Лабораторная работа №8

Анализ трафика с помощью программы Wireshark

Теоретическая часть

Система мониторинга сервера — специальное программное обеспечение, предназначенное для непрерывного контроля и анализа работы сервера. Она позволяет отслеживать основные параметры сервера: загрузка процессора, использование памяти, объем дискового пространства, сетевая активность и другие ключевые показатели.

В простейшем случае, при работе на одной системе, например на смартфоне, возможно использовать одну из множества утилит, которая снимет значения нагрузки процессора, занимаемого объема ОЗУ или же свободного места на носителе (CPU-Z и другие). Если мы работаем на компьютере с ОС Linux, то возможно использовать консольные утилиты htop, netstat и др. Каждая из утилит применяется для мониторинга отдельных параметров - такой подход не применим для централизованного сбора информации. Поэтому необходимо использовать специализированные системы мониторинга для анализа системы.

С необходимостью систем мониторинга люди столкнулись при анализе производительности LAN сетей в рамках одного офиса. В них использовался специальный протокол SNMP (Simple Network Management Protocol), который позволял настраивать двусторонний или односторонний доступ к различным сетевым устройствам: от роутеров и коммутаторов до принтеров. На его основе настраивались такие инструменты как Big Brother и nmon, которые предоставляли информацию о сетевых нагрузках и событиях в сети, используя для этого сетевое соединение.

С появлением веб-сайтов и интернет-сервисов стало необходимо модифицировать системы. В результате были разработаны веб-ориентированные легко масштабируемые инструменты, поддерживающие интернет-протоколы. Среди популярных open-source инструментов начала 2000-х — Zabbix, Nagios и Cacti. В дальнешем весь рост систем мониторинга происходил в сторону увеличения количества метрик или же разнообразия объектов для снятия этих метрик.

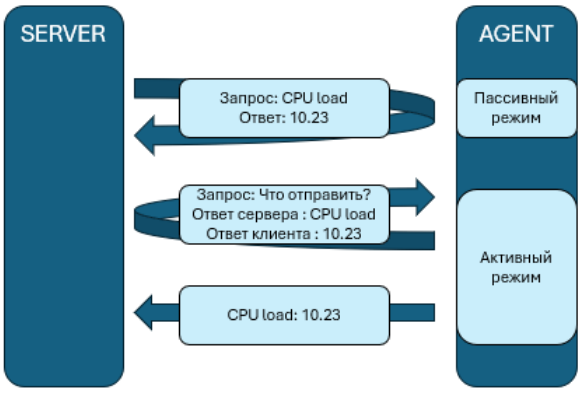
На сегодня основными системами мониторинга являются:

- **PingInfoView, Pingdom** – простые системы, использующие ping для проверки доступности узлов в сети. Позволяют с интервалом проверять состояние узлов и отображать доступность в режиме реального времени.
- **Zabbix** – мощная система с поддержкой агентов и безагентного сбора данных (SNMP, IPMI, HTTP и т.д.), средствами анализа данных, уведомлениями и веб-интерфейсом.

- **PRTG Network Monitor** – система с широким набором сенсоров для сбора данных (SNMP, WMI, ICMP и пр.), анализом данных, уведомлениями и удобным интерфейсом. В отличие от многих других конкурентов имеет ограничение в 100 метрик (сенсоров).
- **Graphite** — это бесплатный инструмент с открытым исходным кодом (FOSS), который строит графики числовых временных рядов, таких как производительность компьютерных систем. Graphite собирает, хранит и отображает данные временных рядов в реальном времени.
- **Prometheus** – система, в основе которой лежит база данных временных рядов (Time series database, TSDB). Поддерживает экспортеры и PushGateway (возможность отсылать метрики из кастомных скриптов и программ), уведомления и интеграцию с Grafana для визуализации.

Существуют две основные модели мониторинга:

- Push-модель – сервер мониторинга ожидает подключений от агентов для получения метрик. Модель обычно используется для мониторинга больших систем, где количество устройств может быть слишком большим для ручного сбора данных.
- Pull-модель – сервер мониторинга сам подключается к агентам мониторинга и забирает данные.



Современные системы, такие как Zabbix и Prometheus работают как Push и Pull модель. Обе системы являются очень популярными на рынке.

Wireshark

Wireshark — программа-анализатор трафика для компьютерных сетей Ethernet и некоторых других. Имеет графический пользовательский интерфейс.
Для установки пакета необходимо воспользоваться программой:

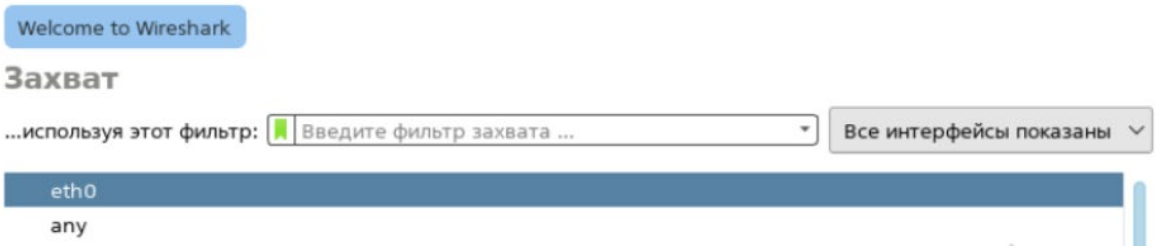
```
sudo apt install wireshark
```

При установке возможно указать, чтобы пользователи не обладающие правами суперпользователя могли производить захват трафика.

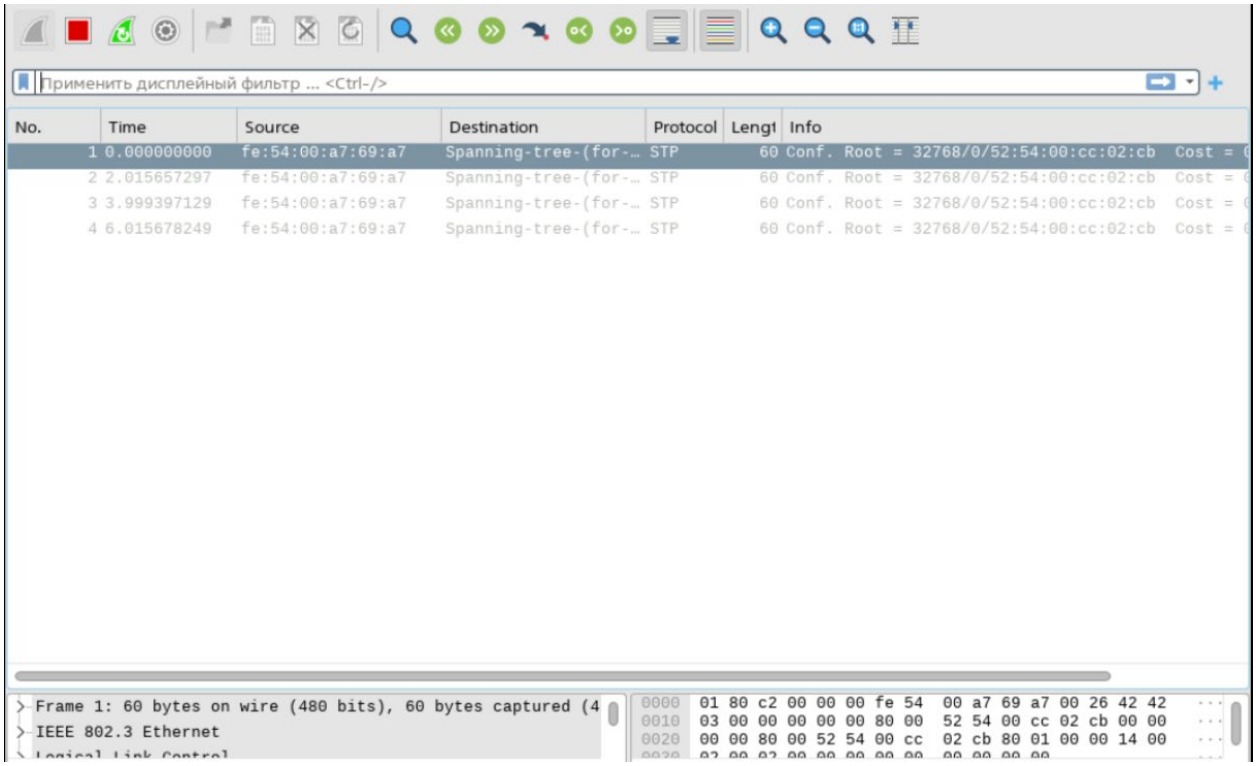
Для запуска программы воспользуйтесь командой:

```
sudo wireshark
```

После запуска программы выберите интерфейс, который вы будете прослушивать:



Далее открывается рабочее окно, в котором возможно наблюдать весь сетевой трафик, проходящий через указанный интерфейс.



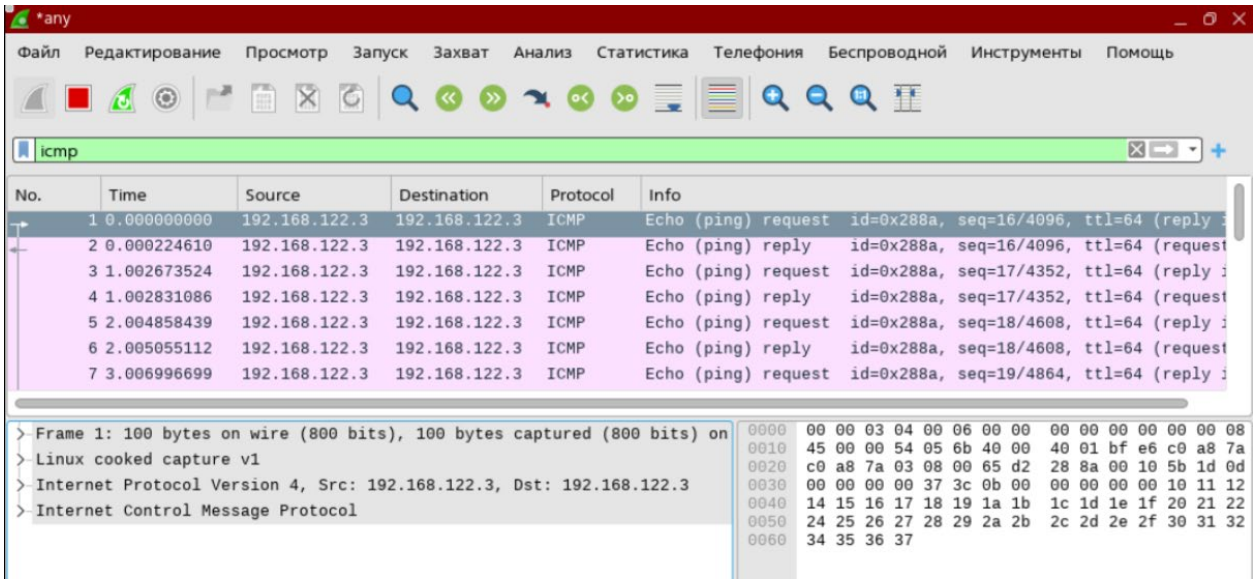
Для отбора необходимых строк возможно использовать фильтры.

Название фильтра	Комментарий	Пример
<Название протокола>	Фильтрация пакетов с определенных,	icmp
ip.src==<ip адрес>	Фильтрация по адресу источника сообщений	ip.src==192.168.122.2
ip.dst==<ip адрес>	Фильтрация по адресу назначения сообщений	ip.dst==192.168.122.3
<Название протокола>.port == <номер порта>	Фильтрация по порту	tcp.port == 80

С помощью символов || фильтры возможно объединять

Например: icmp || tcp || udp

Для анализа пакета возможно воспользоваться окошками, расположенными ниже:



В правом окошке представлен пакет в «сыром» виде, как последовательность байт. В левом окне выводится его расшифровка.

Более подробно по программе wireshark возможно прочитать в источнике 1.

Практическая работа

1. Wireshark
   a. Запустите программу wireshark на сервере
   b. Установите фильтр, выбрав ip-адреса клиента, сервера и сетевого моста.
   c. Выполните следующие команды и проанализируйте результат их работы в wireshark:
      i. ping с сервера на клиент
      ii. ping на адрес, расположенный вне сети
      iii. ping с клиента на сервер по доменному имени сервера
      iv. ping с клиента на сервер по доменному имени сервера, не включенного в список DNS
      v. открытие веб-страницы из предыдущей лабораторной работы на клиенте
      vi. обновление адреса DHCP на клиенте с помощью команд dhclient -r; dhclient
      vii. обмен TCP сообщениями между клиентом и сервером (запуск файлов tcpserver.py, tcpclient.py, исходные коды в приложении)
      viii. обмен UDP сообщениями между клиентом и сервером (запуск файлов udpserver.py, udpclient.py, исходные коды в приложении)

Список литературы:

[1] https://wireshark.wiki

Приложение:

tcpclient.py

```
import socket

HOST = "127.0.0.1"
PORT = 65432
bufferSize = 1024

TCPClientSocket=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
TCPClientSocket.connect((HOST, PORT))
TCPClientSocket.sendall(b"Hello, world")
data = TCPClientSocket.recv(bufferSize)
print(f"Received {data!r}")
```

tcpserver.py

```
import socket

HOST = "127.0.0.1"
PORT = 65432
bufferSize  = 1024

TCPServerSocket=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
TCPServerSocket.bind((HOST, PORT))
TCPServerSocket.listen()

TCPAnswer=TCPServerSocket.accept()
connection = TCPAnswer[0]
address = TCPAnswer[1]
print(f"Connected by {address}")
data = connection.recv(bufferSize)
connection.sendall(data)
```

udpclient.py

```
import socket

msgFromServer       = "Hello world!"
bytesToSend         = str.encode(msgFromServer)

serverAddressPort   = ("127.0.0.1", 65432)
bufferSize          = 1024

UDPClientSocket=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
UDPClientSocket.sendto(bytesToSend, serverAddressPort)
data = UDPClientSocket.recv(bufferSize)
print(f"Received {data!r}")
```

udpserver.py

```
import socket
```

```
HOST = "127.0.0.1"
PORT = 65432
bufferSize  = 1024

UDPServerSocket=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
UDPServerSocket.bind((HOST, PORT))

UDPAnswer=UDPServerSocket.recvfrom(bufferSize)
connection = UDPAnswer[0]
address = UDPAnswer[1]
print(f"Connected by {address}")
UDPServerSocket.sendto(connection,address)
```