРАФИКА

Специальность 05.13.11 –

математическое и программное обеспечение

вычислительных машин, комплексов и компьютерных сетей

Автореферат

диссертации на соискание ученой степени

кандидата технических наук

Москва

2017

Работа выполнена в Федеральном государственном бюджетном учреждении науки Институте системного программирования РАН.

Научный руководитель: **Падарян Вартан Андроникович**, кандидат физико-математических наук

Официальные оппоненты: **Гергель Виктор Павлович**, доктор технических наук, профессор, декан факультета вычислительной математики и кибернетики, Нижегородского государственного университета имени Н. И. Лобачевского, директор Научно-исследовательского института прикладной математики и кибернетики, руководитель Приволжского научно-образовательного Центра суперкомпьютерных технологий

Волконский Владимир Юрьевич, кандидат технических наук, старший научный сотрудник, начальник отделения "Системы программирования" публичного акционерного общества "ИНЭУМ им. И.С. Брука"

Ведущая организация: Вычислительный центр им. А.А. Дородницына Российской академии наук Федерального исследовательского центра «Информатика и управление» Российской академии наук

Защита диссертации состоится 25 мая 2017 г. в 15 часов на заседании диссертационного совета Д 002.087.01 при Федеральном государственном бюджетном учреждении науки Институте системного программирования Российской академии наук по адресу: 109004, Москва, ул. Александра Солженицына, д. 25.

С диссертацией можно ознакомиться в библиотеке Федерального государственного бюджетного учреждения науки Института системного программирования Российской академии наук.

Автореферат разослан «__» _____ 2017 г.

Ученый секретарь
диссертационного совета Д 002.087.01,
кандидат физико-математических наук Зеленов С.В.

Общая характеристика работы

Активное развитие информационных технологий сделало их неотъемлемой частью быта, производства, сферы услуг. Информационные системы сегодня эксплуатируются как в коммерческих, так и в государственных организациях. Взаимодействие между такими системами осуществляется посредством глобальной сети. Наблюдается стремительный рост объема сетевого трафика, усложняется его структура. Анализ трафика становится все более востребованным в областях контроля и управления, оптимизации, защиты от вредоносных воздействий. Под *углубленным анализом* (DPI – Deep Packet Inspection) понимается анализ свойств полезной нагрузки или пакетов, более полных, чем информация заголовков протоколов уровней 2, 3 и 4 модели OSI.

Актуальность

Актуальными являются исследования в области обратной инженерии закрытых сетевых протоколов. Исходные коды таких приложений не публикуются в открытом доступе: разработчик предоставляет конечному пользователю программу в виде бинарного кода. В таких случаях необходимо проведение комплексного аудита безопасности кода на предмет поиска ошибок и уязвимостей, в том числе в реализации используемого протокола. В случае сетевого приложения вместе с бинарным кодом анализируется сетевой трафик компьютера (сетевого интерфейса), на котором это приложение запущено.

Важной практической задачей является разработка методов защиты внутренних сетей предприятий. Один из методов состоит в накоплении базы знаний о структуре проводимых атак и характерных сигнатурах передаваемых данных. Для случаев, когда факт проведения сетевой атаки установлен, необходимо провести расследование: выяснить, как развивался инцидент, какие данные были повреждены или считаны, какие взаимодействия с другими компьютерами происходили. Детальный анализ

подобных инцидентов в перспективе позволяет совершенствовать средства защиты (как программные, так и аппаратные) таким образом, чтобы блокировать определенные виды сетевых атак в режиме реального времени. Атаки, основанные на перекрытии IP-фрагментов или TCP-сегментов, не могут быть обнаружены путем анализа содержимого сетевых пакетов по отдельности. Для детектирования этих атак необходимо провести IP-дефрагментацию и восстановить TCP-потоки. Восстановление потоков данных, передаваемых между сетевыми приложениями, позволяет выявлять случаи распространения вредоносного ПО, а также запрещенного контента.

По статистике доля зашифрованного трафика существенно увеличилась за последние несколько лет: крупные сайты начали использовать протокол HTTPS по умолчанию. В связи с этим актуальной практической задачей является анализ трафика на предмет использования серверами уязвимых криптографических алгоритмов, а также недоверенных сертификатов безопасности.

Широкое распространение также получили туннельные протоколы: в частности, они используются для организации VPN. Особенность туннельных протоколов состоит в том, что потенциально, возможно построение туннеля (т.е. стека используемых протоколов) произвольной конфигурации. Анализ трафика вложенных туннелей позволит судить о безопасности такого рода соединений. При организации сетевого туннеля в общем случае может использоваться произвольный стек протоколов. В условиях отсутствия информации о стеке протоколов разбор трафика туннельных соединений возможен при наличии функциональности по распознаванию протоколов, формирующих стек.

Спектр инструментов, применяемых для решения практических задач, связанных с анализом трафика, очень широк: при этом каждый использует собственные разборщики трафика и оперирует над своим внутренним представлением для разобранных сетевых пакетов. Задачи обеспечения

безопасности сети, а также контроля качества связи решаются в online-режиме, тогда как расследование инцидентов нарушения информационной безопасности, обратная инженерия и отладка протоколов в offline-режиме. Существующие инструменты и применяемые в них методы фактически предназначены для проведения анализа только в одном из режимов.

Поскольку инструменты анализа совместно представляют собой сложный комплекс, для упрощения поддержки протоколов необходима унификация компонентов этих инструментов, отвечающих за разбор сетевых пакетов. При этом одни и те же разборщики должны применяться при проведении анализа в online и offline режимах.

Целью диссертационной работы является исследование и разработка методов и программных средств углубленного анализа сетевого трафика, позволяющих автоматизировать расширение их функциональности. Методы должны обеспечивать устойчивость к потере отдельных сетевых пакетов и переупорядочиванию пакетов при передаче.

Основные задачи

1. Разработать модель представления разобранных сетевых пакетов. Учесть в модели следующие особенности передачи данных по сети:
   - потеря/переупорядочивание отдельных пакетов;
   - сжатие и шифрование данных;
   - вложенное туннелирование.
2. Разработать алгоритм восстановления потоков данных для протоколов произвольного уровня, в том числе прикладного, устойчивый к потере отдельных сетевых пакетов, а также их переупорядочиванию.
3. Разработать архитектуру системы углубленного анализа сетевого трафика, позволяющую разрабатывать и отлаживать модули поддержки протоколов на предварительно сохраненном трафике и впоследствии использовать эти модули для разбора пакетов в режиме online.

4. Для получения экспериментальных оценок качества разбора сетевого трафика разработать и реализовать программные инструменты, базирующиеся на предложенных автором модели, алгоритме и архитектуре.

Научная новизна

Научной новизной обладают следующие результаты работы:

1. Модель представления данных позволяет единообразно описывать разбор заголовков произвольного стека сетевых протоколов.
2. Алгоритм восстановления потоков данных устойчив к переупорядочиванию и потере отдельных сетевых пакетов.
3. Архитектура системы позволяет использовать одни и те же разборщики заголовков сетевых протоколов при проведении анализа в online и offline режимах.

Теоретическая и практическая ценность работы

Разработаны модель представления данных и алгоритм восстановления потоков данных сетевых протоколов. Модель позволяет разделять и выполнять независимо фазы разбора и распознавания данных для произвольного стека сетевых протоколов.

Модель и алгоритм реализованы в виде динамической библиотеки и интерфейса для ее использования в составе программной системы анализа сетевого трафика. На базе предложенной модели разработаны и реализованы инструменты для проведения анализа сетевого трафика в online и offline режимах.

Инструмент для проведения анализа в online-режиме рассчитан на использование в сторонних системах. Инструмент для проведения отложенного анализа используется при решении задач, связанных с обратной

инженерией или отладкой сетевых протоколов, в области научных исследований и учебных курсах ВМК МГУ и ФУМП МФТИ.

Работа поддержана грантом РФФИ.

Методология и методы исследования

Применен формализованный подход, предполагающий использование математических объектов для представления сетевых данных. Математическую основу исследования составляют теория множеств, теория графов, теория алгоритмов, математическая логика, теория формальных языков и теория автоматов.

Апробация работы

Результаты работы обсуждались на следующих конференциях:

- XXIV Общероссийская научно-техническая конференция "Методы и технические средства обеспечения безопасности информации", Санкт-Петербург, 29 июня – 2 июля 2015 г.
- 58 Научная конференция МФТИ, Москва–Долгопрудный–Жуковский, 23 – 28 ноября 2015 г.
- Научно-практическая Открытая конференция ИСП РАН, Москва, 1 – 2 декабря 2016 г.

Публикации и личный вклад автора

По теме диссертации опубликовано 7 научных работ, в том числе научные статьи [1], [4], [6] в рецензируемых журналах из перечня рекомендованных ВАК РФ. В работе [1], в частности, представлен алгоритм обобщения форматов сообщений, разработанный автором лично. В работе [2] автором лично рассмотрены архитектура и область применения анализаторов сетевого трафика, проведено тестирование их функциональности по распознаванию протоколов. В статьях [4], [6] описана разработанная автором модель представления данных. В работе [5] автором лично обозначены

ограничения, присущие программным инструментам углубленного анализа сетевого трафика. В работе [7], в частности, описаны практические задачи анализа сетевого трафика, а также разработанные автором графические компоненты, позволяющие упростить решение этих задач.

Все представленные в диссертации результаты получены автором лично.

Объем и структура диссертации

Работа состоит из введения, пяти глав, заключения и списка литературы. Диссертация изложена на 80 страницах. Список литературы насчитывает 53 наименования. Диссертация содержит 10 таблиц и 31 рисунок.

Краткое содержание работы

Введение

Введение содержит цель диссертационной работы, а также обоснование ее актуальности. Формулируются основные результаты работы, рассматриваются практические возможности применения предлагаемых методов и реализованных инструментов.

Глава 1

Перечисляются особенности процесса передачи данных по сети с коммутацией пакетов. Формулируются требования к разбору сетевого трафика:

1. Проведение анализа с учетом возможной потери и/или переупорядочивания отдельных пакетов.
2. Анализ зашифрованных данных.
3. Поддержка вложенных туннельных протоколов.
4. Простота поддержки сетевых протоколов.

Были рассмотрены следующие свободно распространяемые сетевые анализаторы: Wireshark, Snort, The Bro Network Security Monitor (далее Bro), ntopng (nDPI).

Тестирование по восстановлению TCP-потоков показало, что Bro правильно восстанавливает потоки данных всех соединений, Snort не производит необходимого переупорядочивания, а также не отбрасывает пакеты, переданные повторно. Ntopng проводит разбор на уровне отдельных пакетов, не восстанавливая содержимое потоков. Если Wireshark удается определить размер единицы передачи данных вышележащего протокола (Protocol Data Unit, PDU) до обработки переупорядоченных сегментов с данными этой PDU, то восстановление потока выполняется правильно. В противном случае возникают неточности.

Инструменты Bro, Snort и ntopng не анализируют зашифрованный трафик (проводится разбор только незашифрованных частей пакетов) – при наличии ключей для дешифрования аналитик не сможет ими воспользоваться. Wireshark поддерживает возможность дешифровки.

В случае Snort при анализе вложенных туннельных протоколов возникают затруднения, обусловленные тем, что разобранный сетевой пакет на уровне ядра системы представляется посредством структуры с фиксированным набором полей, в частности IP-заголовков – не более двух. Причем значения этих полей не могут быть перезаписаны в процессе анализа. Таким образом, если в трафике встретится стек протоколов, в котором IP-заголовок представлен трижды, Snort не сможет корректно провести разбор.

В Bro в явном виде отсутствует структура для описания всех заголовков разобранного сетевого пакета. Процедура анализа подразумевает мониторинг соединений – пятерок вида {SRC_IP, DST_IP, SRC_PORT, DST_PORT, DIRECTION}. При этом для соединения сохраняется стек соединений, его инкапсулирующих. Таким образом, ключевая информация о туннелях, необходимая для проведения дальнейшего анализа, сохраняется.

Для описания сетевого пакета в ntopng применяется структура с фиксированным набором полей, что не позволяет осуществлять разбор произвольного стека протоколов.

В Wireshark разобранный сетевой пакет представляется посредством дерева, вершины которого описывают заголовки протоколов и выделенные в них поля. Такой подход позволяет анализировать вложенные туннельные протоколы.

Модули разбора протоколов всех уровней в Wireshark реализованы единообразно, чего нельзя сказать о Snort и Bro: если разборщики протоколов канального, сетевого и транспортного уровней жестко связаны посредством прямых вызовов, то анализ высокоуровневых протоколов

выполняется с учетом возможностей динамического распознавания (Bro) или задания диапазонов портов (Snort). Таким образом, для Bro и Snort поддержка нового низкоуровневого протокола потребует изменить существующий код.

Модули разбора в Wireshark не являются независимыми: SSL-разборщик использует информацию о сборке PDU из нескольких TCP-сегментов, HTTP-разборщику требуются функции, применяемые при анализе SSL-трафика и др. Подобная организация кода увеличивает его связность и затрудняет дальнейшую разработку: в случае изменения кода разборщика данного протокола необходимо проконтролировать корректность кода во всех зависимых разборщиках. Высокоуровневые декодеры Snort и Bro также зависят от низкоуровневых, но в значительно меньшей степени, нежели в Wireshark.

Разборщики ntopng реализованы в рамках одного модуля. Для протоколов прикладного уровня делаются неявные предположения об используемом транспортном протоколе.

Результаты обзора сетевых анализаторов представлены в табл. 1.

Табл. 1 – Сводная таблица с результатами обзора сетевых анализаторов.

	Wireshark	Snort	Bro	ntopng
Восстановление потоков	+/–	–/+	+	–
Анализ зашифрованных данных	+	–	–	–
Анализ вложенных туннелей	+	–	+	–
Добавление поддержки протоколов	–/+	+/–	+/–	+/–

Ни один из рассмотренных инструментов не соответствует предъявленным требованиям полностью. Поэтому для достижения поставленных целей необходимо разработать модель представления данных, а также систему анализа трафика на ее основе.

Основные требования к модели представления данных:

1. Восстановление (последующий разбор) потоков данных для случаев, когда порядок получения пакетов отличается от порядка их отправления и/или отдельные пакеты потеряны.
2. Поддержка сжатого и зашифрованного трафика.
3. Поддержка вложенных туннельных протоколов (с произвольной конфигурацией стека).

Ключевые требования к архитектуре системы:

1. Общая кодовая база: обеспечение переносимости модулей разбора между инструментами offline- и online-анализа.
2. Независимость модулей разбора на уровне исходного кода.

Глава 2

Вводится понятие логического соединения. Описывается модель представления данных, удовлетворяющая перечисленным ранее требованиям. На рис. 1 изображена диаграмма классов, с помощью которых представляется разобранный сетевой пакет.

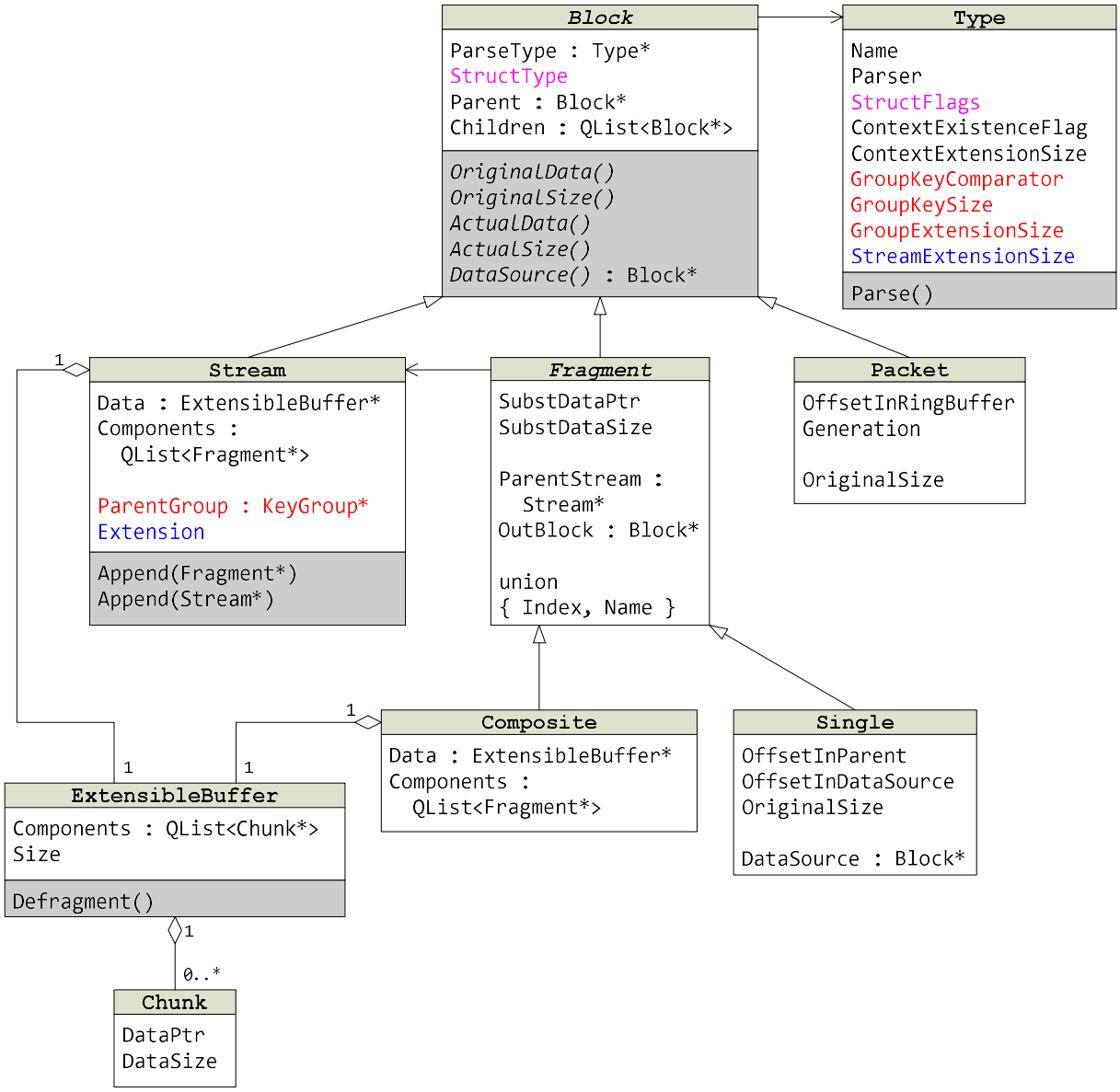

Рис. 1 – Диаграмма классов для описания разобранных сетевых пакетов.

Выделенные в процессе разбора структурные единицы (сетевые пакеты и их поля) описываются посредством *блоков (класс Block)*. Блок представляет массив байт некоторого фиксированного размера. При анализе блока может потребоваться отдельно обработать какую-либо его часть. Для этого будет создан новый блок, являющийся дочерним по отношению к исходному родительскому блоку. В процессе анализа каждому блоку ставится в соответствие *тип разбора (поле ParseType класса Block)*, определяющий функцию *(поле Parser класса Type)* для разбора данных этого блока. Разбор заключается в разделении буфера на именованные поля согласно заданному

формату пакета. Для блока определяется базовый формат данных *(поле StructType класса Block)*. Основные базовые форматы:

- *Struct* – блок содержит разнородные поля, аналог типа *struct* языка Си (*ProtoStruct* для случаев, когда описывается заголовок сетевого протокола);
- *FieldSeq* – последовательность однородных полей, аналог типа массив языка Си;
- *PacketSeq* – последовательность сетевых пакетов (возможно различных форматов) одного протокола.

Вводится два вида блоков: *блок-поток* – для описания восстановленных потоков данных, и *блок-фрагмент* – для представления целых пакетов и отдельных полей, выделенных в потоках данных. Восстановление потока подразумевает объединение полей (как правило, это поле с полезной нагрузкой) из разных сетевых пакетов. С этой целью вводится понятие *накопительного буфера (класс ExtensibleBuffer)*, позволяющего объединять как данные блоков-фрагментов, так и данные блоков-потоков. Отметим, что для блока-потока блок-родитель не определен, поскольку отношение «родитель-потомок» не применяется для восстановленных потоков.

Для размещения в памяти расшифрованных (или распакованных) данных вводится понятие *замещающего буфера*: его размер фиксирован, в отличие от накопительного буфера, увеличивающегося по мере добавления данных. Блок-фрагмент с непустым замещающим буфером будем называть *замещающим блоком-фрагментом*: дальнейшему разбору подлежат данные замещающего буфера. Дочерние блоки замещающего блока-фрагмента характеризуются смещением относительно соответствующего замещающего буфера. Отметим, что размер замещающего буфера никак не связан с размером исходных (зашифрованных или сжатых) данных блока-фрагмента. По определению, блок-поток не может содержать замещающий буфер.

Замещающий блок-фрагмент, таким образом, характеризуется первоначальными (зашифрованными) и фактическими (расшифрованными)

данными. Обобщая эти понятия для блока, будем называть *фактическими* данные блока, подлежащие дальнейшему разбору, и *первоначальными* данные без учета возможного замещения. Для блока-потока первоначальные данные совпадают с фактическими.

Вводится два вида блоков-фрагментов: *составной (класс Composite)* – для описания дефрагментированных PDU, и *одиночный (класс Single)* – для представления простых полей. Для краткости далее будем называть составной блок-фрагмент *составным фрагментом*, а одиночный блок-фрагмент – *одиночным фрагментом*. Дефрагментация PDU, как и восстановление потока, подразумевает объединение полей из разных пакетов, поэтому составной фрагмент владеет расширяемым буфером *(поле Data класса Composite)*. В то же время, составной фрагмент предназначен для описания одной PDU, тогда как в буфере блока-потока, как правило, содержится несколько PDU.

Будем называть *источником данных блока (поле DataSource класса Block)* блок, содержащий буфер, в котором содержатся первоначальные данные первого. Значит, в качестве источника данных может выступать блок-поток, составной фрагмент, замещающий блок-фрагмент. Одиночный блок локализуется в данных своего источника посредством смещения *(поле OffsetInDataSource класса Signle)*.

В соответствии со структурным типом родительского блока блок-фрагмент описывает либо поле структуры (характеризуется именем *Name*), либо элемент последовательности (характеризуется индексом *Index*).

На рис. 2 представлена диаграмма классов, посредством которых описываются логические соединения.

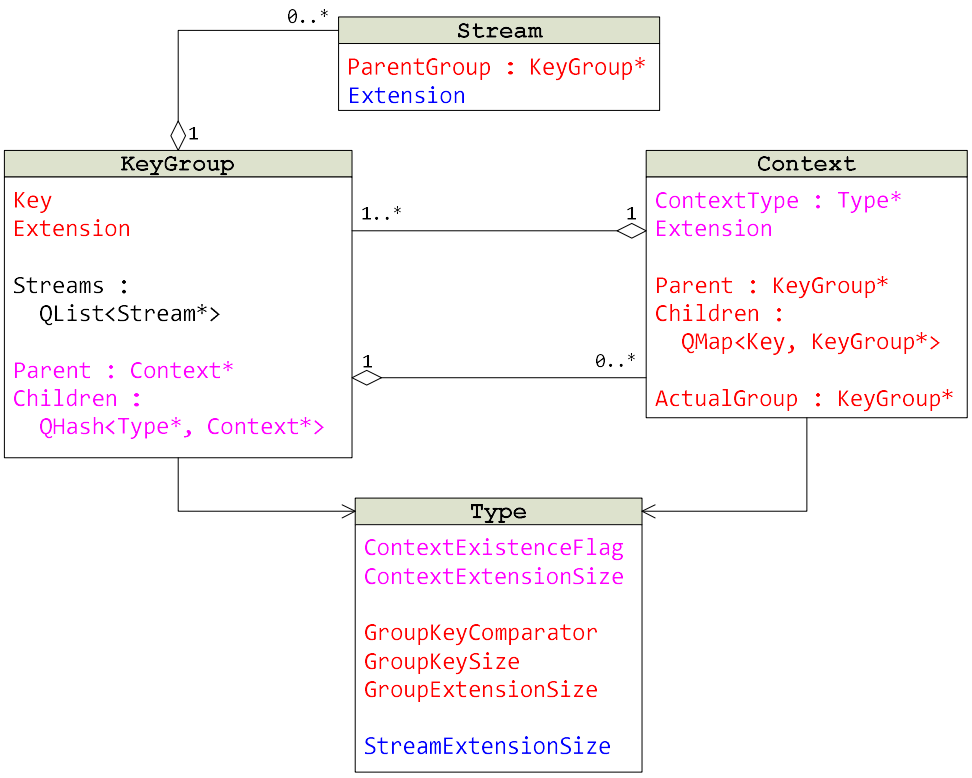

Рис. 2 – Диаграмма классов для описания логических соединений.

Каждое логическое соединение организовано посредством одного сетевого протокола и представляется в виде последовательности пакетов этого протокола. В то же время, на одном уровне стека протоколов могут быть представлены пакеты различных протоколов. Для выделения последовательностей пакетов, относящихся к заведомо различным логическим соединениям, необходимо сгруппировать пакеты заданного уровня стека в соответствии с протоколами этих логических соединений. Группа пакетов некоторого протокола в рамках заданного уровня стека описывается *контекстом (класс Context)*. Сетевой протокол, на основании которого был сформирован контекст, по определению задает *тип контекста (поле ContextType класса Context)*. В *расширении (поле Extension класса Context)* контекста может храниться дополнительная информация, необходимая для проведения разбора пакетов, относящихся к данному контексту (например, состояние декомпрессора при анализе пакетов протокола PPP). Тип контекста используется, в частности, при построении идентификаторов логических соединений.

Логическое соединение протокола некоторого уровня должно быть представлено последовательностью восстановленных пакетов. Для восстановления (извлечения полезной нагрузки) необходимо предварительно выделить соответствующую подпоследовательность из последовательности пакетов этого протокола. Для группировки пакетов в подпоследовательности используется адресная информация из служебных полей. Каждая подпоследовательность описывается *ключевой группой (класс KeyGroup)*. Такая группа в контексте идентифицируется посредством уникального ключа *(Key)*, содержимое и размер которого определяются в соответствии с типом контекста. Контекст выступает в роли контейнера по отношению к таким группам и предоставляет к ним доступ по ключу. Ключевая группа обладает *расширением (Extension)* для сохранения информации, необходимой при проведении разбора блоков (пакетов) этой группы. Ключевая группа фактически представляет собой логическое соединение.

Будем называть *восстановленным потоком протокола P* блок-поток, в накопительный буфер которого добавлены данные блоков-фрагментов одной ключевой группы (возможно переупорядочивание и/или перекрытие блоков-фрагментов, в соответствии с семантикой протокола *P*) контекста с типом, описывающим протокол *P*. Соответствующая ключевая группа называется *родительской (поле ParentGroup класса Stream)* по отношению к восстановленному потоку. В рамках одной ключевой группы поддерживается одновременное существование нескольких потоков *(поле Streams класса KeyGroup)* – необходимость в этом может быть обусловлена как семантикой соответствующего протокола, так и потерей отдельных пакетов при передаче.

В результате восстановления логических соединений различных взаимодействующих по сети приложений будет построено дерево контекстов, ключевых групп и восстановленных потоков (далее, *дерево контекстов*) с вершинами трех типов:

- вершина-контекст задает протокол;
- вершина-группа характеризуется ключом и описывает логическое соединение на уровне протокола родительской вершины;
- вершина-поток описывает восстановленный поток данных протокола вышележащего уровня.

На рис. 3 приведен пример дерева контекстов, в котором в качестве протокола нижнего уровня, наблюдаемого в сетевом соединении, фигурирует Ethernet. Ключ для Ethernet-группы строится путем конкатенации MAC-адресов отправителя и получателя (размер ключа равен 12 байтам). Для протокола IPv4 ключ группы состоит из пары IP-адресов (8 байт), для протокола TCP – из пары портов (4 байта).

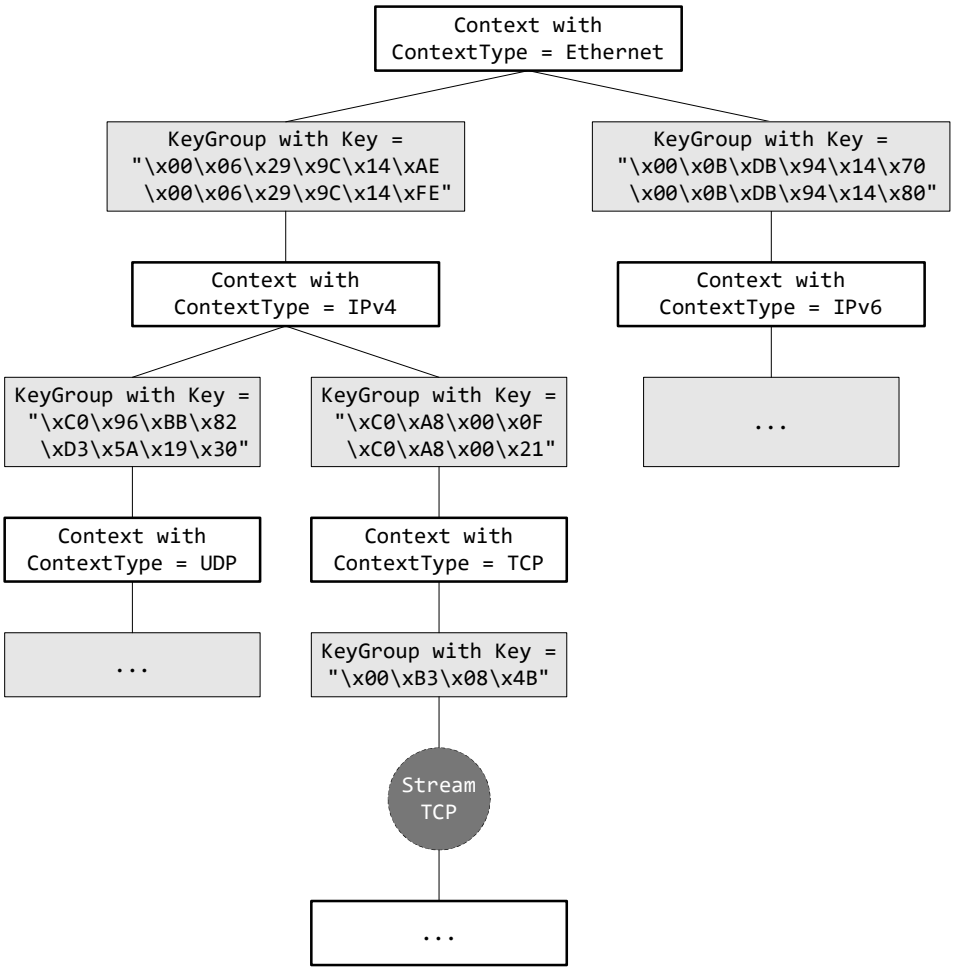

Рис. 3 – Пример дерева контекстов.

Для обеспечения разбора пакетов произвольного стека протоколов вводится понятие распознавателя. *Распознаватель* – это функция, которая по данным блока и, возможно, некоторой дополнительной информации определяет тип разбора этого блока. Обычно функциональность по распознаванию требуется для полей в заголовках протоколов, допускающих данные различных форматов.

Например, полезная нагрузка в пакете протокола IPv4 может содержать пакеты протоколов TCP, UDP, GRE, ICMP и др. Служебное поле, определяющее вышележащий протокол, содержится в IPv4-заголовке. Таким образом, для определения типа разбора полезной нагрузки используются данные блока-родителя. Соответствующую категорию распознавателей будем называть *распознавателями полей*.

Для определения типа разбора восстановленных потоков используется другая категория распознавателей – *распознаватели потоков с известным типом сборки*. Распознаватель получает на вход содержимое накопительного буфера потока и информация о типе разбора блоков, данные которых были добавлены в этот накопительный буфер.

Распознаватели, принимающие на вход только данные буфера, будем называть *универсальными*. Такие распознаватели применяются к блоку-потоку, если его тип сборки неизвестен, а также блоку-фрагменту, при условии, что с помощью распознавателей полей не удалось определить тип разбора.

Будем называть поток восстановленным *полностью*, если выполнены следующие три условия:

1. поток обладает признаком начала (например, флаг SYN для TCP),
2. поток обладает признаком конца (например, флаг FIN для TCP),
3. буфер потока содержит все данные потока без пропусков.

В противном случае будем называть поток *частично* восстановленным.

Будем называть блок b_1 *префиксом* блока b_2, если выполнены следующие три условия:

1. b_1 и b_2 принадлежат одной ключевой группе
2. b_1 и b_2 описывают полезную нагрузку некоторого протокола P
3. в соответствии с семантикой протокола P данные b_1 непосредственно предшествуют данным b_2

На рис. 4 приведено формальное описание алгоритма восстановления потоков (для упрощения опущены детали обработки пакетов-дубликатов).

```
Пусть M - множество частично восстановленных потоков ключевой
группы G.

для каждого блока-фрагмента f группы G {
  для каждого блока-потока s из M {
    если s является префиксом f {
      добавить данные f в s

      для каждого блока-потока s' из M\s {
        если s является префиксом s' {
          добавить данные s' в s
          выход из цикла
        }
      }

      если s восстановлен полностью {
        удалить s из M
        выполнить разбор s
      }

      выход из алгоритма
    }
  }

  создать блок-поток s
  добавить данные f в s
  добавить s в M
}

выход из алгоритма
```

Рис. 4 – Формальное описание алгоритма восстановления потоков.

При использовании такого алгоритма не возникает неопределенности, к какой PDU следует отнести пакет, полученный в неправильном порядке: решение этого вопроса откладывается до тех пор, пока не будет обработан пакет, непосредственно предшествующий данному, или выяснится, что этот

пакет в трафике отсутствует. Предложенный подход требует большего количества физической памяти по сравнению с попыткой определить границы PDU сразу (в момент разбора пакета), однако и результат анализа становится более точным.

Представление разобранного сетевого пакета в виде дерева блоков, а также механизм распознавателей позволяют анализировать трафик вложенных туннелей любой конфигурации. Благодаря замещающим блокам-фрагментам описываются модификации (дешифрование, распаковка) данных. Введенные понятия контекста и ключевой группы позволяют единым образом описывать разбор пакетов протоколов с сохранением состояния, а также выделять множества пакетов для последующего восстановления потоков.

Глава 3

Описывается схема построения системы анализа. Предложенная модель представления данных реализована в виде ядра системы, которое компилируется в динамическую библиотеку. На базе API ядра построена работа модулей разбора и распознавания данных. Было разработано два инструмента – для проведения online- и offline-анализа соответственно, – причем оба используют единую инфраструктуру (рис. 5). В зависимости от установленных параметров проводится сборка ядра и модулей разбора либо для offline-анализатора, либо для online-анализатора. Построенная по такому принципу система позволяет в полной мере использовать преимущества offline-анализа при разработке (отладке) модулей разбора и распознавания и дальнейшей эксплуатации этих модулей (используется тот же исходный код) при работе online.

Главное отличие между offline- и online- версиями ядра заключается в механизме управления памятью: при работе online необходимо следить за своевременным освобождением памяти, поскольку анализу подвергается потенциально бесконечный поток данных. Инструмент анализа сетевых трасс

обладает графическим интерфейсом, отдельные компоненты которого предназначены для решения конкретных классов задач: локализация и детализация сетевых соединений, обратная инженерия протоколов (отладка разборщиков), интерактивный разбор и др.

Система является модульной, как и большинство анализаторов сетевого трафика: для каждого протокола создается отдельный модуль, в котором локализована функциональность по работе с этим протоколом. Каждый из инструментов использует свой комплект модулей разбора. Важно, что используется один и тот же исходный код. В рамках разработанной системы модули разбора также являются независимыми: при добавлении новых разборщиков не требуется вносить изменения в существующие. Такая независимость достигается благодаря механизму связывания разборщиков, основанному на применении распознавателей.

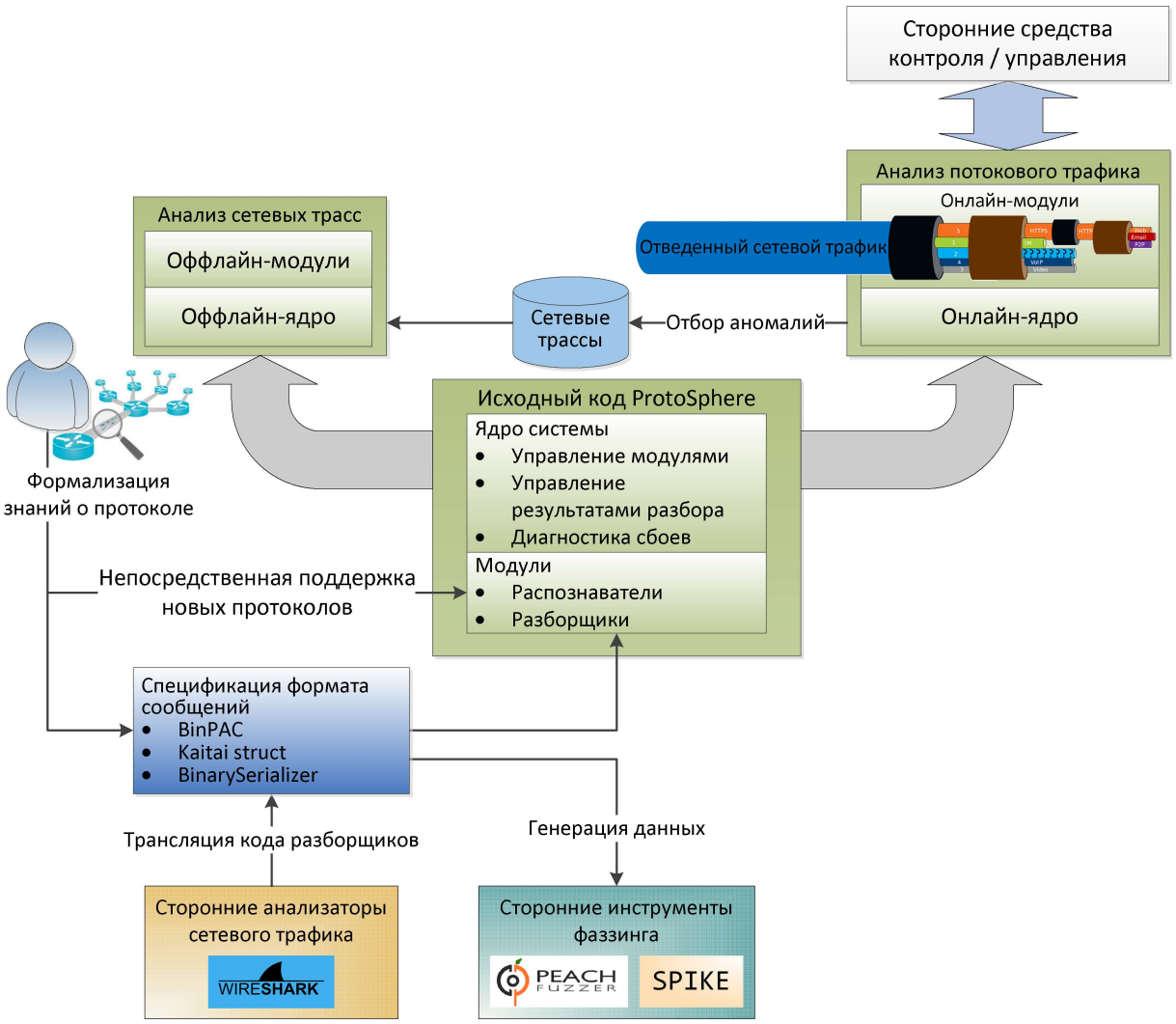

Рис. 5 – Структурные компоненты системы.

Для увеличения базы поддерживаемых протоколов ведется разработка транслятора исходного кода разборщиков Wireshark в промежуточное представление, далее транслируемое в исходный код разработанной системы. Основная трудность здесь состоит в существенных различиях между используемыми интерфейсами: невозможно построить отображение методов одного в методы другого. В качестве языка промежуточного представления для описания формата заголовка протокола на основе проведенного сравнительного исследования был выбран BinPAC, применяемый в системе The Bro Network Security Monitor.

Глава 4

Описываются инструменты для проведения online- и offline-анализа.

Online-анализатор

Инструмент online-анализа предназначен для извлечения данных из трафика в режиме online: он должен работать непрерывно с производительностью, достаточной для разбора пакетов, поступающих на сетевой интерфейс (потенциально бесконечный входной поток данных).

Инструмент состоит из трех компонентов:

- Модуль *listener* осуществляет взаимодействие с сетевым интерфейсом и сохраняет поступающие пакеты в кольцевой буфер;
- Модуль *kernel* выполняет разбор пакетов кольцевого буфера и извлекает необходимые данные;
- Модуль *saver* сохраняет извлеченные данные в файлы на жестком диске (формат файла определяется модулем построения).

В настоящий момент компонент *listener* может получать пакеты либо с помощью API библиотеки Pcap, либо посредством ZeroMQ-сокета. Пакеты записываются в буфер, организованный в виде односвязного кольцевого списка блоков памяти фиксированного размера. Память каждого такого блока заполняется последовательно. Если для очередного пакета не хватает места, для записи выбирается следующий блок. Операции записи и чтения буфера являются потокобезопасными (*thread-safe*).

В режиме online важна высокая скорость обработки: если сетевые пакеты поступают быстрее, чем обрабатываются, то проводить анализ бессмысленно. Операция копирования данных занимает значительное время, поэтому при чтении из буфера не выполняется – вместо этого блок памяти, содержащий пакет, блокируется на чтение. Для записи при необходимости будет выбран следующий блок.

Для поддержания целостности анализируемых данных с каждым блоком памяти кольцевого буфера хранится количество перезаписей его нулевого байта, или *поколение* этого блока. Всякий раз, когда нулевой байт некоторого

блока перезаписывается, его поколение инкрементируется. Компонент *listener* запоминает значение поколения при записи и отправляет его вместе с уведомлением об очередном сохраненном пакете. Получив уведомление, модуль *kernel* проверяет, изменилось ли поколение блока памяти, содержащего пакет: разбор проводится только при совпадении поколений, в противном случае обрабатывается следующее уведомление.

При разборе потенциально бесконечного потока данных необходимо явно ограничивать размер доступной анализатору физической памяти: в противном случае свободная память закончится, и анализатор в лучшем случае завершит работу аварийно. Физическая память требуется для хранения потоков, восстановление которых еще на завершилось, и, как следствие, дерева контекстов. Предложена следующая стратегия освобождения памяти: при достижении порогового значения удалять первыми объекты (потоки, ключевые группы, контексты), которые дольше других не обновлялись. Такая стратегия реализована посредством разработанного *generic*-контейнера *PriorityStorage*.

Online-анализатор отдельно сохраняет (в файл) пакеты, разбор которых происходит с ошибками. Под *ошибкой* в данном случае понимается несоответствие фактических данных описанию протокола в модуле разбора. Если количество таких ошибок превышает заданное пороговое значение, велика вероятность того, что соответствующий модуль разбора содержит неточности. В таком случае необходимо проанализировать сохраненные пакеты с помощью offline-анализатора, после чего внести в код модуля разбора (общий для двух инструментов) необходимые правки.

Инструмент online-анализа обладает интерфейсом для подключения сторонних анализаторов. Результаты разбора сетевых пакетов могут использоваться в различных целях, например, для проверки выполнимости правил, описывающих допустимый трафик.

Offline-анализатор

Инструмент offline-анализа главным образом предназначен для разработки и отладки модулей разбора: с его помощью аналитик получает максимально детализированную "картину происходящего" в сетевой трассе с указанием возникших в процессе разбора ошибок. Инструмент обладает графическим интерфейсом, позволяющим проследить за тем, как происходит восстановление потоков высокоуровневых данных (например, файлов формата PNG или PDF) из сетевых пакетов. Также предусмотрены механизмы поиска по трафику и навигации между различными представлениями результатов разбора.

Центральное место занимают компоненты отображения дерева блоков и журнала ошибок разбора. Дерево блоков представляет собой результат разбора одного блока-потока: корень описывает блок-поток, остальные вершины – выделенные блоки-фрагменты (сетевые пакеты и поля в них). Дерево блоков синхронизировано с компонентом отображения дампа: при выделении блока в дереве соответствующие ему байты в дампе подсвечиваются. Для блока-потока можно просмотреть список блоков, данные которых сформировали его накопительный буфер.

Компонент отображения журнала ошибок синхронизирован с деревом блоков – это позволяет быстро локализовать блок, разбор данных которого привел к возникновению ошибки.

Тип разбора не всегда удается определить с помощью распознавателей: применяемые эвристики могут не учитывать всех возможных вариантов содержимого сетевого пакета (особенно, если для протокола отсутствует документация). Более того, система может определить тип разбора неверно, если критерий соответствия типу сформулирован недостаточно полно. В таких случаях аналитик может указать системе, как нужно разбирать тот или иной блок, предоставив файл с подсказками. В нем для заданных блоков тип

разбора определен непосредственно. С помощью подсказок реализована работа с зашифрованными данными: для выбранной ключевой группы в подсказку записывается путь к файлу, содержащему необходимую для дешифрования информацию – как правило, это ключ шифрования. В процессе разбора ключ считывается из файла, данные блоков дешифруются, и анализ продолжается.

Реализовано два способа визуализации сетевых взаимодействий:

- граф оконечных узлов (*Endpoints*);
- граф, детализирующий сетевые взаимодействия выбранного оконечного узла (*Nodes*).

Граф *Endpoints* отображает сетевые соединения, относящиеся к протоколу самого низкого уровня в анализируемом файле. В качестве вершин здесь могут выступать MAC- или IP-адреса. Ребра соединяют сетевые узлы (отправителей и получателей), между которыми был передан хотя бы один сетевой пакет. Граф *Nodes* детализирует взаимодействия заданного оконечного узла. Для каждого взаимодействия отображается весь стек сетевых протоколов. Дополнительно осуществляется фильтрация графа *Nodes* по данным сетевых пакетов, передаваемых между узлами.

Глава 5

Представлены примеры практического применения инструментов. Для offline-анализатора демонстрируется применение разработанного алгоритма восстановления потоков: восстанавливается TCP-поток, пакеты которого при передаче были переупорядочены. Дальнейший разбор этого потока позволяет провести сборку HTML-страницы из данных HTTP-пакетов.

Второй пример демонстрирует возможность анализа зашифрованных данных с использованием подсказки: записи протокола SSL (TLS) дешифруются с помощью закрытого ключа SSL-сервера, прочитанного из файла.

Для online-анализатора показана возможность извлечения из трафика файлов в формате PNG.

Последним рассмотрен пример с разработкой разборщика закрытого протокола, используемого при организации ботнета.

Заключение

В заключении формулируются результаты, а также определяются направления дальнейшей работы. Представленная система может успешно применяться при анализе трафика как в online-, так и в offline-режиме. Предложенная модель описания данных охватывает существующие особенности передачи сетевого трафика. Разработанный алгоритм восстановления потоков устойчив к потерям отдельных сетевых пакетов и их переупорядочиванию.

Взаимодействие между узлами сети, как правило, не ограничивается одним потоком данных. Например, взаимодействие клиента с FTP-сервером предполагает организацию как минимум двух таких потоков: для отправки команд на сервер (управляющий поток) и для передачи файлов клиенту (поток передачи контента). Аналитику может быть полезна функциональность по выявлению связей между такими потоками: она позволит рассматривать взаимодействие между узлами сети на качественно более высоком уровне.

Для формального описания разборщиков применяется язык C++: аналитик непосредственно реализует функции разбора и распознавания данных. В то же время функции разбора могут быть сгенерированы автоматически на основе некоторого представления структуры сообщения протокола. Разработка и внедрение высокоуровневого декларативного языка описания формата позволит ускорить процесс разработки разборщиков. Формальное описание структуры также может использоваться для генерации трафика с заданными характеристиками.

Основные результаты работы

1. Разработана модель представления разобранных сетевых пакетов. В модели учтены следующие особенности передачи данных по сети:
   - потери/переупорядочивание отдельных пакетов;
   - сжатие и шифрование данных;
   - вложенное туннелирование.
2. Разработан алгоритм восстановления потоков данных для протоколов произвольного уровня, в том числе прикладного, устойчивый к потерям отдельных сетевых пакетов, а также их переупорядочиванию.
3. Разработана архитектура системы углубленного анализа сетевого трафика, позволяющая разрабатывать и отлаживать модули поддержки протоколов на предварительно сохраненном трафике (offline) и впоследствии использовать эти модули в режиме online.
4. На основе предложенных автором модели, алгоритма и архитектуры разработаны и реализованы программные инструменты для проведения углубленного анализа сетевого трафика как в режиме online, так и в отложенном режиме.

Публикации по теме диссертации

[1] А.И. Гетьман, Ю.В. Маркин, В.А. Падарян и др. Восстановление формата данных. // Труды Института системного программирования, том 19, 2010, с. 195-214.

[2] Ю.В. Маркин, А.С. Санаров. Обзор современных инструментов анализа сетевого трафика. // Препринты ИСП РАН, №27, 2014.

[3] Ю.В. Маркин, В.А. Падарян, А.Ю. Тихонов. Программная инфраструктура для глубокого анализа сетевого трафика. // Материалы 24-й научно-технической конференции «Методы и технические средства обеспечения безопасности информации». Санкт-Петербург, 29 июня - 02 июля 2015. Стр. 87-89

[4] А.И. Гетьман, Ю.В. Маркин, В.А. Падарян, А.Ю. Тихонов. Модель представления данных при проведении глубокого анализа сетевого трафика. // Труды Института системного программирования, том 27, выпуск 4, 2015.

[5] Ю.В. Маркин, В.А. Падарян, А.Ю. Тихонов. Ограничения современных программных средств глубокого анализа сетевого трафика и способы их преодоления. // Материалы 58-й научной конференции МФТИ. Москва–Долгопрудный–Жуковский, 23 - 28 ноября 2015 г.

[6] A.I. Get’man, V.P. Ivannikov, Yu.V. Markin, V.A. Padaryan, A.Yu. Tikhonov. Data representation model for in-depth analysis of network traffic. // Programming and Computer Software, Volume 42, Issue 5, 2016, pp 316-323.

[7] А.И. Гетьман, Ю.В. Маркин, Д.О. Обыденков, В.А. Падарян, А.Ю. Тихонов. Подходы к представлению результатов анализа сетевого трафика. // Труды первой научно-практической Открытой конференции ИСП РАН, 2016, стр. 309-315.