Обзор современных инструментов анализа сетевого трафика

Маркин Ю. В., Санаров А. С.
{ustas, intkecsk}@ispras.ru

Аннотация. Данная статья представляет собой обзор популярных в настоящий момент анализаторов сетевого трафика. Описаны архитектурные особенности инструментов, выделены основные достоинства и недостатки с точки зрения функциональности и удобства использования. Проведено тестирование на предмет распознавания протоколов и восстановления потоков протокола TCP. Результаты представлены в сравнительных таблицах.

Ключевые слова: сниффер; распознавание сетевых протоколов; восстановление сессий; система обнаружения вторжений

1. Введение

Задача анализа сетевого трафика приобретает все большую актуальность в связи с развитием и внедрением новых сетевых технологий (и, как следствие, увеличением объема данных, передаваемых по сети), а также появлением большого количества новых сетевых протоколов прикладного уровня. В качестве наиболее популярных областей практического применения можно выделить:

- анализ трафика с целью выявления проблем в работе сети (в том числе, несанкционированной активности);
- восстановление потоков данных («прослушивание»);
- предотвращение различного рода сетевых атак;
- сбор статистики.

Если говорить о комплексном решении задачи анализа сетевого трафика, то в первую очередь следует разделить ее на три в достаточной степени независимые подзадачи (рис. 1): перехват трафика, его хранение и анализ.

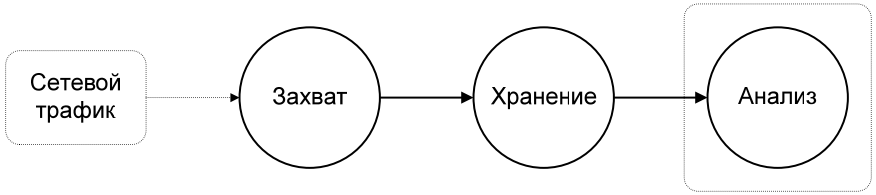

Рис. 1 – Подзадачи системы анализа сетевого трафика.

Система анализа должна обеспечивать захват 100% трафика, а также предоставлять эффективные методы анализа и навигации по его результатам.

Захват трафика осуществляется посредством снифферов. В общем случае, сниффер – это программа или программно-аппаратное устройство, предназначенное для перехвата трафика. В рамках конкретных продуктов могут быть реализованы дополнительные возможности, например, разбор заголовков сетевых протоколов, фильтрация по заданным критериям, восстановление сессий. Перехват сетевого трафика может осуществляться:

- с помощью «прослушивания» сетевого интерфейса;
- подключением сниффера в разрыв канала;
- посредством анализа побочных электромагнитных излучений;
- через атаку на канальном или сетевом уровне, приводящую к перенаправлению трафика жертвы на сниффер.

Сниффер может быть установлен как на маршрутизаторе, так и на оконечном узле сети.

Задачи перехвата и (эффективного) хранения трафика выходят за рамки данного обзора. Что касается задачи анализа, то предпочтение тому или иному инструменту отдается исходя из специфики подзадач, которые необходимо решить. Большинство существующих инструментов, как правило, проводит разбор заголовков сетевых протоколов, а также восстанавливает сессии (базовый анализ). В то же время существуют достаточно специфические задачи, для которых может не оказаться готового инструмента, например:

- анализ туннелированных протоколов произвольной глубины;
- анализ сессий на уровне приложений (выделение связей между потоками данных, передаваемых по сети);
- выполнение определенных сценариев (скриптов) в случае обнаружения в трафике предварительно заданных сигнатур.

Цель данного обзора – выяснить, какие задачи анализа сетевого трафика способны решать существующие (как свободно распространяемые, так и коммерческие) инструменты, насколько расширяема их функциональность.

Прежде, чем перейти к непосредственному рассмотрению сетевых анализаторов, отметим, что выделяют два режима их работы:

- в реальном времени;
- по предварительно сохраненному трафику.

Анализ в реальном времени требует поддержки работы инструмента в непрерывном режиме с производительностью, достаточной для разбора трафика, поступающего на вход. При этом должна быть обеспечена возможность обработки потенциально бесконечного входного потока данных.

В случае отложенного анализа инструмент получает входные данные из файла, что позволяет проводить более детальный анализ сетевой трассы по сравнению с анализом в режиме реального времени на аналогичном трафике.

2. Методика тестирования

Для каждого рассматриваемого инструмента оценивается как возможность проведения анализа трафика в реальном времени, так и отложенный анализ предварительно сохраненных сетевых трасс. Тестирование на трафике в реальном времени существенно зависит от самого трафика. В то же время, для сравнения результатов рассматриваемых инструментов необходимо обеспечить одинаковый трафик на сетевом интерфейсе. Для этого используется программа Colasoft Packet Player [1], позволяющая «воспроизвести» на заданном сетевом интерфейсе содержимое предварительно сохраненной трассы. Более того, Packet Player позволяет выдерживать соответствующие трассе временные интервалы между отправкой сетевых пакетов на интерфейс. Таким образом, можно получить объективные сравнительные результаты работы инструментов при анализе трафика в реальном времени. Что касается тестирования на сохраненных сетевых трассах, то для всех инструментов используется один и тот же набор трасс.

Для каждого инструмента проводится ряд испытаний, позволяющих оценить его возможности:

- восстановление потоков и выделение связей между ними
- распознавание протоколов (на тестовом наборе трасс Wireshark Trace Files [2])

Результаты тестирования представлены в сравнительных таблицах. Кроме того, оцениваются возможности навигации по результатам анализа (туннелированного) трафика.

3. Инструменты анализа сетевого трафика

Для каждого инструмента приводится описание архитектуры (если оно известно), а также основные достоинства и недостатки с точки зрения функциональности, удобства использования, производительности.

3.1 Свободно распространяемые инструменты

3.1.1 Wireshark

Wireshark ([3]) – это программа, которая поддерживает разбор большого количества различных сетевых протоколов (полный список приведен на сайте разработчика [4]), а также предоставляет возможность сортировки и фильтрации трафика. Разработчик – The Wireshark Team, лицензия – GNU General Public License. Wireshark поддерживает следующие ОС: Unix (Apple Mac OS X, FreeBSD), Linux (Ubuntu, Gentoo, Red Hat), Microsoft Windows (XP, Vista, 7). Код Wireshark написан на языке C. Wireshark позволяет пользователю просматривать и сохранять весь проходящий по сети трафик в режиме реального времени (для этого необходимо дополнительно установить библиотеку WinPcap (libpcap)). Кроме того, в рамках ОС Windows существует возможность (посредством специального адаптера) перехвата и анализа трафика беспроводных сетей Wi-Fi (802.11 WLAN).

На рис. 2 представлена архитектура инструмента. Wireshark состоит из двух компонентов:

1. библиотека, посредством которой осуществляется перехват сетевого трафика (WinPcap для ОС Windows, libpcap для ОС Linux);
2. прикладная программа, предоставляющая функции разбора протоколов и графический интерфейс для визуализации результатов анализа и взаимодействия с системой.

Посредством WinPcap (libpcap) осуществляется взаимодействие с драйвером сетевого интерфейса. Непосредственное взаимодействие с драйвером позволяет перехватывать и внедрять сетевые пакеты, а также применять различные критерии фильтрации. Библиотека libwiretap используется для работы (чтение/запись данных) с сетевыми трассами различных форматов (в том числе pcap). Компонент Dumpcap осуществляет запись перехваченных сетевых пакетов в файл, а также передает их на обработку. Библиотека libwireshark отвечает за разбор и последующий анализ сетевых пакетов, а также за визуализацию результатов.

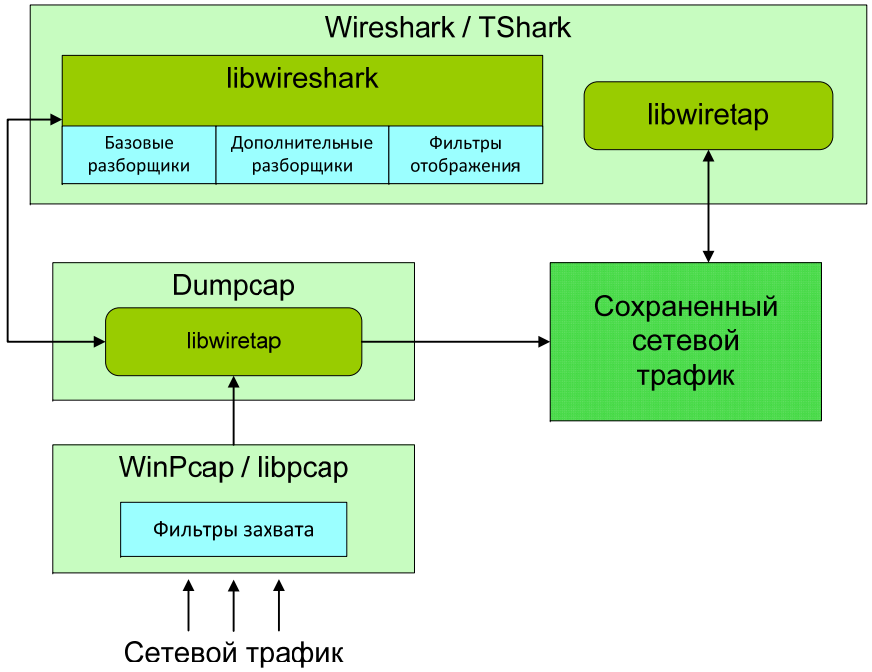

Рис. 2 – Архитектура Wireshark.

Особое внимание необходимо уделить механизму связывания разборщиков (функций разбора). Wireshark предоставляет три сценария взаимодействия:

- прямой вызов (непосредственный вызов разборщика)
- обратный вызов (вызываемый разборщик определяется значением некоторого ключевого поля, полученным при разборе более низкоуровневого протокола; например, значение поля «Порт» в заголовке TCP пакета)
- эвристическая привязка (вызываемый разборщик определяется путем поиска паттернов в анализируемом буфере)

Wireshark предоставляет возможность для подключения дополнительных (в том числе самостоятельно разработанных) модулей разбора трафика.

Основные достоинства инструмента:

1. Поддержка большого количества сетевых протоколов (в том числе протоколов IP-телефонии);
2. Поддержка различных форматов сетевых трасс;
3. Расширяемость (возможность создания и подключения

дополнительных модулей разбора);

4. Детальная система фильтрации сетевых пакетов;
5. Возможность восстановления потоков TCP;

Недостатки:

1. Восстановленный поток не рассматривается инструментом как единый буфер памяти, вследствие чего его дальнейшая обработка невозможна;
2. Код модулей разбора содержит функции, отвечающие за визуализацию результатов (логика разбора перемешивается с логикой отображения в графическом интерфейсе);
3. Отсутствует возможность выполнения некоторого действия в случае обнаружения сигнатур в трафике.

Wireshark поддерживает анализ туннелированного трафика, однако предоставляемая компонента отображения результатов является крайне неудобной. Дело в том, что каждый из последовательно применяемых к сетевому пакету разборщиков перезаписывает информацию в главном окне. В то же время при анализе туннеля необходима визуализация результатов всех разборщиков.

Приведенные недостатки не позволяют применять Wireshark для анализа данных, передаваемых посредством многоуровневого туннеля, а также эффективной работы с восстановленными потоками.

3.1.2 Bro Network Security Monitor

Инструмент Bro Network Security Monitor [5] (далее Bro) позволяет анализировать трафик в реальном времени и выполнять определенные действия в случае обнаружения в нем заданных сигнатур. Разработчик – Vern Paxson, лицензия – BSD license. Анализатор Bro поддерживает ОС на основе Unix (Linux, FreeBSD, Mac OS X). Является однопоточным приложением.

С точки зрения архитектуры можно выделить два компонента:

1. генератор событий (осуществляет анализ трафика, получаемого с помощью libpcap, и генерирует события);
2. обработчики событий (собственный язык создания скриптов позволяет выдавать предупреждения, вести логирование, а также запускать сторонние приложения в случае наступления определенных событий; таким образом, может быть сформулирована некоторая политика безопасности).

Архитектура системы Bro схематично представлена на рис. 3. Инструмент поддерживает анализ трафика в режиме реального времени, а также отложенный анализ сетевых трасс (в формате pcap).

Вместе с системой поставляется большое количество скриптов (полный список приведен здесь [6]). Язык скриптов поддерживает фиксированный набор типов и атрибутов ([7]), среди которых отметим тип *event* (событие). Каждому событию (полный список приведен здесь [8]) можно поставить в соответствие обработчик (один или несколько) – в случае наступления события будет вызван соответствующий обработчик. Если к одному и тому же событию «привязано» несколько обработчиков, необходимо задать атрибут приоритетности, определяющий порядок вызова этих обработчиков. События генерируются ядром системы Bro. Добавление нового типа события подразумевает изменения ядра.

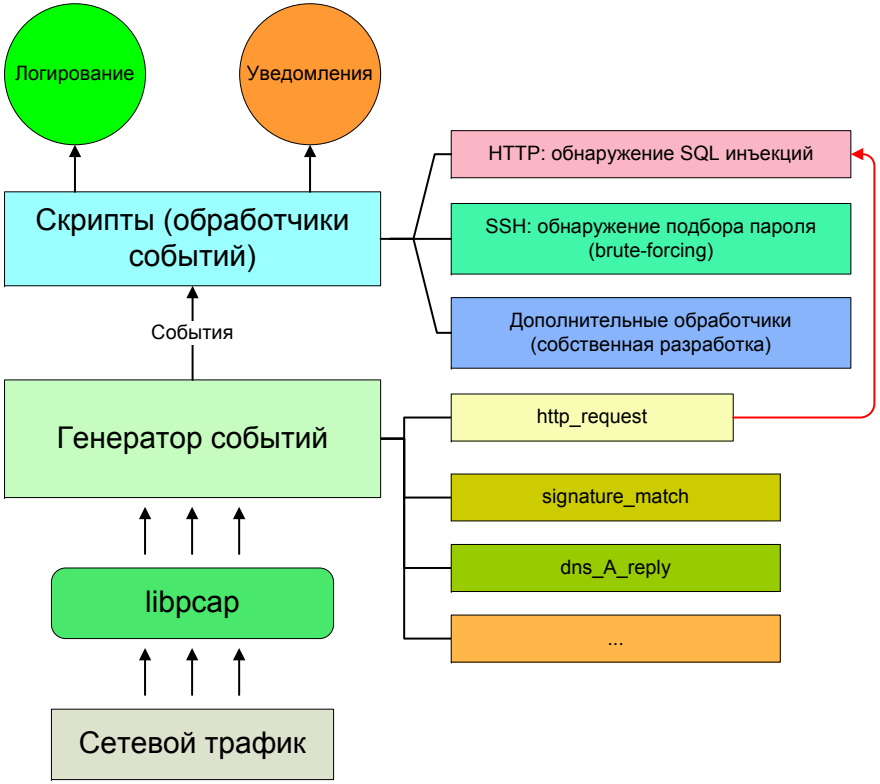

Рис. 3 – Архитектура Bro.

Одним из наиболее важных событий с точки зрения анализа является событие signature_math. Оно генерируется в случае обнаружения некоторого паттерна в разбираемом сетевом пакете.

```
signature
myFirstSignature
{
    ip-proto == tcp
    dst-port == 80
    payload /.*root/
    event "Found root!"
}
```

Рис. 4 – Пример сигнатуры в Bro.

Посредством сигнатуры (рис. 4) осуществляется поиск регулярного выражения (/.*root/)во всех пакетах протокола TCP, отправленных на порт 80. Каждая сигнатура обладает набором атрибутов. Существует два типа атрибутов: conditions (условия) и actions (действия). Условия представляют собой критерии совпадения, а действия определяют операции, производимые в случае совпадения. Условия могут применяться как к заголовкам пакетов, так и к их полезной нагрузке. Существует возможность установления зависимостей между сигнатурами, которая фактически распространяет механизм сигнатурного поиска на последовательности пакетов. В системе определены два действия:

- генерация события signature_match;
- запуск разбора протокола следующего уровня.

Таким образом, разборщики связываются посредством механизма событий. Кроме того, в системе Bro реализована система динамического распознавания протоколов на основе сигнатурного поиска. Поддерживаются протоколы ftp, http, bittorrent, ssh, ssl, pop3, smtp, ayiya. В качестве примера можно привести сигнатуру для протокола HTTP (рис. 5).

```
signature dpd_http_client
{
    ip-proto == tcp
    payload /^[[:space:]]*(GET|HEAD|POST)[[:space:]]*/
    tcp-state originator
}
```

Рис. 5 – Сигнатура для протокола HTTP в Bro.

Управление системой осуществляется через консоль, графический интерфейс отсутствует.

Инструмент поддерживает анализ туннелей. В то же время, автоматическое восстановление стека протоколов туннеля отсутствует. Информация о восстановленных туннелях хранится в текстовом лог файле, навигация отсутствует. Количество поддерживаемых туннельных протоколов крайне мало́.

3.1.3 Snort

Анализ трафика в системе Snort [9] основан на механизме сигнатурного поиска. Разработчик – Sourcefire, лицензия – GNU General Public License. Поддерживаются следующие ОС: Unix (Apple Mac OS X, FreeBSD), Linux (Ubuntu, Fedora, Debian), Microsoft Windows. Snort может работать в двух режимах:

- анализ трафика в реальном времени:
    - Sniffer (вывод в консоль содержимого сетевых пакетов);
    - Logger (запись сетевых трасс);
    - NIDS (Network Intrusion Detection System, анализ трафика);
- отложенный анализ сохраненных ранее сетевых трасс (как правило, используется для тестирования и отладки Snort).

Как уже было отмечено, анализ осуществляется путем поиска указанных в конфигурационном файле паттернов в сетевых пакетах (сигнатурный поиск). В случае успеха выполняются определенные аналитиком действия.

Архитектура Snort представлена на рис. 6.

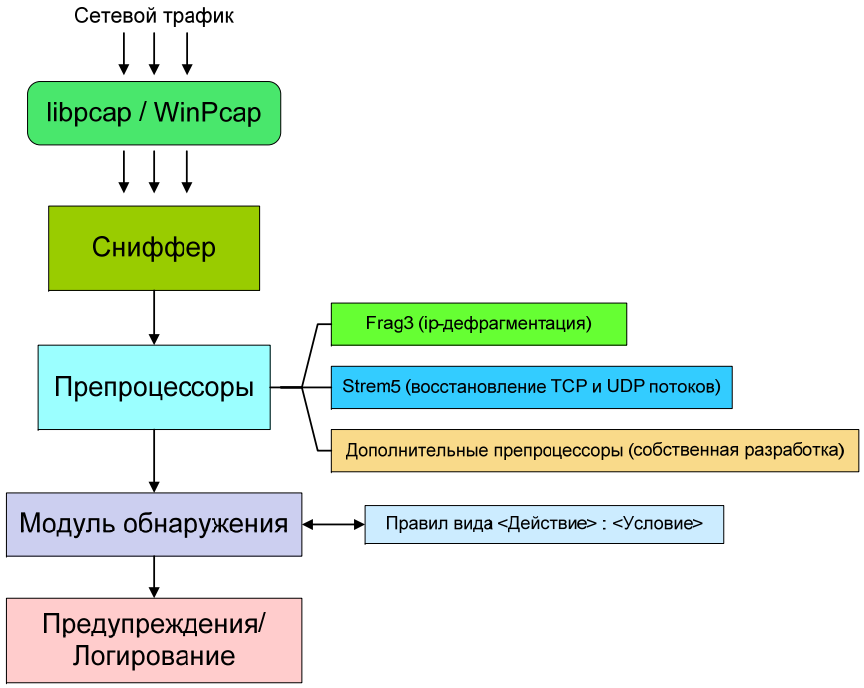

Рис. 6 – Архитектура Snort.

Можно выделить два главных компонента – набор препроцессоров и модуль обнаружения. Фактически препроцессоры занимаются разбором сетевых протоколов (анализ пакетов, восстановление потоков). Snort предоставляет порядка двадцати базовых препроцессоров (в том числе Frag3 – модуль ip-дефрагментации, Stream5 – модуль восстановления TCP и UDP потоков). Аналитик может создавать дополнительные препроцессоры и добавлять их в систему. Каждый препроцессор обладает собственным набором параметров (параметры используемых препроцессоров должны быть описаны в конфигурационном файле snort.conf). Базовые препроцессоры подробно описаны на сайте разработчика [10]. Задача модуля обнаружения состоит в том, чтобы выполнять определенные действия в случае обнаружения заданных аналитиком паттернов в разобранных пакетах и восстановленных потоках. Основу модуля составляют правила. Каждое правило состоит из заголовка и набора опций. Заголовок содержит действие, протокол, к которому применяется данное правило, ip-адрес и порт источника, ip-адрес и порт приемника. Одна из опций представляет собой паттерн, в случае обнаружения которого будет предпринято действие, указанное в заголовке.

```
alert tcp any any -> 192.168.1.0/24 111 \
    (content:"|00 01 86 a5|"; msg:"mountd access";)
```

Рис. 7 – Правило для Snort.

Правило (рис. 7) сгенерирует событие alert и выведет в консоль сообщение «mountd access» в случае, если в пакете протокола TCP, поступившем на порт 111 и ip-адрес 192.168.1.0/24, будет обнаружена последовательность байт «00 01 86 a5». Остановимся подробнее на структуре заголовка и приведем список возможных действий:

- Alert – сгенерировать предупреждение, после чего записать содержимое сетевого пакета в лог файл;
- Log – записать содержимое сетевого пакета в лог файл;
- Pass – пропустить сетевой пакет;
- Activate – сгенерировать предупреждение, после чего активировать другое правило (позволяет распространить сигнатурный поиск на последовательность пакетов);
- Dynamic – правило не выполняется до тех пор, пока не будет активировано каким-либо другим правилом.

В качестве ip-адреса (пары ip-адресов) можно указать ключевое слово *any* – правило будет применяться ко всем пакетам соответствующего протокола. Snort не поддерживает доменные имена. Можно использовать оператор отрицания, а также задавать список ip-адресов. Сказанное справедливо и для портов. Кроме того, можно задавать диапазон портов. Направление передачи сетевых пакетов задается посредством операторов <>, –>, <–. Подробное руководство по созданию правил приводится на сайте разработчика [11].

В Snort реализована поддержка туннелирования (GRE, IP in Ip, PPTP). В то же время, инструмент не может декодировать более одного уровня инкапсуляции (туннелирование вида [Eth] → [IPv4] → [GRE] → [IPv4] → [GRE] → [IPv4] → [TCP] → [Payload] разбираться не будет). Также отметим, что в лог-файл попадает лишь инкапсулированная часть пакетов (при разборе пакетов [Eth] → [IP1] → [GRE] → [IP2] → [TCP] → [Payload] в лог будет записано [Eth] → [IP2] → [TCP] → [Payload]; это создает дополнительные сложности для понимания результатов анализа). По окончании работы инструмента (как в режиме чтения ранее сохраненного pcap-файла, так и в online-режиме) выводится статистика проведенного анализа (отдельно для каждого препроцессора).

3.2 Коммерческие инструменты

3.2.1 Colasoft Capsa

Система анализа трафика Colasoft Capsa главным образом предназначена для выявления проблем в работе сети, а также их локализации. Разработчик – Colasoft, лицензия – Freeware, Shareware. Поддерживаются следующие версии ОС Windows: XP/2003/2008/Vista/7/8. С помощью системы можно анализировать как проводные сети Ethernet, так и трафик беспроводных сетей. Сайт разработчика [12] предоставляет демо-версию Colasoft Capsa (данная версия является ограниченной: нельзя одновременно анализировать трафик нескольких сетевых интерфейсов, недоступен модуль обнаружения сетевых атак, наложено ограничение на количество машин (не более 50), трафик которых анализируется). Тем не менее, для ознакомления ее вполне достаточно.

Система Colasoft Capsa состоит из нескольких модулей, схематично представленных на рис. 8. Модуль сборки трафика перехватывает сетевые пакеты и передает их в анализатор для проведения разбора в реальном времени (анализ заголовков сетевых пакетов, сборка статистики).

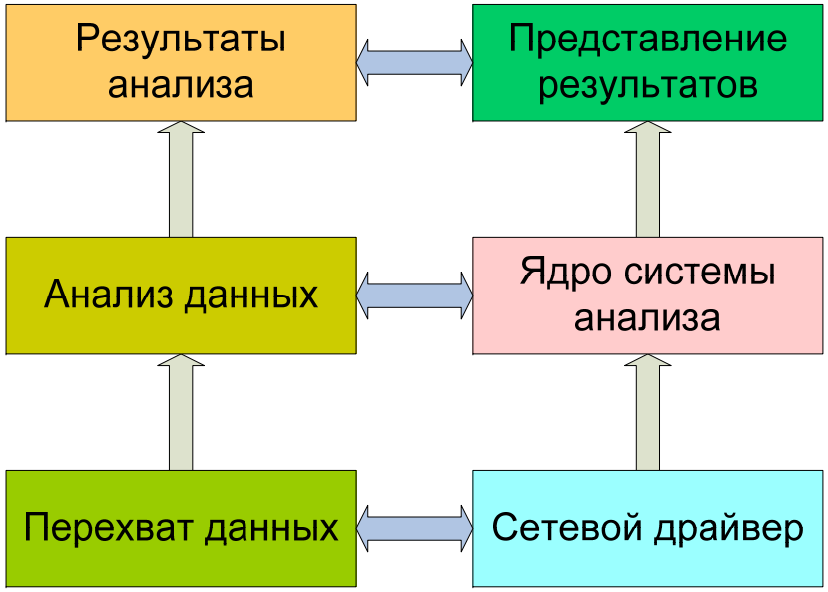

Рис. 8 – Модули Colasoft Capsa.

Рассмотрим процедуру анализа трафика более подробно (рис. 9). Если перехваченный сетевой пакет удовлетворяет критериям фильтрации, то он передается в модуль проведения первичного анализа. Он включает разбор заголовков протоколов канального, сетевого, транспортного уровней, а также сборку статистики. Особо отметим, что фильтрация предшествует проведению анализа – система отбрасывает «неинтересные» пакеты. Тем самым достигается эффективный расход вычислительных ресурсов компьютера. Вторичный анализ включает разбор заголовков протоколов прикладного уровня, работу с потоками, декодирование данных, полученных по сети. Инструмент поддерживает разбор свыше 300 сетевых протоколов (полный список приведен на сайте производителя [13]).

Если говорить об анализе трафика с целью выявления каких-либо особенностей (например, большой объем трафика вследствие сетевой атаки, соответствие определенным сигнатурам), то система Colasoft Capsa предоставляет гибкую систему настройки уведомлений по большому количеству различных критериев. Уведомления могут быть установлены на разных уровнях:

- на уровне отдельного пакета;
- на уровне протокола;
- на уровне отдельного соединения.

На сайте разработчика упоминается отдельный модуль по обнаружению сетевых атак, однако в распространяемой демо-версии этот модуль недоступен.

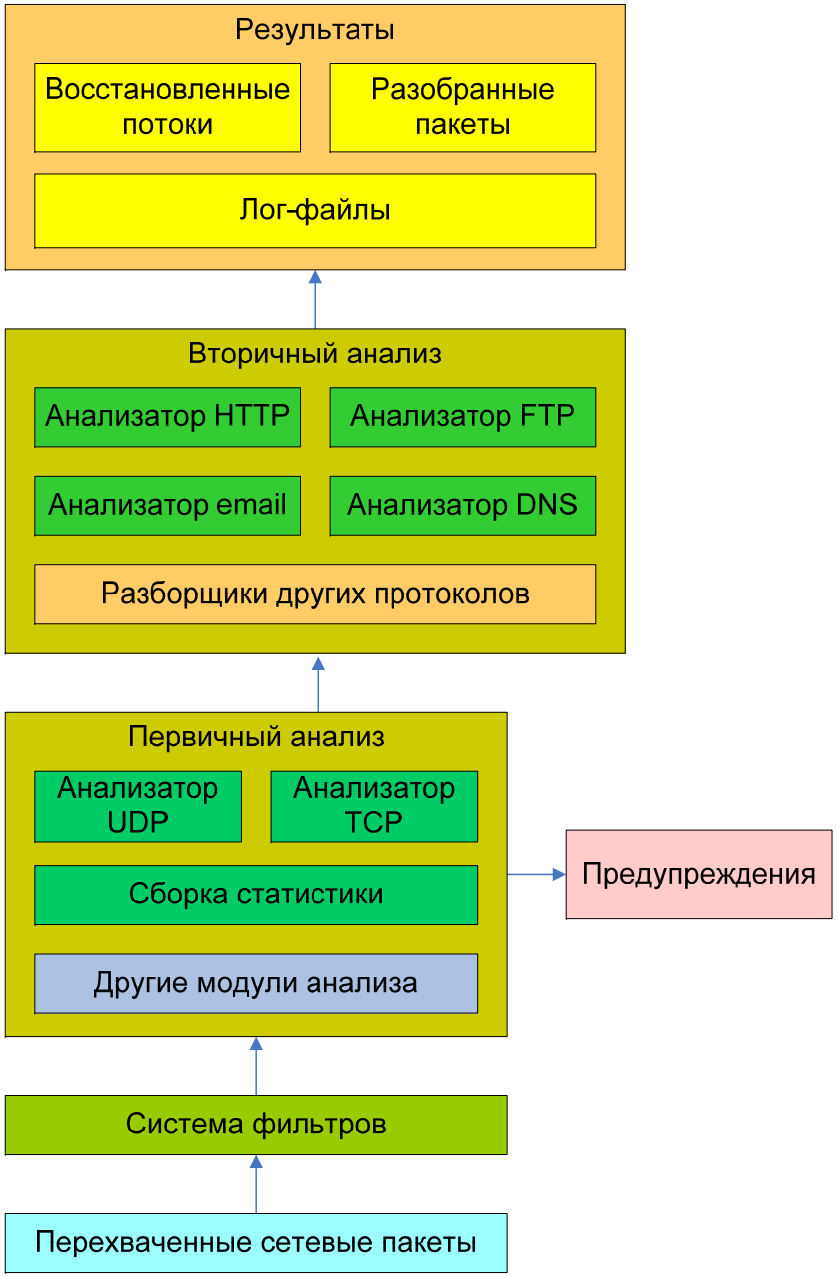

Рис. 9 – Анализ трафика в системе Colasoft Capsa.

Если говорить о технологических аспектах реализации системы Colasoft Capsa, то в первую очередь отметим, что анализ трафика выполняется в многопоточном режиме. Кроме того, используется технология прямого доступа к памяти (DMA), позволяющая существенно ускорить передачу данных.

Система поддерживает анализ туннелей, однако предоставляемые средства навигации не позволяют эффективно отслеживать вложенность сетевых взаимодействий.

3.2.2 ClearSight Analyzer

Система ClearSight Analyzer [14] является частью программно-аппаратного комплекса Network Time Machine (рис. 10). Разработчик – Fluke Networks. Специализированная FPGA сетевая карта позволяет захватывать 100% трафика, передаваемого со скоростью 10-20 Gbps, без задержек. Кроме того, она предоставляет возможность фильтрации сетевых пакетов. Захваченный сетевой картой трафик сразу (в обход операционной системы) записывается на жесткий диск. Одновременно с записью, параллельно, осуществляется подробная индексация трафика по множеству характеристик (посредством модуля Atlas), что в дальнейшем позволяет быстро локализовать интересующий сетевой пакет (или поток) для конкретного приложения. Компонент ClearSight осуществляет сборку пакетов в сессии и анализирует их.

Благодаря аппаратной части (FPGA сетевая карта, индексация трафика) достигаются хорошие результаты при анализе высокоскоростных каналов связи. Если анализируемый сетевой канал имеет пропускную способность ~100 Mbps, то наличие аппаратной поддержки не обязательно. Обычные сетевые карты справляются с такой нагрузкой – трафик записывается на жесткий диск, после чего осуществляется его анализ. При этом индексация не производится (нет соответствующей аппаратуры), однако она и не требуется, поскольку объемы трафика невелики.

Для протоколов (и их параметров) ClearSight Analyzer позволяет устанавливать так называемые «пороговые» значения. В случае если какое-либо из установленных значений будет достигнуто или превышено, аналитик получит уведомление (также автоматически может быть запущена программа на языке JavaScript). Кроме того, трафик, который «вызвал» превышение порогового значения, будет сохранен отдельно и гарантированно не будет перезаписан в дальнейшем. В качестве примера можно задать максимально допустимое время отклика для HTTP сервера.

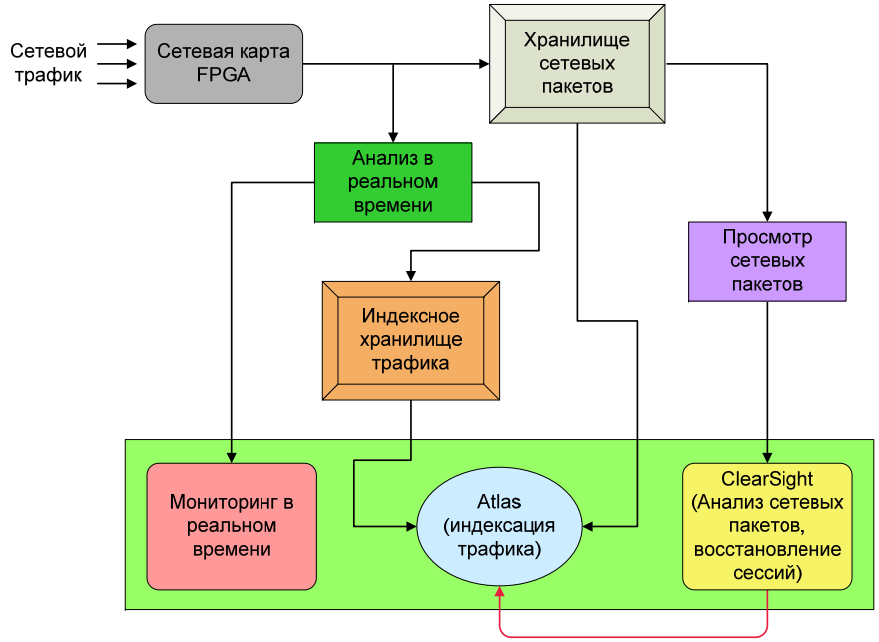

Рис. 10 – Архитектура Network Time Machine.

В интерфейсе пользователь оперирует параметрами сессий и сеансов обмена, а не анализирует отдельные пакеты. Система ClearSight Analyzer реализует принцип Drill Down анализа, когда изучение проблемы происходит от общего к частному. При работе с трафиком ClearSight Analyzer предоставляет полную информацию 7-го уровня, что позволяет быстро выявлять проблемы с точки зрения конечного пользователя (например, ненайденная страница HTTP, занятая линия при звонке VoIP, неправильный пароль при входе на почтовый сервер). Одним из главных преимуществ ClearSight Analyzer является то, что главным образом анализ, в том числе и на 7-ом уровне, происходит в режиме мониторинга, то есть, прежде, чем сетевой трафик будет сохранен. Также отметим удобный и понятный графический интерфейс. Однако, как и в инструменте Colasoft Capsa, при анализе туннелированного трафика крайне затруднительно одновременно проследить за вложенностью соединений и передаваемыми в них данными.

4. Тестирование

4.1 Восстановление TCP сессий

Каждый из рассмотренных инструментов был протестирован на наборе, состоящем из 8 сетевых трасс (каждая трасса содержит пакеты протокола TCP):

- gen-googlemaps.pcap – открытие сайта maps.google.com;
- gen-googleopen.pcap – открытие сайта google.com;
- google-http.pcap – поиск в Google;
- google-https.pcap – поиск в Google с использованием SSL;
- http-google2011.pcap – поиск в Google в 2011 году;
- http-google2012.pcap – поиск в Google в 2012 году;
- http-google.pcap – загрузка сайта Google;
- http-googlesearch.pcap – работа всплывающих подсказок при поиске в Google.

Инструмент Colasoft Capsa 7 Professional Demo не предусматривает восстановление TCP потоков. Вместо этого предлагается анализировать последовательность пакетов (со всеми заголовками) в порядке их прихода, что крайне неудобно.

В табл. 1 приведены TCP-соединения, при восстановлении которых между инструментами возникли различия.

Табл. 1 – Различия при восстановлении TCP сессий.

№	Конечные точки	Размер восстановленной сессии, байт			
		Wireshark v.1.10.5	Snort v.2.9.5.5	Bro v.2.1	ClearSight Analyzer Trial Version
google-http.pcap					
1	192.168.0.105:25162 ↔ 74.125.19.104:80	22925	22925	–	22925
2	192.168.0.105:25161 ↔ 74.125.19.104:80	1216	1216	–	1216
3	192.168.0.105:25160 ↔ 74.125.19.104:80	1897	1897	–	1897
4	192.168.0.105:25168 ↔ 68.142.205.139:80	20125	20125(*)	20125	20125(*)
5	192.168.0.105:25175 ↔ 68.142.205.139:80	6515	6515(*)	6515	6515(*)
6	192.168.0.105:25180 ↔ 68.142.205.139:80	7087	7087(*)	7087	7087(*)
google-https.pcap					
7	192.168.0.105:24044 ↔ 74.125.19.113:443	10480	10527	10480	10527
8	192.168.0.105:24053 ↔ 68.142.205.139:80	6515	6515(*)	6515	6515(*)
9	192.168.0.105:24060 ↔ 68.142.205.139:80	6797	6797(*)	6797	6797(*)
10	192.168.0.105:24089 ↔ 68.142.205.139:80	123832	123832(*)	123832	123832(*)
http-google2011.pcap					
11	24.6.173.220:61960 ↔ 74.125.224.96:443	4429	5143	4429	5143
http-google2012.pcap					
12	24.6.173.220:49922 ↔ 74.125.224.83:80	30489	30704	30489	30704
http-google.pcap					
13	192.168.0.115:37927 ↔ 74.125.19.106:80	31166	31166	–	31166
14	192.168.0.115:37903 ↔ 74.125.19.106:80	15207	15207	–	15207
15	192.168.0.115:37979 ↔ 174.36.36.37:80	0	0	–	0
16	192.168.0.115:37905 ↔ 74.125.19.100:80	773	773	–	773
http-googlesearch.pcap					
17	24.6.173.220:49771 ↔ 74.125.224.105:80	13244	13244	–	13244
18	24.6.173.220:49795 ↔ 74.125.224.83:80	0	0	–	0
19	24.6.173.220:49831 ↔ 63.245.209.93:80	0	0	–	0
20	24.6.173.220:49832 ↔ 96.17.148.90:80	0	0	–	0

Как видно, для некоторых соединений инструмент Bro не восстановил данные. Также, иногда размер сессии, восстановленной посредством Snort и ClearSight Analyzer, превышает размер той же сессии, восстановленной Wireshark и Bro. Содержимое восстановленных сессий также иногда различается (отмечено символом '*'). Анализ показал следующее:

- Bro начинает восстанавливать данные соединения с момента его установления, а Wireshark, Snort и ClearSight – с момента обнаружения первого TCP-сегмента между данными парами адрес-порт. Соединения с номерами 1 – 3, 13 – 20 были установлены до начала записи соответствующей трассы. Поэтому Bro не смог их восстановить;
- Каждое из соединений 4 – 6, 8 – 10 имеет по 2 сегмента, попавших в трассу в неправильном порядке (с точки зрения сборки TCP-потока). При разборе инструменты Bro и Wireshark верно добавили данные этих сегментов в правильном порядке. Snort и ClearSight добавили данные в том порядке, в котором они пришли;
- Соединения 7, 11, 12 имеют сегмент повторной передачи. Данные этих сегментов были отброшены Wireshark и Bro, но добавлены Snort и ClearSight.

Тестирование показало, что инструменты Wireshark и Bro наиболее точно восстанавливают данные TCP-соединений. Однако Bro не восстанавливает (хотя бы частично) соединения, установленные до момента начала перехвата трафика. Snort и ClearSight не производят необходимого переупорядочивания данных, добавляя их в порядке прихода, а также не отбрасывают сегменты, переданные повторно.

4.2 Распознавание протоколов

Каждый из рассмотренных инструментов был протестирован на наборе, состоящем из 9 сетевых трасс:

1. ppp-general.pcap
2. sec-clientdying.pcap
3. sec-nmap-ipscan.pcap
4. smb-joindomain.pcap
5. tcp-pktloss94040.pcap
6. udp-mcaststream-queued2.pcap
7. udp-pentest.pcap
8. vlan-general.pcap
9. voip-extension.pcap

Трассы, попавшие в набор, практически полностью покрывают список протоколов, поддерживаемых инструментом Wireshark.

Инструменты Bro и ClearSight не позволяет получить статистику по протоколам, обнаруженным в трассе. Анализ исходного кода Bro показывает, что инструмент ориентирован на протоколы первых четырех уровней модели OSI. Система ClearSight распознает протоколы всех уровней.

Результаты тестирования распознавания протоколов для инструментов Wireshark, Snort и Colasoft Capsa приведены в табл. 2.

Табл. 2 – Результаты тестирования распознавания протоколов.

Трасса	Инструмент		
	Wireshark v.1.10.5	Snort v.2.9.5.5	Colasoft Capsa 7 Professional Demo
1	eth, ip, gre, ppp, lcp, chap, ccp, ipcp	Eth, IP4, GRE, GRE PPTP	Ethernet II, IP, GRE, PPP, 1st Choise Compression, Link Control Protocol, Challenge Handshake, IP Control Protocol
2	eth, ip, udp, dns, tcp, http, dcerpc, data, smb, irc, nbss, pipe, nntp, mgmt, isystemactivator, tftp	Eth, IP4, UDP, TCP	Ethernet II, IP, UDP, TFTP, DNS, TCP, HTTP, IRC, NNTP, CIFS, FTP
3	eth, ip, igmp, icmpv6, vines_frp, ncs	Eth, IP4, ICMP, UDP, TCP, IP6, IP4/IP4, IP6/IP6, GRE	Ethernet II, IP, VRRP, UDP, TLSP, TCP, SPS, SNP, SMP, SDRP, SCTP, RSVP, RDP, PVP, PUP, PTP, PRM, PIPE, PIM, OSPF, NETBLT, NARP, ISO-TP4, IRTP, IPv6, IP, IGRP, IGMP, IDRP, IDPR, ICMPv6, ICMP, HMP, GRE, GGP, FC, ESP, EIGRP, EGP, DSR, DDP, DCCP, CRUDP, CRTP, CFTP, AH
4	eth, ip, udp, dns, tcp, dcerpc, icmp, arp, smb, cldap, kerberos, nbss, pipe, lsarpc, samr, rpc_netlogon, ldap	Eth, IP4, ICMP, UDP, TCP, ARP	Ethernet II, IP, UDP, LDAP, Kerberos, DNS, TCP, LDAP, CIFS, ICMP, ARP

5	eth, ip, udp, dns, tcp, http, media, image-jfif, png, image-gif, uasip, bootp, dcerpc, data, xml, snmp, ndns, arp, ssl, nbdgm, smb, mailsoft, browser	Eth, IP4, UDP, TCP, ARP	Ethernet II, IP, TCP, HTTP, HTTPS, UDP, SSDP, SNMP, BOOTP, DHCP, DNS, NetBIOS, ARP
6	eth, ip, udp, mp2t, text, mpeg-pes, mpeg_pat, mpeg_pmt, dvb_nit, dvb_sdt	Eth, IP4, UDP	Ethernet II, IP, UDP
7	eth, ip, udp, dns, bootp, icmp, data, snmp, nbns, kerberos, ntp, tftp, rpc, portmap, l2tp, sip, classicstun, isakmp, daytime, radius, rx, echo, rip, quake3, xdmcp, krb4, kpasswd	Eth, IP4, ICMP, UDP	Ethernet II, IP, UDP, SNMP, DNS, NetBIOS, Name Service, L2TP, RIP, RIPv1, RIPv2, BOOTP, DHCP, SNMP Trap, NTP, SIP, Daytime, Kerberos, RADIUS, TFTP, SSH, Qotd, Echo, RPC, FTP, ICMP
8	eth, ip, tcp, vlan, x11	Eth, VLAN, IP4, TCP	Ethernet II, VLAN, IP, TCP, X-Window
9	eth, ip, udp, sip, rtp, rtcp, rtpevent	Eth, IP4, UDP	Ethernet II, IP, UDP, RTP, SIP, RTCP

В таблице приведены сокращения, используемые инструментами Wireshark, Snort и Colasoft Capsa. Бóльшая часть интерпретируется однозначно, в случае затруднений следует обратиться к документации соответствующего инструмента. Легко видеть, что при распознавании Snort «останавливается» на протоколах транспортного уровня и не идет «вверх». Отметим, что при тестировании использовалась версия Snort со стандартным набором правил. Инструменты Wireshark и Colasoft Capsa одинаково хорошо распознают протоколы всех уровней модели OSI.

5. Заключение

В статье были рассмотрены популярные в настоящий момент инструменты анализа сетевого трафика: свободно распространяемые (Wireshark, Bro Network Security Monitor, Snort) и коммерческие (Colasoft Capsa, ClearSight Analyzer). Для каждого инструмента приведено описание архитектуры, а также основные достоинства и недостатки с точки зрения функциональности и удобства использования.

Если говорить о свободно распространяемых инструментах, то лидером с точки зрения распространенности является Wireshark. Wireshark поддерживает разбор и распознавание более 1000 сетевых протоколов. Bro, Snort, и даже Colasoft Capsa не могут похвастаться таким количеством поддерживаемых протоколов. Кроме того, Wireshark предоставляет пользователю графический интерфейс, тогда как Bro и Snort являются консольными приложениями. В Wireshark отсутствует возможность выполнения некоторого действия в случае обнаружения каких-либо паттернов в трафике. Snort и Bro предоставляют функциональность для данного сценария работы.

Идеологически Wireshark является средством просмотра и сбора статистики, но не анализа. Snort позиционирует себя как систему обнаружения вторжений – отсюда архитектурное решение в пользу модели событий. Bro – некоторый компромисс между Wireshark и Snort. Так или иначе, ни один из свободно распространяемых инструментов не позволяет эффективно анализировать сессии – каждый из инструментов способен лишь восстанавливать потоки протокола TCP, но не устанавливать связи, существующие между потоками в рамках работы приложений. Данное ограничение возникает на уровне архитектуры и не может быть разрешено.

Что касается коммерческих инструментов, то предпочтение отдается системе ClearSight Analyzer (в составе программно-аппаратного комплекса Network Time Machine): поддержка большого количества сетевых протоколов, поддержка высокоскоростных сетей (~10 Gbps), гибкая система уведомлений с возможностью запуска собственных Java-скриптов, удобный графический интерфейс. Одновременно ClearSight Analyzer является наиболее дорогостоящим инструментом – цена составляет около $7000 (цена комплекса Network Time Machine колеблется от $40000 до $120000 в зависимости от комплектации). Однако не стоит забывать, что рассмотренные коммерческие анализаторы не пригодны для восстановления потоков протокола TCP.

В заключение отметим, что ни один из рассмотренных инструментов не предоставляет удобного интерфейса для работы с результатами анализа туннелированного трафика.

6. Список литературы

[1]. Colasoft Packet Player. http://www.colasoft.com/packet_player/, дата обращения 13.04.2014
[2]. Wireshark Trace Files. http://www.wiresharkbook.com/studyguide_supplements/9781893939943_traces.zip, дата обращения 13.04.2014
[3]. Wireshark. http://www.wireshark.org/, дата обращения 13.04.2014
[4]. Wireshark Display Filter Reference. http://www.wireshark.org/docs/dfref/, дата обращения 13.04.2014
[5]. The Bro Network Security Monitor. http://www.bro.org/, дата обращения 13.04.2014
[6]. The Bro Network Security Monitor: Scripts. http://www.bro.org/sphinx/scripts/index.html, дата обращения 13.04.2014
[7]. The Bro Network Security Monitor: Types and Attributes. http://www.bro.org/sphinx/scripts/builtins.html, дата обращения 13.04.2014
[8]. The Bro Network Security Monitor: Protocol Independent Events. http://www.bro.org/sphinx/scripts/base/bif/event.bif.html, дата обращения 13.04.2014
[9]. Snort. http://www.snort.org/, дата обращения 13.04.2014
[10]. Snort: Preprocessors. http://manual.snort.org/node59.html, дата обращения 13.04.2014
[11]. Writing Snort Rules. http://manual.snort.org/node291.html, дата обращения 13.04.2014
[12]. Colasoft Capsa. http://www.colasoft.com/capsa/, дата обращения 13.04.2014
[13]. Colasoft Capsa: Supported Protocols. http://kb.colasoft.com/technical-problems-and-errors/97-supported-protocols, дата обращения 13.04.2014
[14]. ClearSight Analyzer. http://www.fluke-networks.ru/clearsight-analyzer-analysis-of-the-protocols-in-real-time/clearsight-analyzer-analysis-of-the-protocols-in-real-time-review, дата обращения 13.04.2014

The modern network traffic analyzers overview

Y. V. Markin, A. S. Sanarov
{ustas, intkecsk }@ispras.ru

Abstract. This paper describes the modern network traffic analyzers. Each of the instruments examined is capable of analyzing both live traffic and previously saved network traces. Network trace is a sequence of packets going over network interface and some additional information depends on the trace format. Most of modern network interface controllers support promiscuous mode. This mode causes the controller to pass all traffic it receives to the CPU rather than passing only the frames that the controller is intended to receive. So this mode is normally used for packet sniffing.

Architecture of each analyzer is given. Highs and lows are also underlined from the points of functionality and usability. Network protocols recognition and stream assembling testing are made and the results are presented in comparative tables.

Keywords: sniffer; network protocols recognition; session assembling; intrusion detection system

References

[1]. Colasoft Packet Player. http://www.colasoft.com/packet_player/
[2]. Wireshark Trace Files. http://www.wiresharkbook.com/studyguide_supplements/9781893939943_traces.zip
[3]. Wireshark. http://www.wireshark.org/
[4]. Wireshark Display Filter Reference. http://www.wireshark.org/docs/dfref/
[5]. The Bro Network Security Monitor. http://www.bro.org/
[6]. The Bro Network Security Monitor: Scripts. http://www.bro.org/sphinx/scripts/index.html
[7]. The Bro Network Security Monitor: Types and Attributes. http://www.bro.org/sphinx/scripts/builtins.html
[8]. The Bro Network Security Monitor: Protocol Independent Events. http://www.bro.org/sphinx/scripts/base/bif/event.bif.html
[9]. Snort. http://www.snort.org/
[10]. Snort: Preprocessors. http://manual.snort.org/node59.html
[11]. Writing Snort Rules. http://manual.snort.org/node291.html
[12]. Colasoft Capsa. http://www.colasoft.com/capsa/
[13]. Colasoft Capsa: Supported Protocols. http://kb.colasoft.com/technical-problems-and-errors/97-supported-protocols
[14]. ClearSight Analyzer. http://www.fluke-networks.ru/clearsight-analyzer-analysis-of-the-protocols-in-real-time/clearsight-analyzer-analysis-of-the-protocols-in-real-time-review