ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра вычислительных систем

КУРСОВОЙ ПРОЕКТ

по дисциплине

“ Сетевое программирование ”

Реферат по теме «Протокол SNMP»

Выполнил студент Шевельков П.С
Ф.И.О.

Группы ИП-015

Работу принял ассистент кафедры ВС Третьяков Глеб Николаевич
подпись

Защищена Оценка

Новосибирск – 2023

Оглавление

Введение

Определение протокола SNMP.

Протокол SNMP (Simple Network Management Protocol) является одним из основных протоколов управления сетями. Он разработан для мониторинга и управления сетевыми устройствами, такими как маршрутизаторы, коммутаторы, серверы и принтеры. SNMP позволяет сетевым администраторам собирать информацию о состоянии устройств, настраивать и контролировать их работу.

Основными компонентами SNMP являются управляемые устройства, агенты SNMP и системы управления сетью (SNMP NMS). Управляемые устройства, такие как сетевые маршрутизаторы, имеют агента SNMP, который отвечает за сбор и передачу информации о состоянии устройства. SNMP NMS выполняет функцию мониторинга и управления сетью. Он отправляет запросы агентам SNMP для получения информации и настройки устройств.

История развития SNMP

Протокол SNMP (Simple Network Management Protocol) был разработан в 1988 году компанией Carnegie Mellon University, а затем стандартизирован в рамках Internet Engineering Task Force (IETF). Начиная с первой версии SNMPv1, протокол прошел через несколько этапов развития и улучшений.

SNMPv1 был первой версией протокола, в которой были определены основные концепции и операции. Он использовал простой синтаксис и предоставлял возможность считывания и записи информации с устройств. Однако SNMPv1 был ограничен в своих возможностях безопасности и не предоставлял механизмов аутентификации и шифрования.

В ответ на это, SNMPv2 был представлен в 1993 году. Он включал улучшенные механизмы безопасности, включая модель аутентификации и авторизации. SNMPv2 также добавил новые операции, такие как массовое получение (GETBULK) и оповещения (TRAP), которые позволяли более эффективное управление сетью.

В 1998 году SNMPv3 был представлен как ответ на проблемы безопасности, связанные с предыдущими версиями. SNMPv3 включал расширенные

механизмы аутентификации, аутентификацию и шифрование, обеспечивая конфиденциальность и целостность передаваемых данных. Он также предоставлял более гибкие возможности управления доступом.

Клиент-серверная модель протокола SNMP

Протокол SNMP (Simple Network Management Protocol) основан на клиент-серверной модели, где взаимодействие происходит между системой управления сетью (SNMP NMS) в роли клиента и управляемыми устройствами (агентами SNMP) в роли сервера. Давайте рассмотрим эту модель подробнее.

1. SNMP NMS (Network Management System): SNMP NMS представляет собой систему управления сетью, которая выполняет функции мониторинга и управления устройствами в сети. Она является клиентом в клиент-серверной модели SNMP. SNMP NMS отвечает за инициирование запросов и получение информации от агентов SNMP. Она может быть реализована в виде специализированного программного обеспечения или инструментария управления сетью.
2. Агенты SNMP: Агенты SNMP являются серверами в клиент-серверной модели SNMP. Они представляют собой управляемые устройства, такие как маршрутизаторы, коммутаторы, серверы и принтеры. Агенты SNMP установлены на каждом управляемом устройстве и отвечают за сбор и передачу информации о состоянии устройства SNMP NMS. Они также принимают команды от SNMP NMS для выполнения определенных операций, таких как изменение настроек устройства.
3. Протоколы транспортного уровня: Для обмена данными между SNMP NMS и агентами SNMP используются различные протоколы транспортного уровня, такие как UDP (User Datagram Protocol) или TCP (Transmission Control Protocol). SNMP обычно использует UDP для отправки сообщений, поскольку UDP обеспечивает простую и быструю доставку, хотя без гарантии доставки и управления соединением, которые предоставляет TCP.
4. Операции SNMP: SNMP NMS и агенты SNMP взаимодействуют с помощью операций SNMP, таких как GET (получение), SET

(установка), GETNEXT (получение следующего), GETBULK (массовое получение) и TRAP (оповещение). SNMP NMS отправляет запросы агентам SNMP, используя эти операции, чтобы получить информацию о состоянии устройств или изменить их настройки. Агенты SNMP отвечают на запросы и предоставляют запрошенную информацию или выполняют указанные операции.

Структура SNMP-сообщения.

SNMP-сообщение состоит из заголовка и тела, которые определяют информацию о запросе или уведомлении, передаваемом между системой управления сетью (SNMP NMS) и агентом SNMP. Давайте рассмотрим структуру SNMP-сообщения подробнее:

1. Заголовок SNMP-сообщения: Заголовок содержит информацию о версии протокола SNMP и используемом сообществе (community). Community является простым паролем, который используется для аутентификации и авторизации доступа SNMP NMS к агенту SNMP. Заголовок также содержит идентификатор операции, такой как GET, SET, GETNEXT, GETBULK или TRAP, который указывает на тип операции, выполняемой в SNMP-сообщении.
2. Тело SNMP-сообщения: Тело SNMP-сообщения содержит переменные привязки (bindings) и данные, связанные с запросом или уведомлением. В зависимости от типа операции в SNMP-сообщении, содержимое тела может отличаться:
    - Для операции GET (получение) или SET (установка) тело содержит переменные привязки, которые определяют OID (идентификаторы объектов) и соответствующие значения, запрашиваемые или устанавливаемые для управляемых объектов.
    - Для операции GETNEXT (получение следующего) тело содержит OID переменной привязки, указывающий на начальный объект, от которого требуется получить следующий доступный объект.
    - Для операции GETBULK (массовое получение) тело содержит информацию о количестве переменных привязки и запрашиваемом количестве следующих объектов.

- Для операции TRAP (оповещение) тело содержит информацию об уведомлении, которое агент SNMP отправляет SNMP NMS для информирования о событии или состоянии в сети.

Структура SNMP-сообщения определяет формат и содержимое данных, которые передаются между SNMP NMS и агентами SNMP для обмена информацией о состоянии устройств и выполнения управляющих операций.

Безопасность и SNMP.

Безопасность является важным аспектом протокола SNMP (Simple Network Management Protocol), поскольку SNMP-сообщения могут содержать конфиденциальную информацию об устройствах и сетевой инфраструктуре. Ниже рассмотрены основные меры безопасности, применяемые в SNMP:

1. Аутентификация: SNMPv3 включает механизмы аутентификации для проверки подлинности участников взаимодействия. Он использует расширение Security Parameters в заголовке сообщения, чтобы передавать информацию о подлинности, такую как пароль или ключ. Это обеспечивает идентификацию и проверку подлинности SNMP NMS и агентов SNMP.
2. Шифрование: SNMPv3 поддерживает шифрование данных с использованием протокола шифрования данных (Data Encryption Standard, DES) или более современных алгоритмов шифрования, таких как Advanced Encryption Standard (AES). Шифрование защищает конфиденциальность передаваемой информации и предотвращает несанкционированный доступ к данным.
3. Управление доступом: SNMPv3 включает модель управления доступом на основе просмотров (View-based Access Control Model, VACM). Он позволяет администратору настроить права доступа для каждого пользователя или группы пользователей на основе OID (идентификатора объекта), давая возможность определить, какие объекты можно просматривать или изменять.
4. Защита от атак: Протокол SNMP может быть уязвим к различным видам атак, таким как подделка сообщений, отказ в обслуживании (DoS) или перехват информации. Для предотвращения таких атак рекомендуется использовать безопасные сетевые настройки, такие как

ограничение доступа к SNMP портам и использование механизмов защиты сети, таких как фильтры пакетов или виртуальные частные сети (VPN).

5. Мониторинг и журналирование: Для повышения безопасности SNMP необходимо активно мониторировать и журналировать SNMP-события и активности. Журналы могут быть использованы для обнаружения подозрительной активности или попыток несанкционированного доступа, а также для анализа сетевого состояния и выявления потенциальных угроз.

Обеспечение безопасности в SNMP крайне важно для защиты сети от несанкционированного доступа и утечки конфиденциальной информации. Рекомендуется использовать последнюю версию протокола SNMP (SNMPv3) с настройками безопасности, а также применять соответствующие сетевые и системные меры безопасности для защиты SNMP-инфраструктуры.

Лабораторная работа №2: Параллельный (мультипроцессный) сервер

Server.c

```
#include <arpa/inet.h>
#include <netinet/in.h>
#include <pthread.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <unistd.h>

#define BUFLEN 81

void zombie(int sig) {
  int status;
  while (wait3(&status, WNOHANG, (struct rusage *)0) >= 0)
    ;
}

int main() {
  int sockMain, sockClient;
  socklen_t addrLength;
  struct sockaddr_in servAddr, clientAddr;
  char *buf = malloc(sizeof(char) * BUFLEN);

  if ((sockMain = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("ERROR: Server can't open socket for TCP");
    exit(1);
  }

  bzero((char *)&servAddr, sizeof(servAddr));
  servAddr.sin_family = AF_INET;
  servAddr.sin_addr.s_addr = htonl(INADDR_ANY);
  servAddr.sin_port = 0;

  if (bind(sockMain, (struct sockaddr *)&servAddr, sizeof(servAddr)) < 0)
{
    perror("ERROR: bind() fail");
    exit(1);
  }

  addrLength = sizeof(servAddr);
  if (getsockname(sockMain, (struct sockaddr *)&servAddr, &addrLength) <
0) {
    perror("ERROR: getsockname() fail");
    exit(1);
  }
```

```
  printf("SERVER: Port number -\t %d\n", ntohs(servAddr.sin_port));

  listen(sockMain, 5);

  signal(SIGCHLD, zombie);
  while (1) {
    if (sockClient = accept(sockMain, 0, 0) < 0) {
      perror("ERROR: accept() failed");
      exit(3);
    }
    int child = fork();
    if (child < 0) {
      perror("ERROR: fork() failed");
      break;
    }
    if (child == 0) {
      close(sockMain);
      while (1) {
        int bytes_read = recv(sockClient, buf, 1024, 0);
        if (bytes_read <= 0)
          break;
        buf[bytes_read] = '\0';
        printf("SERVER: Message: %s\n", buf);
        printf("SERVER: Sending...\n");
        send(sockClient, buf, bytes_read, 0);
      }

      close(sockClient);
      exit(0);
    }

    close(sockClient);
  }
  close(sockMain);
  return 0;
}
```

Client.c

```
#include <errno.h>
#include <netdb.h>
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
```

```
  int sock, bytes_read;
  struct sockaddr_in servAddr, clientAddr;
  struct hostent *hp, *gethostbyname();
  socklen_t length;
  char *buf = malloc(sizeof(char) * 81);

  if (argc < 6) {
    printf("INSERT: IP PORT NumberOfMessages Delay Message\n");
    exit(1);
  }

  if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    perror("ERROR: Can't get a socket\n");
    exit(1);
  }

  bzero((char *)&servAddr, sizeof(servAddr));
  servAddr.sin_family = AF_INET;
  hp = gethostbyname(argv[1]);
  bcopy(hp->h_addr, &servAddr.sin_addr, hp->h_length);
  servAddr.sin_port = htons(atoi(argv[2]));

  if (connect(sock, (struct sockaddr *)&servAddr, sizeof(servAddr)) < 0)
{
    perror("ERROR: connect() failed");
    exit(1);
  }
  for (int i = 0; i < atoi(argv[3]); i++) {
    printf("CLIENT: Sending...\n");
    if (send(sock, argv[5], sizeof(argv[5]), 0) < 0) {
      perror("ERROR: send() failed");
      exit(1);
    }
    recv(sock, buf, 1024, 0);
    printf("CLIENT: Message: %s\n", buf);

    sleep(atoi(argv[4]));
  }
  close(sock);
  return 0;
}
```