И. С. ГЛОТОВ, студент гр. КЗИ-108;
С. С. ВЛАСЕНКО, студент гр. ИБм-111;
А. В. ТИШИН, студент гр. ИБм-111;
А. С. БУШЕВ, ассистент каф. ИЗИ;
Ю. М. МОНАХОВ, к.т.н., доцент каф. ИЗИ.

РАЗРАБОТКА ЭЛЕМЕНТОВ АРХИТЕКТУРЫ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ

Рассмотрена общая схема архитектуры проекта на основе различных уровней абстракции, используя объектно-ориентированный подход. Определено взаимодействие серверного уровня приложения на центральном сервере: механизм идентификации сенсоров на основе сессий, способ приема и обработки трафика, выделены классы-элементы для работы с оператором и решателями. Проведен обзор других уровней и их компонентов. Разработана схема потока трафика.

Ключевые слова: РАСПРЕДЕЛЕННАЯ СИСТЕМА ОБНАРУЖЕНИЯ ВТОРЖЕНИЙ, АРХИТЕКТУРА ВЗАИМОДЕЙСТВИЯ МЕЖДУ ЧАСТЯМИ ПРИЛОЖЕНИЯ, СЕРВЕРНЫЙ УРОВЕНЬ ПРИЛОЖЕНИЯ, ДЕКОМПОЗИЦИЯ КОМПОНЕНТОВ РСОВ, СХЕМА ПОТОКА ТРАФИКА.

6 рис., 0 табл., 14 источников.

Объектом разработки являются элементы архитектуры распределенной системы обнаружения вторжений.

Цель работы – разработать принципиальную схему взаимодействия компонентов распределенной системы обнаружения вторжений, спроектировать порядок и способ обмена сенсорной подсистемы с серверным уровнем.

Система обнаружения вторжений, представляет собой группу следующих компонентов (см. рис. 1):

- Головное приложение на центральном сервере («Head») - совокупность элементов, взаимодействующих друг с другом и условно различающихся по уровню обработки данных;
- Сенсоры - аппаратно-обособленные объекты, осуществляющие сбор сетевого трафика из сети;
- База данных («Хранилище») - объект, хранящий в себе сериализованные структуры данных захваченного трафика;
- Решатели - программы, позволяющие различными методами определить энтропию наличия атаки путем анализа предоставленных наборов данных.

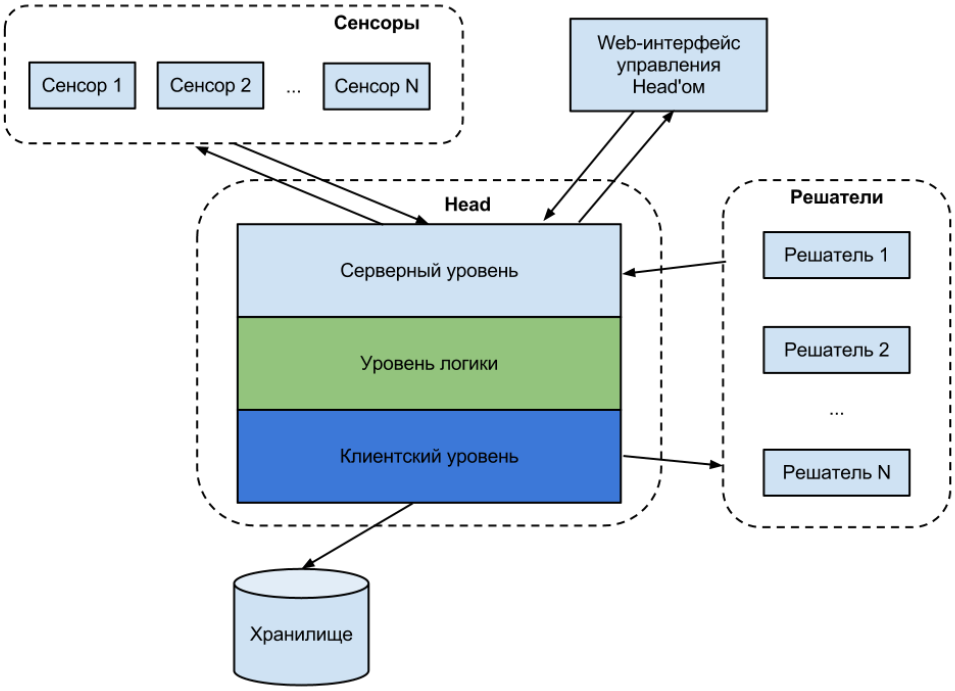

Рис. 1. Общая схема архитектуры проекта.

Головное приложение разделено на несколько условных уровней обработки данных: серверный уровень, уровень логики, клиентский уровень.

Приложение на центральном сервере (Head) состоит из трех уровней:

- Серверный уровень;
- Уровень логики;
- Клиентский уровень.

Декомпозиция компонентов всех уровней представлена на рис. 2.

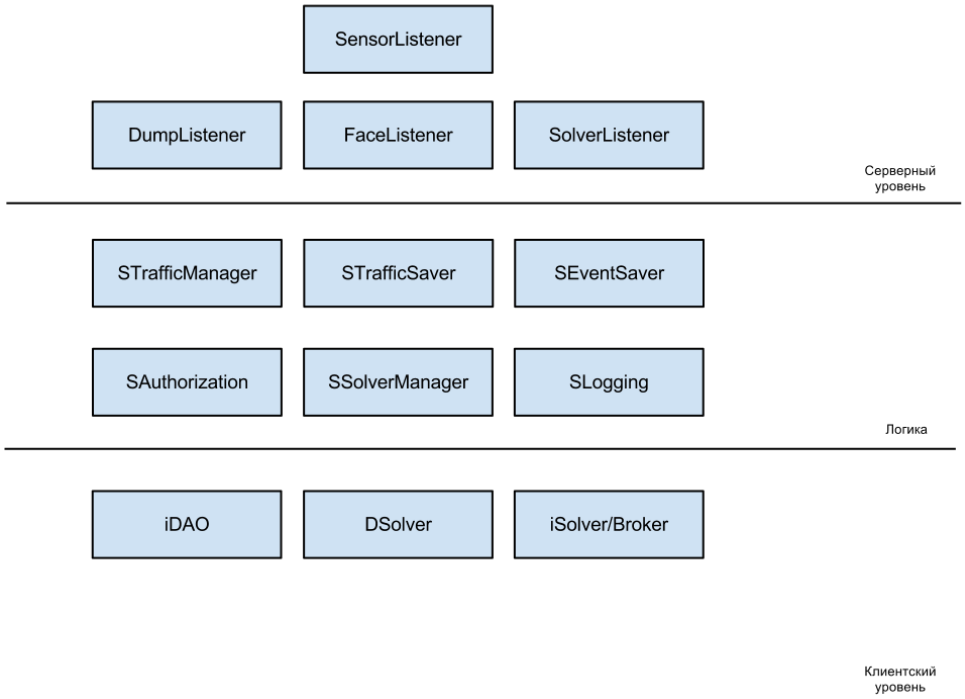

Рис. 2. Общий состав компонентов головного приложения.

Серверный уровень приложения - элемент, выступающий для внешних объектов в виде сервера. Согласно клиент-серверной архитектуре - принимающая сторона, или другими словами совокупность “слушателей” (Listeners), которые не являются инициаторами сетевого соединения.

Данный уровень содержит в себе следующие компоненты:

- SensorListener - предоставляет двухстороннее взаимодействие между сенсорами и приложением на центральном сервере (Head), при котором сенсоры могут получить, либо обновить идентификаторы, необходимые для успешной валидации отправляемого в DumpListener трафика.
- DumpListener - получает от сенсоров сформированные блоки перехваченных данных, распаковывает и обрабатывает их, после чего передает на уровень ниже.
- FaceListener - предоставляет различную информацию для оператора: о сенсорах, возможных угрозах, решателях, статистических данных и др.
- SolverListener - предоставляет двухстороннее взаимодействие между решателями и приложением на центральном сервере; реализует

интерфейс настройки формата передаваемого трафика от сенсоров решателям; получает результаты работы решателей.

Концептуально взаимодействие сенсоров и подсистем можно разделить на следующие этапы (см. рис. 3):

Этап 1: установление защищенного соединения с компонентом SensorListener при помощи асимметричной системы шифрования.

Этап 2: запрос SensorListener на получение или обновление идентификатора сессии и времени жизни сессии у объекта SAuthorization.

Этап 3: отправка сенсору по защищенному соединению идентификаторов.

Этап 4: пока время жизни сессии не истекло – отправка трафика и контрольных сумм идентификаторов сенсором в объект DumpListener.

Этап 5: отделение контрольных сумм идентификаторов и их проверка на валидность через методы объекта SAuthorization.

Этап 6: при удачной валидации передача трафика к дальнейшей обработке в STrafficManager.

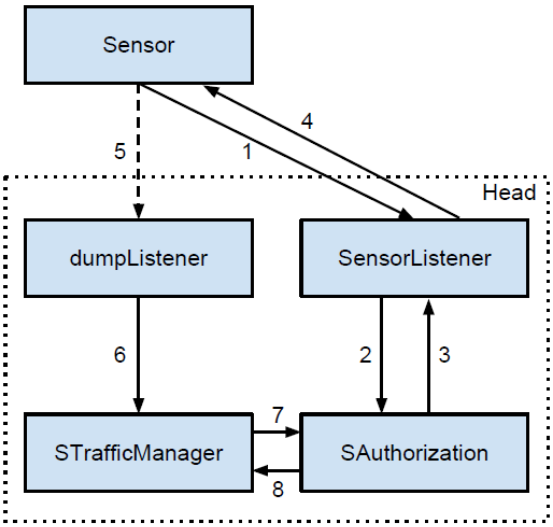

Рис. 3. Схема взаимодействия сенсоров и приложения на центральном сервере.

Для наглядности взаимодействия были построены UML-диаграммы последовательности (см. рис. 4), классов (см. рис. 5).

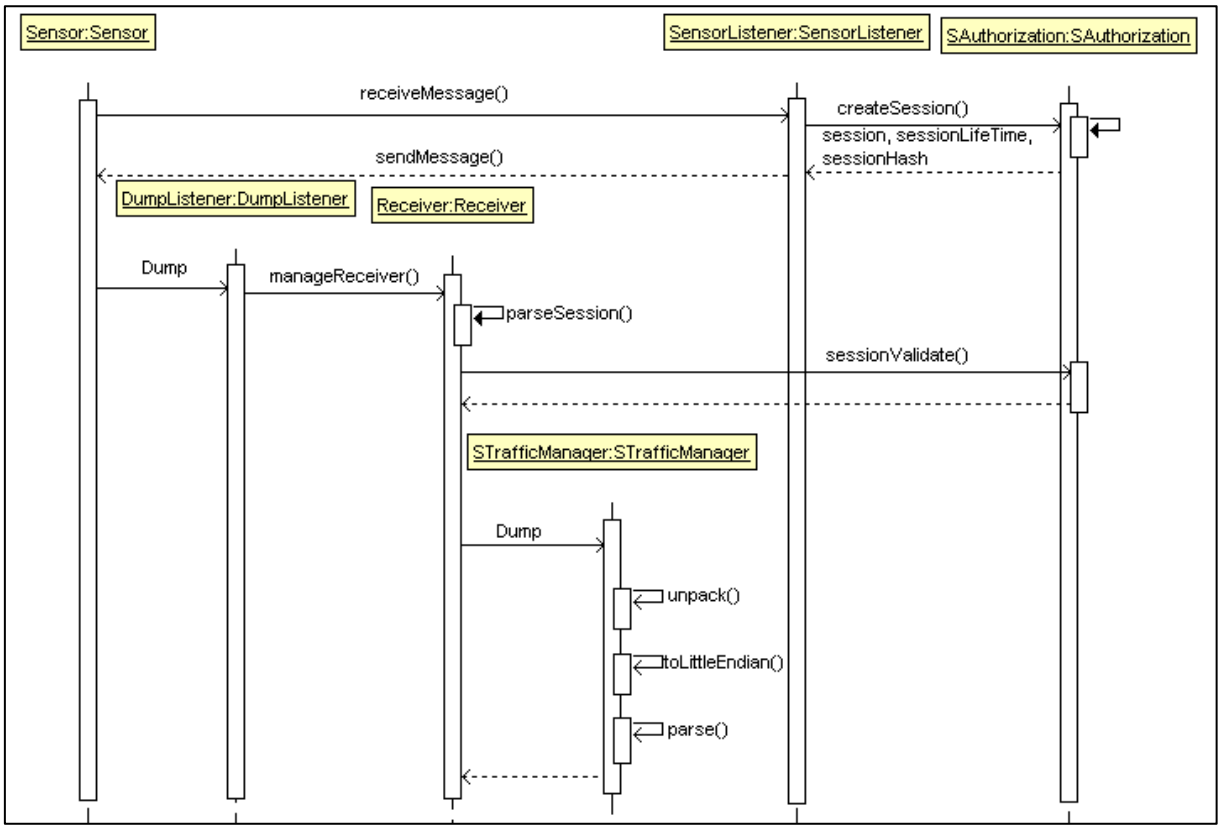

Рис. 4. Диаграмма последовательности.

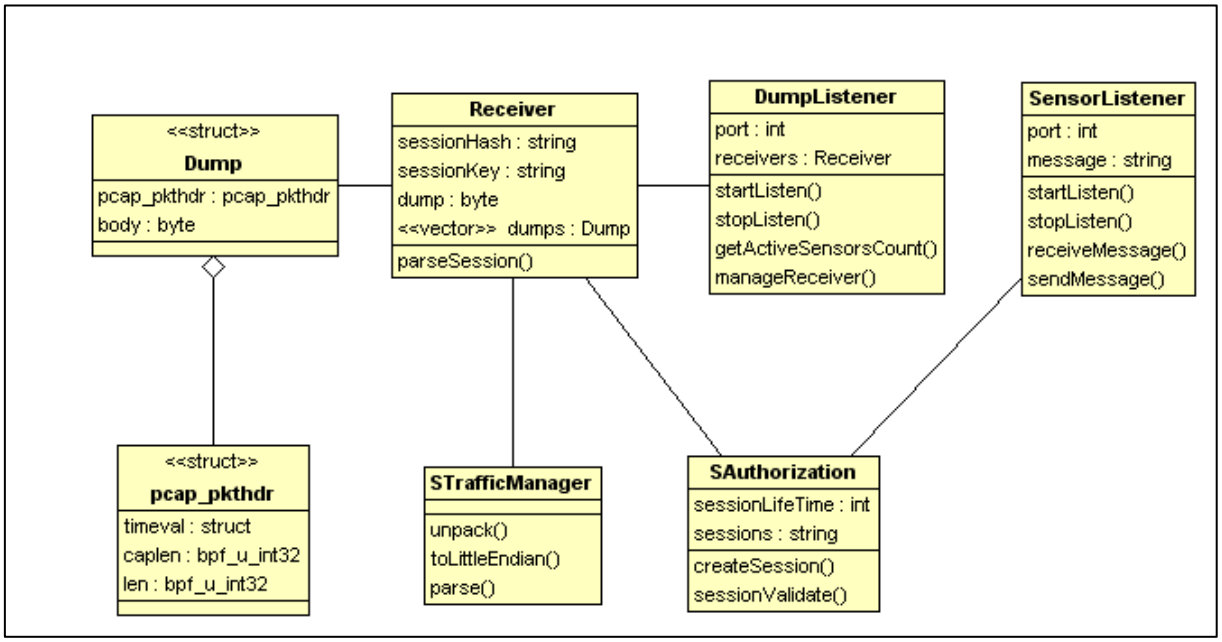

Рис. 5. Диаграмма классов.

Организацию транспортировки трафика между компонентами предполагается осуществить при помощи специализированной библиотеки обмена сообщениями. Преимущества использования подобных библиотек: большое количество биндингов для многих языков программирования;

гибкость и простота проектирования, достигающаяся через унификацию процесса обмена сетевыми сообщениями. Учитывая еще такой параметр, как открытость (LGPL) [1], производительность за счет безброкерной схемы передачи [2],[3],[4], была выбрана система ZeroMQ.

Разработанная схема потока трафика указана на рис. 6.

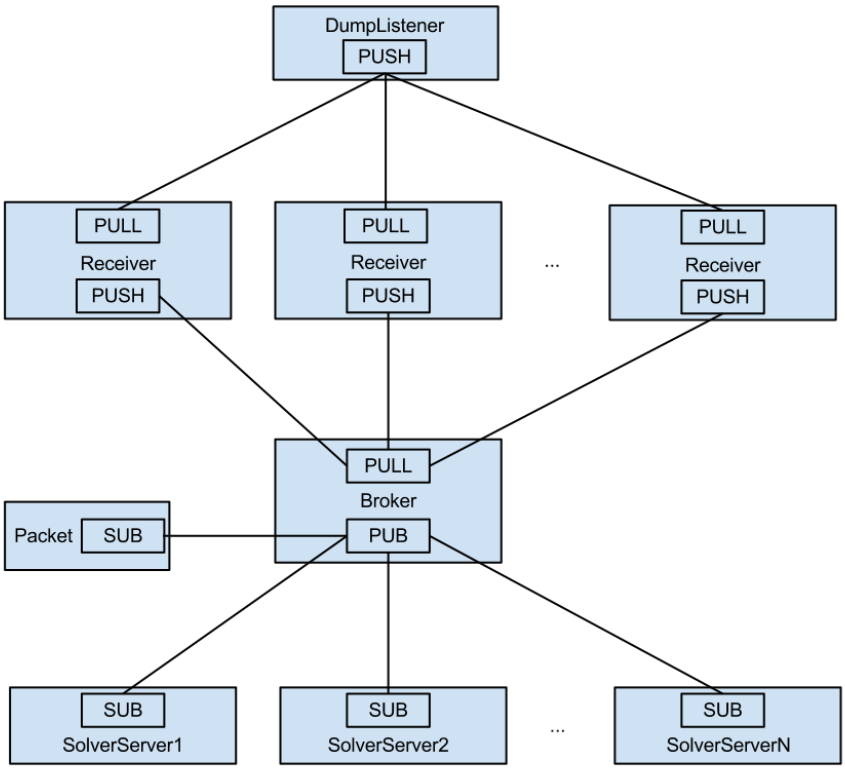

Рис. 6. Поток трафика с использованием ZeroMQ.

Таким образом, данные будут распределяться из DumpListener по инициализированным потокам Receivers. Трафик будет обрабатываться в методах объекта STrafficManager и выходить на PULL объекта Broker, который является коллектором трафика и занимается его публикацией на решатели (SolverServer1, SolverServer2, …, SolverServerN) и на объект интерфейса iDAO - Packet, сохраняющего трафик в хранилище.

Текущие задачи рабочей группы заключаются в описании логического и клиентского уровней, точном определении набора обрабатываемых и передаваемых данных на периметральных участках системы и составлении на этом основании UML-диаграмм; разработке ядра системы, инициализирующего все компоненты; определении порядка взаимодействия

головного приложения с веб-интерфейсом; проведении аналитической работы по имеющимся свободным реализациям баз данных на предмет производительности, отказоустойчивости и масштабируемости в условиях, описанных в настоящем докладе; разработке схемы таблиц хранения событий и трафика; дополнение общей схемы потока трафика между всеми компонентами. Планируется оформить техническое описание всей системы и ее компонентов в частности.

Литература

1. iMatix Corporation, “ØMQ Licensing” http://www.zeromq.org/area:licensing

2. *Mike Hadlow*, “Message Queue Shootout!” http://mikehadlow.blogspot.com/2011/04/message-queue-shootout.html

3. *Njål Karevoll*, “Managing Index Repartitioning”. Norwegian University of Science and Technology, Department of Computer and Information Science. March 2011

4. ØMQ - The Guide. http://zguide.zeromq.org/page:all

5. *Алексей Лукацкий*. Обнаружение атак. BHV - Санкт – Петербург. — 596 стр. —2003 г.

6. Информационная безопасность. Сравнение систем обнаружения вторжений. [Электронный ресурс] — Режим доступа: http://inf-bez.ru/?p=783

7. Компьютерные атаки и технологии их обнаружения. [Электронный ресурс] — Режим доступа: http://protect.htmlweb.ru/attack.htm

8. *Корт С.С., Рудина Е.А.* Архитектура универсального ядра систем мониторинга и защиты от вторжений. // Проблемы информационной безопасности. Компьютерные системы. ГОУ «СПбГПУ» — с. 27-39 — г. Санкт-Петербург, 2008.

9. *Котенко И.В.* Многоагентные технологии для анализа уязвимостей и обнаружения вторжений в компьютерных сетях // Новости искусственного интеллекта. 2004. № 1.

10. *Лапонина О.Р.* Intrusion Detection Systems (IDS). Лекция из курса «Межсетевое экранирование» [Электронный ресурс] — Режим доступа: http://www.citforum.ru/security/internet/firewalls_ids/

11. Сетевая система обнаружения вторжений. Википедия [Электронный ресурс] — Режим доступа: http://ru.wikipedia.org/wiki/сетевая_система_обнаружения_вторжений

12. *Стивен Норткатт, Джуди Новак*. Обнаружение вторжений в сеть. И.: Издательство: «Лори». — 416 .с — 2002 г.

13. *Ali A. Ghorbani, Wei Lu, Mahbod Tavallaee*. Network Intrusion Detection and Prevention: Concepts and Techniques (Advances in Information Security). Springer. — 234 p. — October, 2009.

14. *Chris Fry, Martin Nystrom*. Security Monitoring: Proven Methods for Incident; Publisher: O'Reilly Media; 1 edition, Paperback: 256 pages. – February, 2009.