Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)
Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ
заведующий каф. КИБЭВС
_____А.А. Шелупанов
«_____»_____2017г.

ПРОГРАММНО - АППАРАТНЫЙ КОМПЛЕКС ДЛЯ ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ В ОБЛАСТИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Отчет по групповому проектному обучению
Группа КИБЭВС-1502

Ответственный исполнитель
студент гр. 777
_____И.И. Иванов
«_____»_____2017г.

Научный руководитель
ассистент каф. КИБЭВС
_____А.И. Гуляев
«_____»_____2017г.

Томск 2017

РЕФЕРАТ

Отчет содержит 32 страницы, 9 рисунков, 22 источника, 1 приложение.

CTF, ATTACK-DEFENSE, СОРЕВНОВАНИЯ, KEVA, ЗАЩИТА ИНФОРМАЦИИ, GIT, PYTHON, C, ZABBIX, DPDK, ТЕСТИРОВОЧНЫЙ СТЕНД, СЕТЕВОЙ ТРАФИК, ЗЕРКАЛИРОВАНИЕ ТРАФИКА, СЕТЕВАЯ КАРТА, НЕЙРОННЫЕ СЕТИ, АНАЛИЗ ТРАФИКА.

Объект исследования и разработки — сбор сетевого трафика.

Цель работы — решение проблем мониторинга сетевого трафика при высокой загрузке сети на соревнованях по информационной безопасности, путём создания тестового стенда для зеркалирования трафика с использованием фреймворка DPDK, и установки и настройки системы мониторинга сетевого оборудования Zabbix.

Результаты работы в данном семестре:

– проведено исследование методов для анализа трафика;

– проведено ознакомление со значительным количеством сетевых утилит и программ для захвата трафика;

– изучено устройство и принципы работы с набором библиотек DPDK;

– разработаны требования к тестовому стенду для прозрачного зеркалирования трафика;

– спроектирован и подготовлен тестовый стенд для дальнейшего использования на соревнованиях SibirCTF;

– произведено тестирование стенда, в том числе в условиях, приближенных к реальным;

– был установлен и настроен Zabbix-сервер для мониторинга нагрузки и производительности сетевого оборудования инфраструктуры соревнований.

В качестве инструментария для выполнения данной работы были использованы: система контроля версий Git, система мониторинга компьютерной сети Zabbix, языки программирования C, Python, фреймворк DPDK.

Пояснительная записка выполнена при помощи системы компьютерной вёрстки LaTeX.

THE ABSTRACT

Course work contains 32 pages, 9 pictures, 22 references, 1 attachment.

CTF, ATTACK-DEFENSE, COMPETITIONS, KEVA, INFORMATION SECURITY, GIT, PYTHON, C, ZABBIX, DPDK, DEMOSTAND, NETWORK TRAFFIC, TRAFFIC MIRRORING, NETWORK CARD, NEURAL NETWORKS, TRAFFIC ANALYSIS.

The study object is network traffic grabbing.

The purpose of the work is solving network traffic monitoring problems in high-load networks on information security competitions by creation of the demostand for traffic mirroring via DPDK framework and Zabbix monitoring system installation and configuring.

This term results:

– performed research of traffic analysis methods;

– performed research of network traffic grabbing utilities and programs;

– performed research of DPDK framework methods;

– developed demostand requirements for hidden traffic mirroring;

– designed and prepared demostand for using on SibirCTF;

– performed demostand testing (including environment closed to real);

– installed and configured Zabbix server for payload and perfomance monitoring of network devices.

This work is made with: version control system Git, Zabbix network monitor, C and Python programming language, DPDK framework.

Course work is made with LaTeX.

Список исполнителей

Иванов И.И. – ответственный исполнитель, программист.

Задачи:

– изучение фреймворка DPDK применительно к теме проекта;

– разработка программных и аппаратных требований, предъявляемых к тестовому стенду;

– разработка программной части тестового стенда;

– повышение квалификации в области информационной безопасности.

Смирнов И.И. – системный администратор.

Задачи:

– изучение предметной области и основ работы с Zabbix;

– установка и настройка Zabbix, а также его интеграция в сетевую инфраструктуру соревнований;

– повышение квалификации участников за счет участия в соревнованиях, посвященных аспектам информационной безопасности.

Сидоров И.О. – документатор.

Задачи:

– подготовка технического задания;

– подготовка отчета о проделанной работе команды в системе компьютерной верстки LaTeX;

– подготовка презентации для защиты проекта;

– повышение квалификации за счет участия в соревнованиях, посвященных аспектам информационной безопасности.

Косаченко Т.С. – аналитик, тестировщик.

Задачи:

– обзор существующих работ по обнаружению вторжения;

– тестирование работоспособности тестового стенда

– повышение квалификации за счет участия в соревнованиях, посвященных аспектам информационной безопасности.

Сапунов А.В. – программист.

Задачи:

– изучение существующих средств сбора трафика;

– подбор необходимых комплектующих и сборка тестового стенда;

– разработка программной части тестового стенда;

– повышение квалификации за счет участия в соревнованиях, посвященных аспектам информационной безопасности.

Содержание

1 Введение

1.1 Краткая теория

CTF (Capture the flag с англ. «Захват флага») — это игра, в которой несколько команд соревнуются друг с другом. В игре проверяются способности специалистов защитить сложную незнакомую систему с сохранением необходимой функциональности.

По типу, соревнования делятся на два типа: task-based (квесты), attack-defense (классические соревнования).

В случае соревнований task-based (или jeopardy) игрокам предоставляется набор тасков (заданий), к которым требуется найти ответ и отправить его. Ответ представляет собой флаг: это может быть набор символов или произвольная фраза. За верно выполненное задание команда получает определенное количество очков. Чем задание сложнее, тем больше очков будет полагаться за правильный ответ. Все задания в CTF-соревнованиях формата task-based можно разделить на несколько категорий: например, это задачи на администрирование, криптографию и стеганографию, задачи на нахождение веб-уязвимостей и любимые многими задания категории joy – развлекательные задачи разнообразной тематики.

Задача, с которой сталкиваются участники классического типа соревнований (attack-defense), сходна с реальной работой консультанта по информационной безопасности в новой организации. Необходимость защищать свой сервер и, в то же время, исследовать, атаковать и размещать свои флаги на чужих серверах делает соревнование более динамичным и зрелищным.

Команды получают идентичные образы. В ходе игры жюри будет размещать на серверах команд некоторую информацию (флаги) и проверять их доступность. Задача команд – обнаружить как можно большее количество флагов противника, при этом не давая обнаружить свои. За найденные флаги команда получает очки за нападение, за утерянные – ничего. За поддержание сервисов в рабочем и исправном состоянии команда получает очки за защиту, соответственно, за нерабочий сервис – ничего. Команда победитель определяется путем подсчета баллов по окончанию игры, победителем считается та команда, которая будет иметь большее количество баллов.

1.2 Назначение и область применения

Разрабатываемый программно-аппаратный комплекс предназначен для внедрения в сетевую инфраструктуру соревнований в области информационной безопасности Capture The Flag формата Attack-Defense, с целью сбора игрового трафика, и мониторинга состояния сети, и оборудования в ней.

1.3 Постановка проблемы

Основной задачей группы ГПО КИБЭВС-1502 в прошлом семестре была разработка программно-аппаратного комплекса для проведения соревнований в области информационной безопасности. В рамках данной задачи осенью 2016 года были проведены III ежегодные межвузовские межрегиональные соревнования по защите информации SibirCTF.

Во время проведения соревнований команда организаторов столкнулась с рядом сложностей.

1.3.1 Проблема маршрутизации, связанные со спецификой соревнований, обязательным требованием которых является наличие «маскарада» – средства сокрытия оригинального источника входящего трафика от участников команд. При большом объёме трафика, появляющегося во второй половине игры, из-за активного использования участниками автоматизированных средств эксплуатации уязвимостей, модуль «маскарада» автоматически отключается, как неспособный обработать весь объём трафика, для обеспечения продолжения функционирования базовых функций маршрутизатора.

1.3.2 Проблема сбора и хранения трафика регулярно возникает на соревнованиях CTF формата attack-defense начиная с момента их появления. Особенно остро данная проблема встала на соревнованиях SibirCTF2016. Сбор и последующее хранение трафика игровой сети необходимо осуществлять по ряду причин, среди которых:

– необходимость наличия копии трафика для вынесения жюри аргументированного решения в случае подозрения противоречащих правилам соревнований действий со стороны участников;

– необходимость последующего анализа трафика для повышения профессиональных качеств команды организаторов;

– необходимость наличия копии трафика с угрозами информационной безопасности для последующего использования в качестве обучающего материала для разрабатываемых интеллектуальных систем связанных с информационной безопасностью.

1.3.3 Проблема мониторинга нагрузки, производительности, и используемых ресурсов оборудования игровой инфраструктуры. В целях оперативного реагирования организаторами на нештатные ситуации, возникающие в процессе эксплуатации участниками команд игровой сети, необходимо постоянно отслеживать в реальном времени потребление ресурсов, текущую нагрузку, а также другие параметры используемого оборудования.

1.4 Основное направление на данный семестр

По итогам прошлого семестра ГПО был разработан программно-аппаратный комплекс для проведения соревнований по информационной безопасности, включающий в себя:

– ядро программно-аппаратного комплекса - жюрейную систему;

– сетевую инфраструктуру для проведения соревнований, состоящую из доступного команде организаторов сетевого оборудования.

Таким образом, основным направлением работы на данный семестр стало решение проблемы сбора и хранения игрового трафика, а также решение вопросов мониторинга.

2 Инструментарий

2.1 Система контроля версий Git

Для разработки программного комплекса для проведения соревнований в области информационной безопасности было решено использовать Git.

Git — распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux как противоположность системе управления версиями Subversion (также известная как «SVN»).

При работе над одним проектом команде разработчикоа необходим инструмент для совместного написания, резервного копирования и тестирования программного обеспечения. Используя Git, мы имеем:

– возможность удаленной работы с исходными кодами;

– возможность создавать свои ветки, не мешая при этом другим разработчикам;

– доступ к последним изменениям в коде, т.к. все исходники хранятся на сервере git.keva.su;

– исходные коды защищены, доступ к ним можно получить лишь имея RSA-ключ;

– возможность откатиться к любой стабильной стадии проекта.

Основные постулаты работы с кодом в системе Git:

– каждая задача решается в своей ветке;

– необходимо делать коммит как только был получен осмысленный результат;

– ветка master мержится не разработчиком, а вторым человеком, который производит вычитку и тестирование изменения;

– все коммиты должны быть осмысленно подписаны/прокомментированы.

Для работы над проектом был поднят собственный репозиторий на сервере. Исходные файлы проекта: git@gitlab2.keva.su:sibirctf/sibirctf-attack-defense.git

2.2 Система компьютерной вёрстки LaTeX

TeX — это созданная американским математиком и программистом Дональдом Кнутом система для вёрстки текстов. Сам по себе TeX представляет собой специализированный язык программирования.Каждая издательская система представляет собой пакет макроопределений этого языка.

LaTeX — это созданная Лэсли Лэмпортом издательская система на базе TeX 'а.

LaTeX позволяет пользователю сконцентрировать свои услия на содержании и структуре текста, не заботясь о деталях его оформления.

Для подготовки отчётной и иной документации нами был выбран LaTeX так как совместно с системой контроля версий Git он предоставляет возможность совместного создания

и редактирования документов. Огромным достоинством системы LaTeX то, что создаваемые с её помощью файлы обладают высокой степенью переносимости.

Совместно с LaTeX часто используется BibTeX — программное обеспечение для создания форматированных списков библиографии. Оно входит в состав дистрибутива LaTeX и позволяет создавать удобную, универсальную и долговечную библиографию. BibTeX стал одной из причин, по которой нами был выбран LaTeX для создания документации.

2.3 Система мониторинга сервисов компьютерной сети Zabbix

Zabbix — свободная система мониторинга и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования.

Архитектура Zabbix:

– Zabbix-сервер — это ядро программного обеспечения Zabbix. Сервер может удаленно проверять сетевые сервисы, является хранилищем, в котором находятся все конфигурационные, статистические и оперативные данные;

– Zabbix-прокси — собирает данные о производительности и доступности от имени Zabbix сервера. Все собранные данные заносятся в буфер на локальном уровне и передаются Zabbix серверу, к которому принадлежит прокси-сервер;

– Zabbix-агент — контроль локальных ресурсов и приложений (таких как жесткие диски, память, статистика процессора и т. д.) на сетевых системах. Эти системы должны работать с запущенным Zabbix агентом;

– Веб-интерфейс — интерфейс является частью Zabbix сервера, и, как правило (но не обязательно), запущен на том же физическом сервере, что и Zabbix сервер. Работает на PHP, требует веб-сервер (например, Apache).

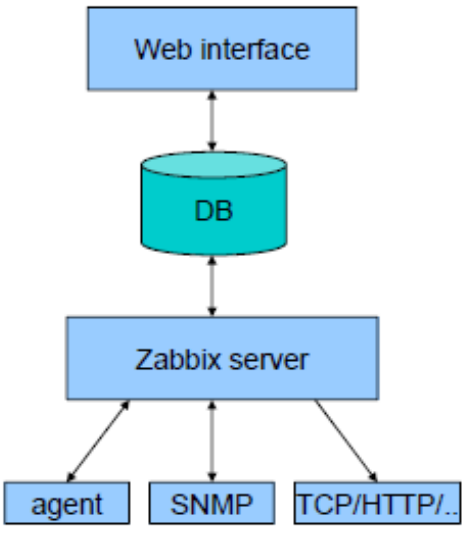

Рисунок 2.1 – Система мониторинга Zabbix

2.4 Язык программирования Python

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций.

Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Код в Python организовывается в функции и классы, которые могут объединяться в модули.

Основными преимуществами языка программирования Python являются большое количество библиотек, кроссплатформенность, широкие возможности профилирования кода.

Язык обладает чётким и последовательным синтаксисом, продуманной модульностью и масштабируемостью, благодаря чему исходный код написанных на Python программ легко читаем. При передаче аргументов в функции Python использует вызов по соиспользованию.

Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом.

2.5 Система управления проектами и задачами Teamwork

Teamwork — это сервис, предоставляющий возможность управлять своими проектами, задачами, персоналом.

Основным преимуществом является удобный интерфейс управления проектом, этапами (вехами), задачами.

Проект можно разбить на списки отдельных задач, привязывая их к определенным этапам, а также ограничивая определенным кругом лиц.

Каждой задаче можно назначить одного и более ответственного, установить приоритет, назначить дату начала и окончания, добавить описание, а также настроить оповещение в случае наступления срока окончания задачи.

Программа бесплатна при условии, что команда работает только над одним проектом.

Также доступна диаграмма Ганта.

3 Возможные пути и методы решения проблем

3.1 Методы сбора трафика

3.1.1 NetFlow — сетевой протокол, предназначенный для учёта сетевого трафика, разработанный компанией Cisco Systems. Является фактическим промышленным стандартом и поддерживается не только оборудованием Cisco, но и многими другими устройствами (в частности, Juniper, ZTE и Enterasys). Также существуют свободные реализации для UNIX-подобных систем.

Netflow предоставляет возможность анализа сетевого трафика на уровне сеансов, делая запись о каждой транзакции TCP/IP. Информация не столь подробна, как предоставляемая tcpdump'ом, но представляет довольно подробную статистику.

Netflow имеет три основных компонента:

– сенсор;
– коллектор;
– система обработки и представления данных.

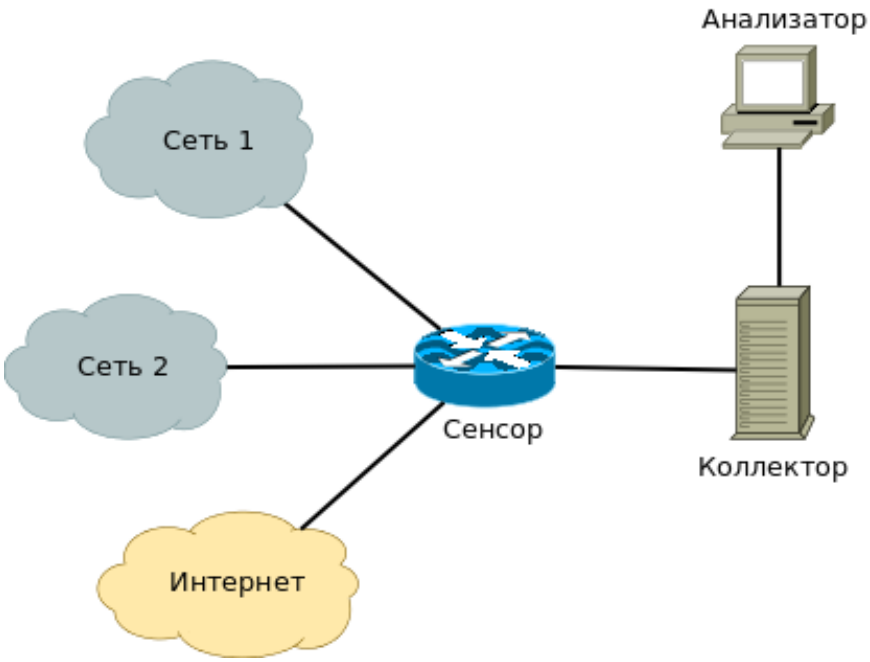

Рисунок 3.1 – Структура протокола Netflow

Сенсор — демон, который слушает сеть и фиксирует данные сеанса. Коллектор должен иметь возможность подключиться к хабу, «зеркалированному» порту коммутатора или любому другому устройству, для просмотра сетевого трафика. Если вы используете систему пакетной фильтрации на базе BSD или Linux, то это превосходное место для коллектора Netflow, так как весь трафик будет проходить через эту точку. Сенсор будет собирать информацию о сеансах и сбрасывать ее в коллектор.

Коллектор — второй демон, который слушает на UDP порту, указанному вами и осуществляет сбор информации от сенсора. Полученные данные он сбрасывает в файл для дальнейшей обработки. Различные коллекторы сохраняют данные в различных форматах.

Система обработки читает эти файлы и генерирует отчеты в форме, более удобной для человека. Эта система должна быть совместима с форматом данных, предоставляемых коллектором.

Обычно коллектор и анализатор являются частями одного программного комплекса, работающего на сервере. Разновидностей ПО коллектор/анализатор множество, платные и бесплатные, под Windows и Unix-системы.

Коллектор и стоящий за ним анализатор являются «пассивными» элементами системы. Сенсор шлет на коллектор отчеты о трафике, коллектор принимает, анализатор анализирует, и заполняет свою базу данных на сервере. По сути, при поднятом сервере, не нужно вручную подключать устройства, подпадающие под мониторинг, на сервере. Пока сенсор шлет отчеты, коллектор их принимает, анализатор регистрирует. Если сенсор выключен, он «исчезает» из текущей «онлайн» статистики.

Ввиду того, что этот протокол в большей степени предназначен для сбора статистики по трафику, сложно технически реализуем, а также не позволит производить глубокий анализ трафика в режиме реального времени, этот протокол решено не использовать.

3.1.2 tcpdump (от TCP и англ. dump — свалка, сбрасывать) — утилита UNIX (есть клон для Windows), позволяющая перехватывать и анализировать сетевой трафик, проходящий через компьютер, на котором запущена данная программа.

Для выполнения программы требуется наличие прав суперпользователя и прямой доступ к устройству (так, например, запуск из Jail во FreeBSD невозможен).

Основные назначения tcpdump в основном используется для отладки сетевых приложений и сети и сетевой конфигурации в целом.

Программа состоит из двух основных частей: части захвата пакетов (обращение к библиотеке, libpcap (Unix) или pcap (Windows)) и части отображения захваченных пакетов (которая на уровне исходного кода является модульной и для поддержки нового протокола достаточно добавить новый модуль).

Часть захвата пакетов (при запуске) передаёт «выражение выбора пакетов» (идущее после всех параметров командной строки) напрямую библиотеке захвата пакетов, которая проверяет выражение на синтаксис, компилирует его (во внутренний формат данных), а затем копирует во внутренний буфер программы сетевые пакеты, проходящие через выбранный интерфейс и удовлетворяющие условиям в выражении.

Часть отображения пакетов выбирает захваченные пакеты по одному из буфера, заполняемого библиотекой, и выводит их (в воспринимаемом человеком виде) на стандартный вывод построчно, в соответствии с заданным (в командной строке) уровнем детальности.

Если задан подробный вывод пакетов, программа проверяет для каждого сетевого

пакета, имеется ли у неё модуль расшифровки данных, и, в случае наличия, соответствующей подпрограммой извлекает (и отображает) тип пакета в протоколе или передаваемые в пакете параметры.

Если программа tcpdump вызвана для прослушивания некоторого интерфейса, она переводит его в «promiscuous mode» — «неразборчивый режим». В этом режиме интерфейс ловит вообще все пакеты, которые до него добрались, а не только пакеты адресованные непосредственно ему. Таким образом, если сеть собрана не на коммураторах (switch), а на репитерах (hub), то tcpdump позволит перехватить трафик между посторонними машинами, т.е. подслушать разговор двух сторонних машин. Сказанное не означает, что перехват трафика невозможен в сети собранной на коммутаторах. Впрочем, интерфейс можно и не переводить в promiscous mode, если передать программе аргумент -p.

Ввиду недостаточной скорости работы и невозможности глубокого анализа трафика в реальном времени, решено не использовать эту утилиту.

3.1.3 NetLimiter — коммерческое программное обеспечение для семейства ОС Windows NT, решающая проблему контроля сетевого трафика. NetLimiter после установки и запуска следит за деятельностью каждого приложения, использующего доступ к Интернету, а также активно управляет трафиком, контролируя скорость потока данных. Вы можете самостоятельно настроить скорость загрузки и отправки информации для каждого отдельного приложения или соединения.

Также NetLimiter ведет подробную статистику по всем соединениям, отображая ее в виде графиков или таблиц.

С ее помощью можно легко контролировать весь трафик приложений и программ. Отслеживать доступ в интернет сторонних программ и возможность блокировки входящего/исходящего трафика. Например, можно заблокировать доступ программе для отключения автоматических обновлений или проверки ключей или лицензий.

Ввиду того, что NetLimiter — программа для коммерческого использования, а также того, что ОС Windows не вписывается в существующую сетевую инфраструктуру, данное программное решение решено не использовать.

3.2 Обзор существующих работ по анализу трафика

3.2.1 Экспертные системы

Как правило, экспертные системы создаются для решения практических задач в некоторых узкоспециализированных областях, где большую роль играют знания «бывалых» специалистов. Экспертные системы были первыми разработками, которые смогли привлечь большое внимание к результатам исследований в области искусственного интеллекта.

Экспертные системы имеют одно большое отличие от других систем искусственного интеллекта: они не предназначены для решения каких-то универсальных задач, как напри-

мер нейронные сети или генетические алгоритмы. Экспертные системы предназначены для качественного решения задач в определенной разработчиками области, в редких случаях – областях.

Дороти Деннинг, при содействии Питера Неймана, опубликовали модель СОВ в 1986, сформировавшую основу для большинства современных систем. Её модель использовала статистические методы для обнаружения вторжений и называлась IDES (Intrusion detection expert system — экспертная система обнаружения вторжений). Система работала на рабочих станциях Sun и проверяла как сетевой трафик, так и данные пользовательских приложений. IDES использовала два подхода к обнаружению вторжений: в ней использовалась экспертная система для определения известных видов вторжений и компонент обнаружения, основанный на статистических методах и профилях пользователей и систем охраняемой сети.

Самым крупным недостатком экспертной системы в качестве системы обнаружения вторжений является неспособность в принципе обнаруживать новые виды атак. Кроме того, известно множество технологий обхода систем обнаружения вторжений на основе экспертных систем, например, polymorphic shell code, insertion, exclusion и т.п.

3.2.2 Искусственные нейронные сети

Экспертные системы хорошо себя зарекомендовали, но только в узкоспециализированных областях. Для создания более универсальных интеллектуальных систем требовался другой подход. Наверное, это привело к тому, что исследователи искусственного интеллекта обратили внимание на биологические нейронные сети, которые лежат в основе человеческого мозга.

Искусственная нейронная сеть – это математическая модель, а также её программная или аппаратная реализация, построенная по принципу организации и функционирования биологических нейронных сетей (сетей нервных клеток живого организма).

Жигулин П.В. и Подворчан Д.Э. использовали для детектирования атак двухслойной персептрон с одним скрытым слоем по схеме 38 входных, 38 скрытых, 10 выходных нейронов . Точность определения типа атаки достигла 98%.

Исследователи Моради М. и Зелкернин М. из университета Queen использовали несколько схем реализации нейросети и получили следующие результаты:

– первая схема: 35 входных, 35 скрытых нейронов первого уровня, 35 скрытых нейронов второго уровня, 3 выходных нейрона показала 91% верных решений на тестовых примерах;

– вторая схема: 35 входных, 45 скрытых, 3 выходных нейронов показала 87% верных решений на тестовых примерах;

– третья схема: 41 входных, 40 скрытых нейронов первого уровня, 40 скрытых нейронов второго уровня, 1 выходной нейрон показала 99% верных решений на тестовых примерах.

Третья схема с большой вероятностью определяет наличие атаки, но не ее тип.

Как видно из полученных результатов, увеличение числа скрытых слоев не приводит

к значительному улучшению качества работы сети (всего 4%) при экспоненциально возросшей сложности схемы и, как следствие, времени анализа.

Исследователи Клионский Д.М., Большев А.К. и Геппенер В.В. разработали систему на основе HNIDS (Heuristic Network Intrusion Detection Systems), которая использует однослойный классификатор на базе искусственных нейронных сетей (ИНС).

На рисунке ?? представлена ИНС, реализующая работу однослойного классификатора. В качестве функции активации используется сигмоидальная функция активации где:

– h – количество нейронов скрытого слоя;
– n – коэффициент скорости обучения;
– m – коэффициент инерционности.

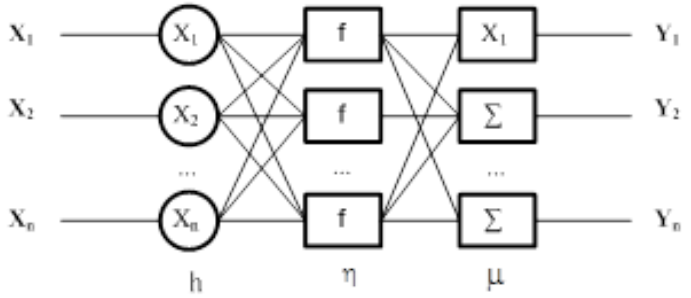

Рисунок 3.2 – Нейронная сеть, представляющая собой однослойный классификатор

Главным отличием такого типа систем от других является то, что система обучается на «нормальном» трафике целевой сети и в случае обнаружения отклонений сообщает об атаке или аномалии. Система работает на сетевом (транспортном) уровнях модели открытых систем и анализирует завершенные TCP-сессии между хостами.

При тестировании прототипа исследователями в тестовой выборке использовали 17 вторжений сетевого уровня. С помощью варьирования параметров ИНС проводились минимизации по критериям ложной тревоги (FP) и пропуска сигналов (FN). При минимизации по критерию FP, прототип обнаружил 12 атак при 2 ложных срабатываниях.

При минимизации по критерию FN, прототип обнаружил 16 атак при 1878 ложных срабатываниях.

В своих исследованиях Жульков Е.В. предложил разбить трафик на векторы и при помощи системы обнаружения вторжений (СОВ), построенной по модульному принципу, анализировать трафик как векторы.

Вероятность обнаружения известных атак составила 91%, вероятность обнаружения неизвестных атак составила 86%.

Хафизов А.Ф. предложил использовать гибридную нейронную сеть для анализа пифограмм атак. На первом этапе работы гибридной искусственной нейросети, на множестве входных векторов обучается слой Кохонена. В результате нейроны этого слоя самоорганизу-

ются таким образом, что векторы их весов наилучшим образом отображают распределение данных обучающих векторов. Далее веса фиксируются и на вход подается обучающая выборка, затем происходит финальная корректировка весов нейронов. На втором этапе обучается персептронная сеть. Обучение происходит с учителем. Для данной сети обучающие сигналы формируются из выходных сигналов слоя Кохонена и вектора ожидаемых значений.

Результатом работы такой нейронной сети является отнесение входных данных к классу атак или к классу нормальных взаимодействий.

3.2.3 Нечеткие системы

Нечеткие системы также нашли свое применение в качестве компонента системы обнаружения вторжений, так как они оперируют «нечеткими» и «размытыми» данными, которыми и являются векторы атак на вычислительные сети. В качестве примера применения нечетких систем в IDS, рассмотрим несколько работ отечественных и иностранных ученых. Исследователи из Индии Шанмагавадива Р. и Нагаражан Н. создали систему обнаружения вторжений на основе нечеткой логики. Схема работы сети представлена на рисунке ??.

Рисунок 3.3 – Схема работы СОВ на основе нечеткой логики

Авторам удалось достичь более 90% срабатываний, причем лишь 10% набора использовалось для создания базы нечетких правил.

Исследователи Слеповичев И.И., Ирматов П.В., Комарова М.С. и Бежин А.А. разработали систему обнаружения SYN Flood атак на основе нечеткой нейронной сети.Метод обучения ИНС - метод обратного распространения ошибки.

На основании разработанной модели исследователями была разработана программа, которая, используя математический аппарат нечеткой логики и нейронных сетей, определяет степень уверенности в наличии атаки.

При синтезе алгоритмов активного аудита информационной системы Кашаевым Т.Р. была разработана система обнаружения вторжений на основе искусственных иммунных систем.

Разработанная система использует нечеткие сети Петри. Система в общем случае работает следующим образом:

– определяются нормальные шаблоны активности системы (множество S) в виде строк равной длины l, составленных из букв конечного алфавита;

– генерируется набор детекторов R, каждый из которых не совпадает ни с одной из строк из нормального шаблона активности. При этом кандидат в детекторы считается совпадающим с нормальным шаблоном в том и только в том случае, когда совпадают символы в r одинаковых позициях. Величина r подбирается в соответствии с решаемой задачей;

– данные контролируются путем сопоставления детекторов с поведением системы. Любое совпадение на данном шаге означает изменение в работе системы (аномалию).

На основании разработанной модели был реализован прототип. Тестирование показало, что система обнаруживает до 85% атак.

3.2.4 Генетические алгоритмы

Генетические алгоритмы предназначены для поиска оптимального решения на основе механизма естественного отбора в популяции. Популяция представляет собой множество хромосом, каждая из которых моделируется в виде битовой строки. Популяция развивается на основе трёх генетических операций – скрещивания, селекции и мутации. Развитие популяции продолжается до тех пор, пока не будет достигнут заданный критерий оптимальности, который определяется в виде специальной функции. В случае применения генетических алгоритмов для выявления атак в качестве элементов популяции выступают вектора определённой длины, каждый элемент которых соответствует определённой атаке. В результаты развития такой популяции можно получить оптимальный вектор, который будет указывать на то, какие атаки происходят в системе в текущий момент времени.

Ученые Ануп Гоял и Четан Камар создали систему обнаружения вторжений GA-NIDS, основанную на генетическом алгоритме. Схема работы системы представлена на рисунке ??.

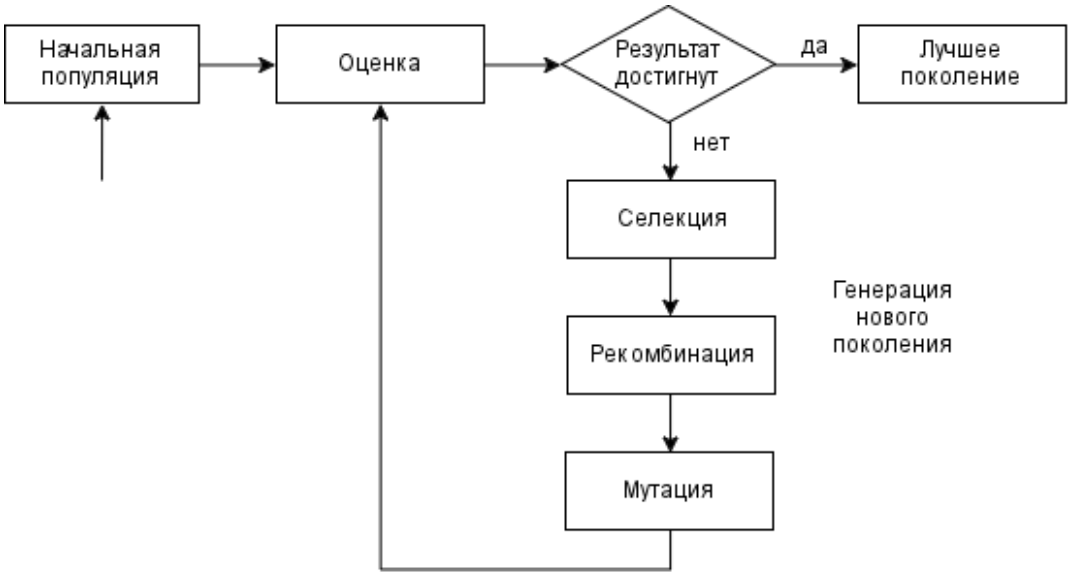

Рисунок 3.4 – Схема работы GA-NIDS

Для генерации правил использовалось 10% набора KDD CUP 99. Сгенерированные правила показали более 95% правильных решений на тестовых примерах.

3.2.5 Результат исследования

В результате проведенного исследования было решено использовать искуственные нейронный сети для системы обнаружения вторжений.

Первое преимущество в использовании нейронной сети в выявлении вторжений — это гибкость, которую предоставляет эта сеть. Нейронная сеть способна анализировать данные из сети, даже если данные неполные или искажены. Кроме того, сеть будет обладать способностью проводить анализ с данными в нелинейной форме. Обе эти характеристики имеют важное значение в сетевой среде, где полученная информация подвержена случайным ошибкам системы. Кроме того, поскольку некоторые атаки на сеть могут быть проведены скоординированным вторжением нескольких злоумышленников, способность обрабатывать данные из нескольких источников в нелинейной форме особенно важна.

Скорость, свойственная нейронным сетям, является еще одним преимуществом этого подхода. Поскольку защита вычислительных ресурсов требует своевременного выявления атак, скорость обработки нейронной сети может обеспечить реагирование на вторжение до того, как будет нанесен непоправимый ущерб системе.

Поскольку результат работы нейронной сети выражается в виде вероятности, нейронная сеть обеспечивает возможность прогнозирования для обнаружения случаев вторжения. Система обнаружения вторжений на основе нейронных сетей определит вероятность того, что конкретное событие или ряд событий, свидетельствуют о нападении на систему. По мере получения опыта, нейронная сеть улучшает способность определять, какие события и где могут произойти в процессе атаки. Эта информация затем может быть использована, чтобы

сгенерировать последовательность событий, которые должны произойти, если имеет место быть попытка вторжения. Отслеживая последующие возникновения этих событий, система будет способна улучшить анализ событий и, возможно, провести защитные меры, прежде чем атака будет удачно выполнена.

Тем не менее, наиболее важным преимуществом нейронных сетей в выявлении вторжений является способность нейронной сети «обучаться» признакам атак и определять случаи, которые нехарактерны для тех, что наблюдались ранее. Нейронная сеть может быть обучена распознавать известные подозрительные события с высокой степенью точности. Это очень ценное умение (злоумышленники часто повторяют «успехи» других) так же позволит получить возможность применять эти знания для выявления фактов о нападении, которые не соответствуют точным характеристикам предыдущих вторжений. Вероятность вторжения в систему может быть предполагаемая и помечена как потенциальная угроза, когда вероятность превышает определенный порог.

4 Фреймворк DPDK

Ввиду описанных в пункте 3.1 недостатков существующих систем сбора сетевого трафика, было принято решение разработать собственную систему сбора, удовлетворяющую следующим требованиям:

– достаточная для корректного сбора трафика скорость обработки пакетов;

– потенциальная возможность анализа всего трафика в режиме реального времени;

– соответствие и безболезненная интеграция в существующую сетевую инфраструктуру.

Таким образом, было решено разрабатывать систему сбора сетевого трафика, основанную на технологии DPDK.

DPDK — это фреймворк, который предоставляет набор библиотек и драйверов для ускорения обработки пакетов в приложениях, работающих на архитектуре Intel. DPDK поддерживается на любых процессорах Intel от Atom до Xeon, любой разрядности и без ограничения по количеству ядер и процессоров. В настоящее время DPDK портируется и на другие архитектуры, отличные от x86 — IBM Power 8, ARM и др.

DPDK позволяет полностью исключить сетевой стек Linux из обработки пакетов. Приложение, работающее в User Space, напрямую общается с аппаратным обеспечением.

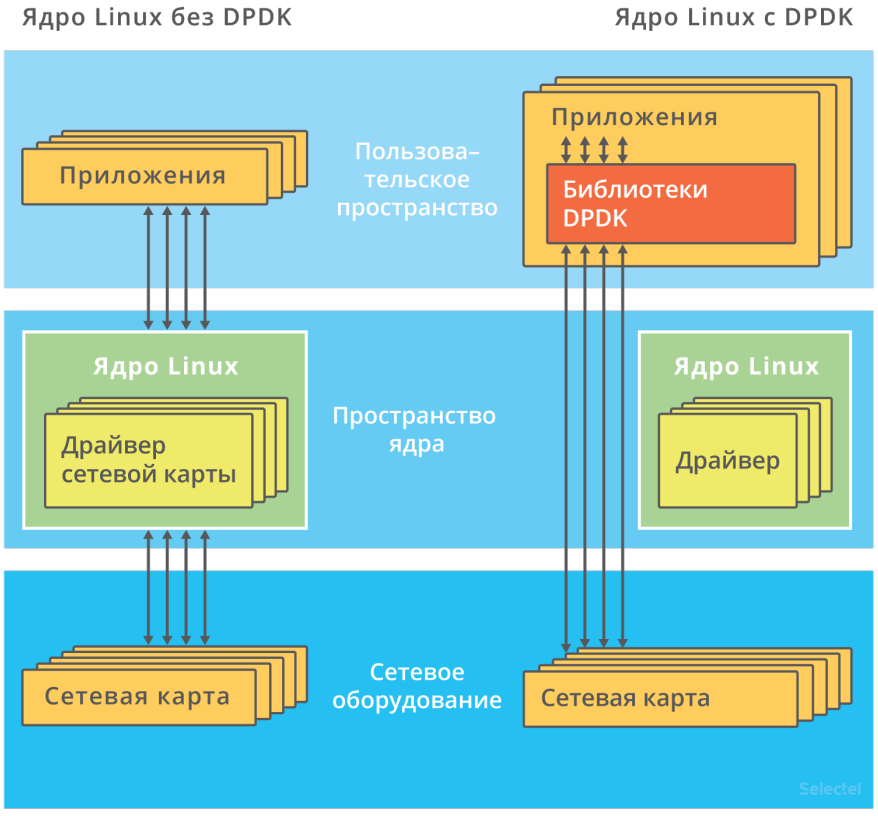

Рисунок 4.1 – Схема работы DPDK

Использование DPDK также позволяет привязывать задачу к определенному ядру. Это исключает накладные расходы, создаваемые планировщиком Linux при переключении задач. Благодаря использованию многопоточности, DPDK сокращает количество обращений к памяти и PCI, более эффективно используя процессорные мощности. Также DPDK позволяет оптимизировать использование памяти путем выравнивания структуры данных к размеру кэша, тем самым сводя к минимуму доступ к внешней памяти.

Наибольший прирост производительности можно увидеть при обработке большого количества маленьких пакетов. На графике ниже приведено сравнение производительности L3 forwarding с использованием DPDK и без него.

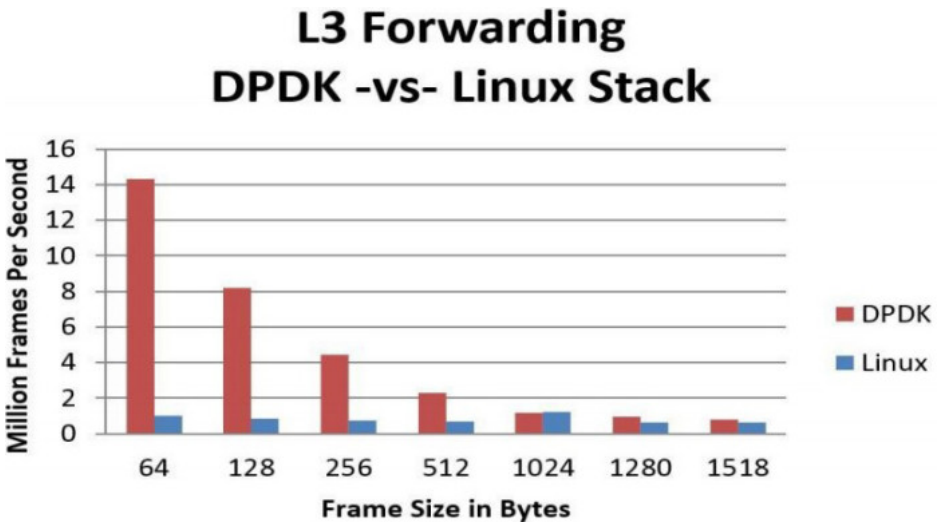

Рисунок 4.2 – Сравнение производительности L3 forwarding с использованием DPDK и без него

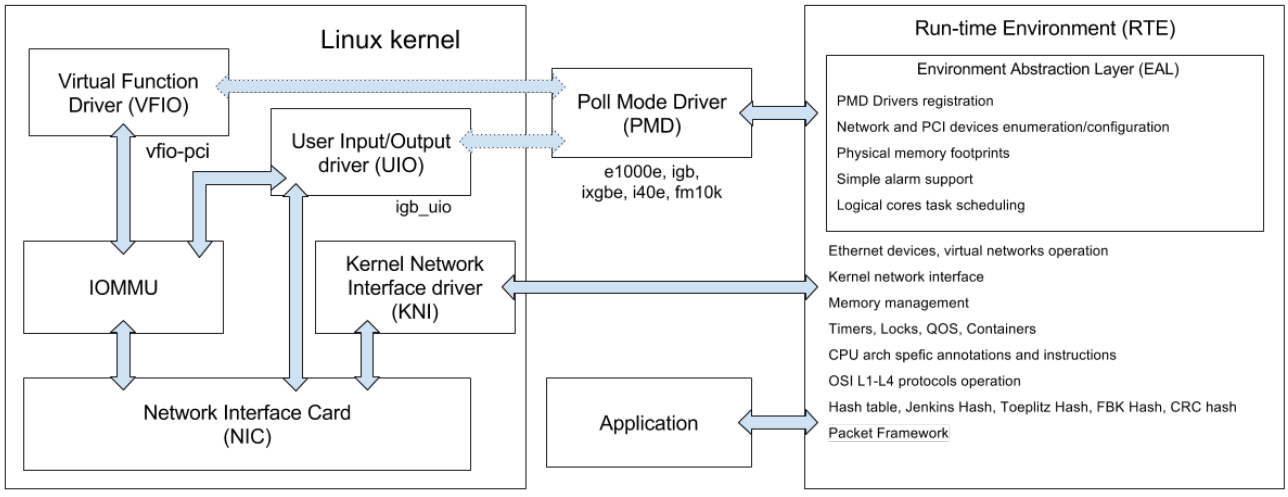

Рисунок 4.3 – Архитектура DPDK

DPDK дает возможность:

– принимать и отправлять пакеты с использованием наименьшего количества циклов ЦП (обычно не более 80 циклов);

– разрабатывать алгоритмы быстрой записи пакетов (наподобие tcpdump);

– запускать быстрые стеки сторонних разработчиков.

Производительность некоторых функций обработки пакетов составляет миллионы кадров в секунду при использовании 64-байтовых пакетов с сетевыми платами PCIe.

DPDK это OpenSource проект Intel'a, на основе которого были построены целые конторы (6WIND) и для которого производители изредка предоставляют драйвера, например Mellanox. Естественно, коммерческая поддержка решений на его основе просто замечательная, её предоставляет довольно большое количество вендоров (6WIND, Aricent, ALTEN Calsoft Labs, Advantech, Brocade, Radisys,Tieto, Wind River, Lanner, Mobica).

DPDK имеет наиболее широкий функционал, и лучше всего абстрагирует существующее железо, он создан достаточно гибким для достижения высокой, возможно максимальной, производительности. DPDK не является набором сетевых протоколов и не реализует такие функции, как перенаправление уровня 3, IPsec, брандмауэр и т.д.

5 Описание тестового стенда

В результате работы в рамках ГПО в данном семестре был собран тестовый стенд, предназначенный для зеркалирования трафика с целью его последующего сбора.

Тестовый стенд прозрачно включается в гигабитный канал связи Ethernet, после чего весь трафик, проходящий через стенд в любую сторону зеркалируется на выделенный сетевой интерфейс для его сбора отдельным модулем сбора и хранения трафика.

Аппаратные требования, предъявляемые к тестовому стенду:

– процессор Intel, поддерживающий архитектуру x86_64;

– совместимая с процессором материнская плата, не менее чем с 4 слотами расширения PCI-Express;

– 3 сетевых карты PCI-Express производства Intel, с одним из следующих NIC: ixgbe (82598, 82599, 82557, X520, X540, X550), i40e (X710, XL710, X722);

– сетевая карта PCI-Express для управления стендом;

– совместимый с материнской платой жесткий диск/твёрдотельный накопитель, объёмом не менее 20Gib;

– совместимая с материнской платой оперативная память, объёмом не менее 4Gib.

Программные требования, предъявляемые к тестовому стенду: операционная система Ubuntu 16.04 x86_64.

Итоговая аппаратная конфигурация тестового стенда:

– процессор Intel Core 2 6300 @ 1.86GHz;

– материнская плата ASUS P5B;

– сетевая карта PCI-Express Intel 82557/8/9/0/1 Ethernet Pro 100 (rev 05);

– сетевая карта PCI-Express Intel 82557/8/9/0/1 Ethernet Pro 100 (rev 08);

– сетевая карта PCI-Express Intel 82557/8/9/0/1 Ethernet Pro 100 (rev 08);

– сетевая карта PCI-Express Realtek RTL8111/8168/8411 PCI Express Gigabit Ethernet Controller (rev 01);

– жёсткий диск Seagate Barracuda, объёмом 250Gib;

– оперативная память DDR2, объёмом 4Gib.

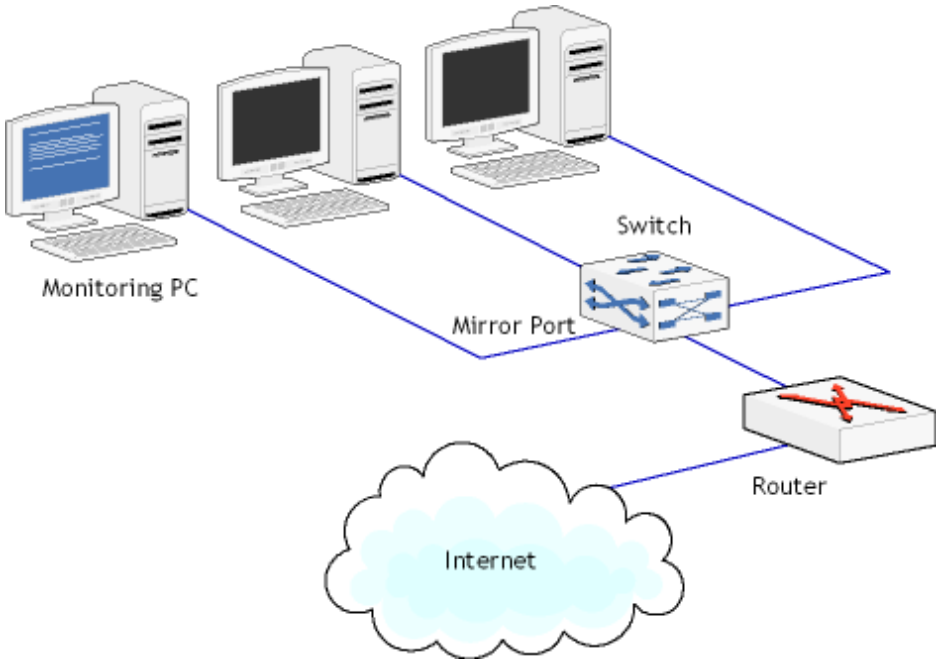

Рисунок 5.1 – Схема включения тестового стенда в сетевую инфраструктуру

Программная часть тестового стенда состоит из операционной системы Ubuntu 16.04 x86_64 с установленным фреймворком DPDK, а также программы для зеркалирования трафика, написанной на языке C, с использованием библиотек DPDK.

Перед началом работы сетевые интерфейсы стенда переводятся на использование DPDK-совместимых драйверов. После этого программа начинает зеркалировать весь трафик, проходящий между необходимыми сетевыми интерфейсами на зеркальный сетевой интерфейс. Для удалённого управления стендом используется выделенный сетевой интерфейс с запущенным SSH-сервером.

6 Тестирование

Тестирование корректности работы стенда производилось следующим образом:

1. стенд запускался;
2. стенд включался в локальную сеть между коммутатором и первым ПК, используемым для тестирования стенда
3. к зеркальному порту стенда подключался второй ПК, используемый для тестирования стенда;
4. на обоих ПК, запускалась утилита для сбора сетевого трафика tcpdump, записывающая трафик с сетевых интерфейсов ПК, подключенных к портам стенда;
5. при помощи утилиты dpdk-replay тестовый pcap-файл проигрывался на зеркалируемый сетевой интерфейс стенда, не подключенный к ПК;
6. полученные при помощи tcpdump копии трафика на ПК сличались с трафиком из проигранного pcap-файла;
7. подключенные к зеркалируемым портам стенда устройства менялись местами;
8. повторялись шаги с 4 по 6;
9. идентичность всех полученных копий трафика трафику из проигранного pcap-файла свидетельствовала о корректности работы тестового стенда.

Тестирование корректности работы стенда с сетью производилось следующим образом:

1. стенд запускался;
2. стенд включался в локальную сеть между двумя ПК, используемыми для тестирования стенда;
3. к зеркальному порту стенда подключался третий ПК, используемый для тестирования стенда;
4. на втором и третьем ПК, запускалась утилита для сбора сетевого трафика tcpdump, записывающая трафик с сетевых интерфейсов ПК, подключенных к портам стенда;
5. на втором компьютере запускалась программа vlc для воспроизведения потокового видео с первого ПК;
6. на первом ПК запускалась потоковая трансляция тестового видео, объёмом не менее 4Gib, и длительностью не менее 3 часов;
7. из полученных при помощи tcpdump копий трафика на ПК извлекался видеофайл;
8. рассчитывались хэш-суммы извлечённых видеофайлов;
9. подключенные к зеркалируемым портам стенда ПК менялись местами;
10. повторялись шаги с 4 по 8;
11. равенство всех полученных хэш-сумм хэш-сумме воспроизводимого видео-файла свидетельствовало о корректности работы тестового стенда с сетью.

По итогам тестирования стенд был признан функционирующим корректно.

7 Система мониторинга Zabbix

Для решения проблемы недостаточного контроля ресурсов сетевого и серверного оборудования был выбран и использован Zabbix.

Основные критерии при выборе были:

– свободная лицензия;

– поддержка SNMP;

– гибкая система шаблонов и групп;

– автоматическое обнаружение.

В нашем проекте Zabbix был использован для:

– мониторинга состояния сети;

– контролирование ресурсов сервера;

Была проделана следующая работа:

– был обновлен и настроен Zabbix-сервер;

– для игрового оборудования были написаны шаблоны для получения данных по SNMP и проверка на доступность;

– для сервера был установлен и настроен Zabbix-агент;

– было автоматизирован поиск и добавление zabbix-агентов в систему.

В самом начале был написан шаблон получения данных с маршрутизаторов. Вот так выглядят получаемые данные (рисунок ??).

HOSTS	ACTIVE FAN	CPU 01 LOAD	CPU 02 LOAD	CPU FREQUENCY	CPU LOAD (1ST CORE)	CPU TEMPERATURE	FLASH DISK SIZE	FLASH DISK SIZE USED	MOTHERBOARD TEMPERATURE	NAME	NUMBER OF NETWORK INTERFACES	PING	SOFTWARE ID	TOTAL MEMORY
gw.keva.su	main	0	23	1.07 MHz	0 %	45 °C	134.22 Mb	43.75 Mb	37 °C	KEVA-GW		UP (1)	5848-FNRH	1011.

Рисунок 7.1 – Получаемые данные с оборудования

Для автоматизации процесса было решено сделать проверку на доступность и триггер на срабатывание (рисунок ??).

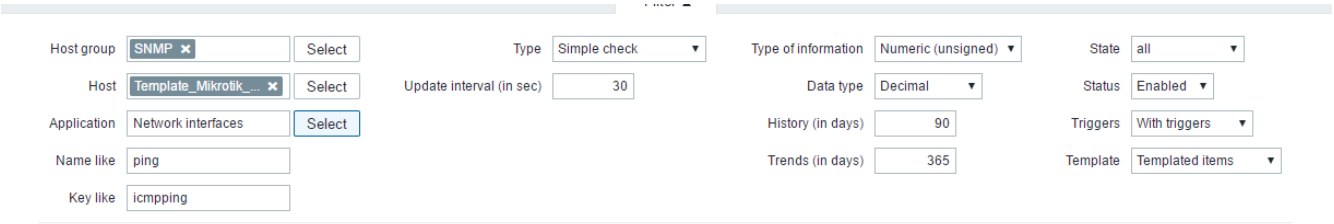

Рисунок 7.2 – Проверка на доступность

Также был настроен автоматизированный поиск агентов в указанной сети. Во время игры агенты будут предустановлены на виртуальные машины и при запуске игры система сама добавит их в свой список мониторинга (рисунок ??).

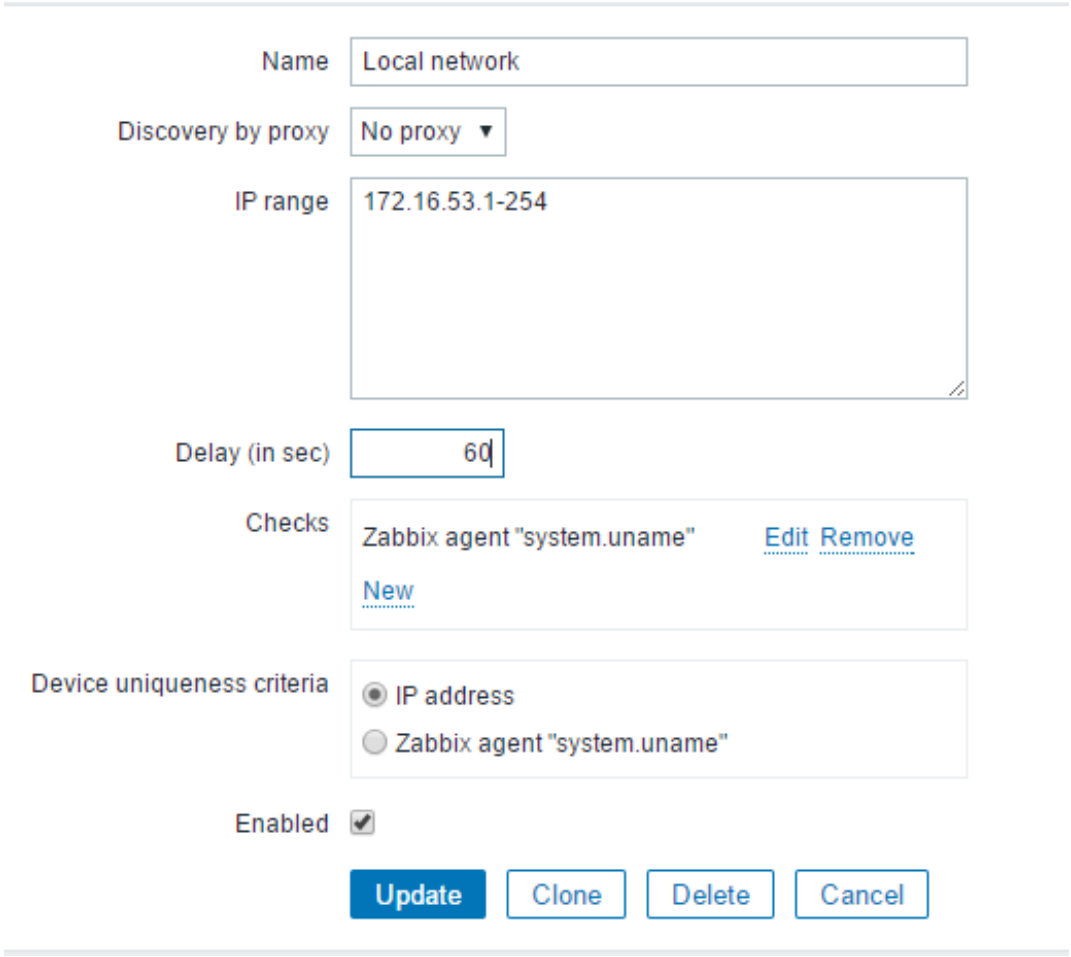

Рисунок 7.3 – Правило автоматического обнаружения

После данных настроек Zabbix-сервер готов для использования на SibirCTF в автоматизированном режиме.

8 Заключение

В результате работы ы ткущем семестре было выполнено следующее:

– проведен обзор методов для анализа трафика;

– проведено ознакомление со значительным количеством сетевых утилит и программ для захвата трафика;

– изучено устройство и принципы работы с набором библиотек DPDK;

– спроектирован и подготовлен тестовый стенд для дальнейшего использования на соревнованиях SibirCTF;

– произведено тестирование стенда, в том числе в условиях, приближенных к реальным;

– был установлен и настроен Zabbix-сервер для мониторинга нагрузки и производительности сетевого оборудования инфраструктуры соревнований.

В следующем семестре основной задачей, связанной с разработанными тестовым стендом и системой мониторинга, является их нагрузочное тестирование в боевых условиях на соревнованиях SibirCTF2017.

Главная задача следующего семестра заключается в построении программного маршрутизатора для решения проблемы маршрутизации.

Список использованных источников

– Что такое Intel DPDK? [Электронный ресурс] / SDNBLOGGER // SDNBLOG : портал. – Опубл.: 28.03.2016. – URL: https://sdnblog.ru/what-is-intel-dpdk, свободный. – Загл. с экрана (дата обращения: 22.05.2017).

– Ярош, Ю. Использование DPDK для обеспечения высокой производительности прикладных решений (часть 0) [Электронный ресурс] // Хабрахабр : сайт. – Опубл.: 10.12.2015. –URL: https://habrahabr.ru/post/267591/, свободный. – Загл. с экрана (дата обр.: 22.05.2017).

– Programmer's Guide : [Electronic resource] // DPDK: Data Plane Development Kit : [website]. San Francisco, 2013–2017. – URL: http://dpdk.org/doc/guides/prog_ guide/, free. – Tit. screen (usage date: 22.05.2017).

– Сердюк, В. Вы атакованы – защищайтесь (методология выявления атак) [Электронный ресурс] // Диалог Наука : сайт. – М., 1998–2017. – URL: https://www.dialognauka.ru/press-center/article/4770, свободный. – Загл. с экрана (дата обращения: 22.05.2017).

– Scott Chacon. Pro Git: professional version control. 2011.
URL: http://progit.org/ebook/progit.pdf.

– С.М. Львовский. Набор и вёрстка в системе L A TEX. МЦНМО, 2006. С. 448.

– И. А. Чеботаев, П. З. Котельников. L A TEX 2 по-русски. Сибирский Хронограф, 2004. 489 с.

– Python. [Электронный ресурс] // ru.wikipedia.org:[сайт]. 2015.
URL: https://ru.wikipedia.org/wiki/Python.

– Зенов, А. Ю. Применение нейросетевых алгоритмов в системах охраны периметра / А. Ю. Зенов, Н. В. Мясникова // Изв. вузов. Поволж. регион. Техн. науки. – 2012. – № 3. – С. 15–24.

– Исаев, С. Популярно о генетических алгоритмах [Электронный ресурс] // Algolist.manual.ru: Алгоритмы. Методы. Исходники : сайт. – 2000–2017. – URL: http://algolist.manual.ru/ai/ga/ga1.php# ga , свободный. – Загл. с экрана (дата обращения: 22.05.2017).

– Генетические алгоритмы. От теории к практике [Электронный ресурс] / knok16 // Хабрахабр : сайт. – Опубл.: 13.02.2012. –URL: https://habrahabr.ru/post/138091, свободный. – Загл. с экрана (дата обращения: 22.05.2017).

– Генетический алгоритм. Просто о сложном [Электронный ресурс] / Марк@mrk-andreev // Хабрахабр : сайт. – Опубл.: 20.09.2011. – URL: https://habrahabr.ru/post/128704, свободный. – Загл. с экрана (дата обращения: 22.05.2017).

– Камаев, В. А. Методология обнаружения вторжений / В. А. Камаев, В. В. Натров // Изв. Волгогр. техн. ун-та. – 2006. – № 4. – С. 148–153.

– Введение в нечеткие системы [Электронный ресурс]. – URL: http://www.igce.comcor.ru/AI_ mag/NN/FuzzySystems/FuzzySystems.html, свободный. – За-

гл. с экрана (дата обращения: 22.05.2017).

– Нейронные сети [Электронный ресурс] // AIPortal : портал. – 2009–2017. – URL: http://www.aiportal.ru/articles/neural-networks/neural-networks.html, свободный. – Загл. с экрана (дата обращения: 22.05.2017).

– Большев, А. К. Применение нейронных сетей для обнаружения вторжений в компьютерные сети /, А. К. Большев, В. В. Яновский // Вестн. СПбГУ. Сер. 10, Приклад. математика:Информатика. Процессы упр. – 2010. – Вып. 1. – С. 129–135.

– Scholz D. A Look at Intel's Dataplane Development Kit [Electronic resource] // Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM) : Proc. of the Seminars, Summer Semester / Eds. Georg Carle, Daniel Raumer, Lukas Schwaighofer. – Munich, 2014. – Vol. NET-2014-08-1. – P. 115–123. – doi: 10.2313/NET-2014-08-1_ 15

– Жигулин, П. В. Анализ сетевого трафика с помощью нейронных сетей [Электронный ресурс] / П. В. Жигулин, Д. Э. Подворчан // ТУСУР : информ. портал. – URL: http://storage.tusur.ru/files/425/КИБЭВС-1005_ Жигулин_ П.В._ _ Подворчан_ Д.Э.pdf

– Мустафаев, А. Г. Нейросетевая система обнаружения компьютерных атак на основе анализа сетевого трафика // Вопр. безопасности. – 2016. – № 2. – С. 1–7.

– Сухов, В. Е. Система обнаружения аномалий сетевого трафика на основе искусственных иммунных систем и нейросетевых детекторов // Вестн. РГРТУ. – 2015. – № 54, ч. 1. – С. 84–90.

– Басараб, М. А. Анализ сетевого трафика корпоративной сети университета методами нелинейной динамики [Электронный ресурс] / М. А. Басараб, А. В. Колесников, И. П. Иванов // Наука и образование. – 2013. – № 8. – С. 341–352. – doi: 10.7463/0813.0587054

– Гончаров, В. А. Исследование возможностей противодействия сетевым информационным атакам со стороны защищенных ос и систем обнаружения информационных атак / В. А. Гончаров, В. Н. Пржегорлинский // Вестн. РГРТА. 2007. – Вып. 20. – С. 10–14.

Приложение А
(Обязательное)
Компакт - диск

Компакт-диск содержит:

– электронную версию пояснительной записки в форматах *.tex и *.pdf;
– исходные коды программной составляющей тестового стенда;
– исходные коды набора библиотек DPDK.