**УДК 004.415.2.031.43**

# ПОДХОД К ПРОЕКТИРОВАНИЮ МНОГОКОМПОНЕНТНОЙ СИСТЕМЫ ОБНАРУЖЕНИЯ СЕТЕВЫХ АТАК

***П.А. ЖУРАВКОВ***
***(Представлено: Е.Р. СУХАРЕВ)***

*Рассматриваются походы к проектированию служб, осуществляющих сбор и обработку сетевых пакетов, поступающих в корпоративную систему. Анализируется принцип разработки современных веб-приложений. Представлены два компонента системы обнаружения атак: агент, выполняющий сбор и обработку сетевых пакетов, и клиентское веб-приложение.*

В связи с проникновением новых сетевых технологий в различные сферы деятельности человека становятся всё более актуальными вопросы защиты компьютерных сетей от постороннего вмешательства. Ведущими IT-компаниями мира в настоящее время разрабатываются различные принципы организации защиты облачных вычислений. Актуальность подобных исследований следует из современных возможностей подключения вычислительных ресурсов к сетям Интернет и предоставления пользователям облачных услуг, что, в свою очередь, создает реальную угрозу их безопасности. Обратим внимание на два компонента системы обнаружения атак: агент, выполняющий (Windows-служба) сбор и обработку сетевых пакетов, работающий на узлах корпоративной системы, и клиентское веб-приложение.

**Агенты сбора и обработки сетевых пакетов**

Основная цель работы агента заключается в сборе и анализе количества, типов, IP-адрес, откуда пакет был послан, а также размера принимаемых и передаваемых пакетов на узле. Разрабатываемый компонент должен быть реализован в виде Windows-сервиса, который после загрузки операционной системы способен автоматически запуститься.

Фундаментальная задача разрабатываемой службы – бесперебойная работа 24/7 основных компонент, отвечающих за сбор и анализ данных. В противном случае система перестанет анализировать данные, что может вылиться в серьезные убытки для компании, содержащей атакуемый узел. По данным Лаборатории Касперского за 2015 год, потери компаний пострадавших от сетевых атак, основанных на небезграничности ресурсов атакуемой системы, в среднем составляют до 400$ тыс. В среднем 61% пострадавших временно теряли доступ к критичной информации, 38% не могли осуществлять свою непосредственную деятельность, а 33% компаний сообщили об упущенных сделках и контрактах. В связи с этими фактами система анализа и мониторинга обязана быть доступна в любое время.

Для достижения этой цели необходимо продумать и реализовать механизм проверки работы агента. На основании одного из пункта в настройках служба должна отправлять подобие ping-запроса в базу данных, сообщая тем самым информацию о том, что она корректно функционирует [2].

Необходимо разработать такую инфраструктуру, которая позволяла бы продолжить работу какой-либо подсистемы даже после ошибки в ней. В качестве основы для такой системы необходимо взять подход на основе потоков, когда для единицы работы создается отдельный поток (Task) операционной системы. Потоки, они же задачи, в операционных системах позволяют абстрагироваться от текущего последовательного выполнения программы. Единица работы начинает выполняться в отдельной нити, позволяя полностью отстраниться от главного контекста выполнения программы. Главным плюсом данного выбора в данной ситуации является то, что если в потоке происходит ошибка выполнения, она никак не влияет на работу остальной системы. Система продолжит работу.

В случае если какая-то подсистема службы вышла из строя (произошла ошибка выполнения программы), эта же подсистема должна корректно обработать произошедшую ошибку и продолжить корректно работать. Одним из паттернов потокового проектирования является реализация так называемых менеджеров задач, которые отвечают на создание, выполнение задач, а также обработку ошибок в этих задачах. Весь процесс создания задачи берет на себя менеджер, автоматически обертывая единицу работы в поток. Такая реализация позволяет значительно сократить объем кода, а также улучшить читаемость кода. После того как поток создан, его необходимо запустить на выполнение в отдельной нити. Менеджер автоматически запускает задачу после ее создания. Наконец, если в потоке произошла ошибка выполнения, менеджер обработает эту ошибку, тем самым предотвратив аварийное завершение работы всей системы [4].

Одной из задач проектирования агента является возможность удаленного конфигурирования службы. Данный функционал необходимо реализовать как часть клиентского веб-приложения. Также одним из требований является незаметное для работы агента изменение текущих настроек агента во вре-

мя его работы, то есть служба должна автоматически подгружать необходимые настройки из базы данных и применять их, не нарушая и не останавливая свою работу. Данный функционал необходимо реализовать с помощью кеш-механизма, который должен через какой-то промежуток времени проверять актуальные настройки. Основной критерий выбора данного подхода – уменьшение нагрузки на базу данных.

Основной задачей агента, исходя из требований, является сбор данных для последующего анализа. Единицей данных, передаваемой по сети, служит сетевой пакет – это блок данных, который сформирован определенным образом. Основной задачей злоумышленника, который хочет нарушить работу какого-либо узла в системе, является многократное увеличение передаваемых на узел пакетов, тем самым узел не в состоянии справиться с таким количество информации. В итоге, легальные пользователи не могут частично или в полной мере пользоваться корпоративной системой. Сбор информации о количестве пакетов, а также их содержимого является первостепенной задачей.

Захват пакетов должен быть реализован на уровне драйвера сетевой карты, который использует NDIS для чтения пакетов, которые получает сетевая карта. Существует библиотека Pcap, реализующая всю низкоуровневую логику по работе с данными драйвера, тем самым позволяя программисту миновать работу напрямую с драйвером. Рассматриваемая библиотека написана на языке программирования C, что не всегда удобно при работе с языками более высокого уровня, такими как Java и C#. Поэтому в разработке необходимо использовать обертку для Pcap на языке разработки дипломной работы – SharpPcap. SharpPcap реализует весь функционал библиотеки Pcap, дополняя ее специфическими для языка C# особенностями, а именно событийной моделью обработки поступающих пакетов. При разработке также необходимо учитывать размер и тип пакетов. Низкоуровневая библиотека Pcap реализует обработку следующих типов пакетов: ARP, IPv4, IPv6, ICMP, ICMPv6, IGMP, UDP, TCP. Все перечисленные типы пакетов разрабатываемая система должна собирать и подсчитывать. Для этого в режиме реального времени отдельный поток (задача) выполняет количественный подсчет каждого пакета, а также его размер и тип [4].

Подсистема фильтров должна осуществлять фильтрацию всех вышеперечисленных типов пакетов. На стадии проектирования необходимо продумать концепцию переиспользования фильтров, так называемых многоразовых фильтров, которые могут быть применены для разных сетевых устройств на разных узлах в корпоративной системе, что позволит удобно управлять процессом настройки подсистемы фильтров. На основе кеш-механизма подгрузки настроек агента нужно реализовать кешированную подгрузку фильтров из базы данных для сетевых устройств на данном узле, где работает агент.

**Веб-приложение**

Основной задачей клиентской веб-части является возможность настройки агентов, а также показ информации об их текущей нагрузки. Настройка агентов включает в себя отображение информации о работе агентов на узлах; вывод ошибок агентов, позволяющий специалисту по информационной безопасности наблюдать за корректностью работы агентов; вывод срабатываний различных фильтров на соответствующих узлах. Также веб-приложение должно показывать в режиме реального времени данные, собранные с помощью агентов сбора и обработки на узлах, а также строить наглядный график.

Веб-приложение должно быть разработано с применением паттерна MVC. MVC – паттерн проектирования, с помощью которого модель приложения, интерфейс пользователя и взаимодействие с пользователем разделены на три отдельных компонента так, чтобы изменение одного из компонентов минимально влияло на работу остальных. Контроллер перехватывает событие извне и в соответствии с заложенной в него логикой реагирует на это событие, изменяя модель посредством вызова соответствующего действия. Затем модель использует событие о том, что она изменилась, и все подписанные на это события отображения, получив его, обращаются к модели за обновленными данными и отображают их. Таким образом, придерживаясь такого подхода на протяжении всего цикла разработки ПО, код становится масштабируемым и легко читаемым. Схема работы MVC подхода показана на рисунке 1.

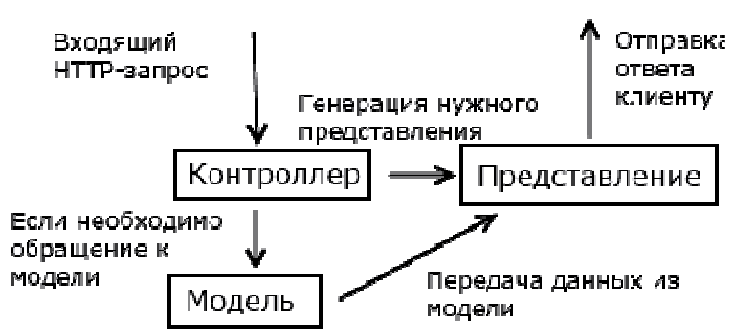


**Рисунок 1. – Функциональная схема устройства MVC патттерна**

Рассмотрим реализацию пользовательского интерфейса. На главной странице должна быть представлена общая сводка статистики со всех узлов. Для более наглядного восприятия на странице должны располагаться график нагрузки на узлах, график процентного соотношения типов пакетов, график количества пакетов на всех узлах и график количества трафика на узлах. Макет описываемой страницы представлен на рисунке 2.

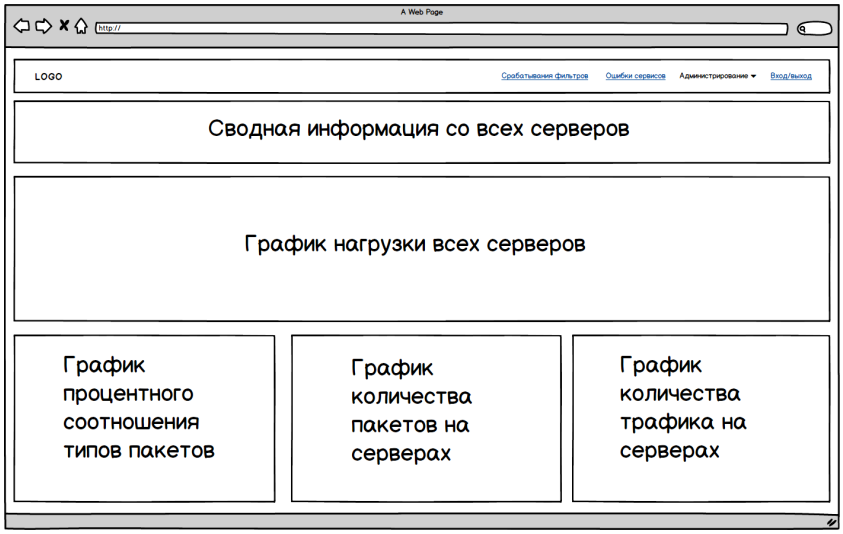


**Рисунок 2. – Макет главной страницы веб-приложения**

ЛИТЕРАТУРА

1. Система обнаружения вторжений – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https://ru.wikibooks.org/wiki/Система_обнаружения_вторжений. – Дата доступа: 26.09.2016.
2. Ping – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https: //ru.wikibooks.org/wiki/Ping. – Дата доступа: 25.09.2016.
3. Обработка исключений (библиотека параллельных задач) – MSDN – сеть разработчиков Microsoft [Электронный ресурс]. – 2016. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/dd321409(v=vs.110).aspx. – Дата доступа: 25.09.2016.
4. Pcap – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https: //ru.wikibooks.org/wiki/Pcap. – Дата доступа: 27.09.2016.
5. Model-View-Controller – Викиучебник / Wikimedia Foundation Inc. [Электронный ресурс]. – 2016. – Режим доступа: https://ru.wikibooks.org/wiki/Model-View-Controller. – Дата доступа: 27.09.2016.