МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Бакалаврська кваліфікаційна робота

на тему: Розробка системи моніторингу та аналізу мережевого трафіку

Виконав студент 4 курсу групи К-41
спеціальність 122«Комп'ютерні науки»
Панченко Іван Олександрович

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант _____

Рецензент к.т. н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2020

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ДБЖ - джерело безперебійного живлення

ЛВС - локальна обчислювальна мережа

ОС - операційна система

СКС - структуровані кабельні системи

CSMA / CD -Carrier Sense Multiple Access / Collision Detection (метод множинного доступу з упізнанням несучої і виявленням колізій)

ISO -International Standards Organization (міжнародна організація стандартів)

LAN - Local Area Network (локальна обчислювальна мережа)

MAN - Metropolitan Area Network (регіональні комп'ютерні мережі)

OSI -Open Systems Interconnection (взаємодія відкритих систем)

SCS -Structured Cabling System (структуровані кабельні системи)

STP - Shielded Twisted Pair (кручена пара)

TCP - Transmission Control Protocol (протокол контролю передачі)

UTP -Unshielded Twisted Pair (неекранована кручена пара)

VPN - Virtual Private Network (віртуальна приватна мережа)

WAN -Wide Area Network (територіальні комп'ютерні мережі)

Wi-Fi - Wireless Fidelity (протокол побудови бездротової мережі)

WWW - World Wide Web (глобальна комп'ютерна мережа)

ВСТУП

В роботі комп'ютерної мережі та мережевого стеку вузлів іноді виникають проблеми, причини яких важко виявити загальновідомими утилітами збору статистики (такими наприклад, як netstat) і стандартними додатками на основі протоколу ICMP (ping, traceroute/tracert тощо). У подібних випадках для діагностики неполадок часто доводиться використовувати більш специфічні засоби – програми-сніфери, які дозволяють відобразити (прослухати) мережевий трафік і проаналізувати його на рівні одиниць передачі окремих протоколів.

Аналізатори мережевих протоколів або сніфери є виключно корисними інструментами для дослідження поведінки мережевих вузлів і виявлення неполадок в роботі мережі. Зрозуміло, сніффер може бути благом в руках системного адміністратора або інженера з інформаційної безпеки, так і знаряддям злочину в руках комп'ютерного зловмисника.

Метою кваліфікаційної роботи є розробка програми-аналізатора мережевого трафіку, яка буде захоплювати і візуалізувати мережевий трафік (Ethernet, WiFi), що передається по протоколу IP.

Для досягнення поставленої мети в роботі вирішуються наступні завдання:

- дослідження особливостей реалізації і структури стека протоколів TCP/IP;
- аналіз принципів побудови і функціонування аналізаторів мережевого трафіку;
- порівняльний аналіз сучасних програм-аналізаторів мережевого трафіку;
- обґрунтування вибору мережевого інтерфейсу прикладного програмування;
- разработка программы – сниффера IP-пакетов.

Сніффер, що розробляється, буде використаний надалі виключно для дослідження поведінки мережевих вузлів і виявлення неполадок у роботі власної мережі. Не передбачається використовувати його з метою прослуховування чужого трафіку і організації хакерської атаки на мережу.

Структура дипломного проекту складається з вступу, чотирьох розділів, висновків, переліку посилань на 16 найменувань, додатків. Повний обсяг проекту складає 61 сторінка, містить 12 рисунки і 3 таблиці.

1 ОГЛЯД МЕРЕЖЕВОЇ МОДЕЛІ TCP/IP

1.1 Відповідність стека TCP/IP моделі OSI

Так як архітектура TCP/IP була розроблена задовго до становлення в 1980х роках еталонної моделі взаємодії відкритих систем (OSI), не дивно, що конструктивна модель TCP /IP дещо відрізняється від еталонної моделі OSI (рис.1.1) [1][1)].

Рисунок 1.1 – Відповідність стека TCP/IP моделі OSI

Прикладний рівень стека TCP/IP відповідає трьом верхнім рівням моделі OSI: прикладного, подання і сеансовому. Він об'єднує сервіси, що надаються системою користувальницьким додаткам.

Транспортний рівень стека TCP/IP може надавати вышележащему рівнем два види сервісу:

[1)] [1] Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов В.Г. Олифер, Н.А. Олифер. 4-е изд. СПб: Питер, 2010. 944.

– гарантовану доставку забезпечує протокол управління передачею (Transmission Control Protocol, TCP);

– доставку по можливості, або з максимальними зусиллями, забезпечує протокол користувацких дейтаграм (User Datagram Protocol, UDP).

Для того щоб забезпечити надійну доставку даних, протокол TCP передбачає встановлення логічного з'єднання, що дозволяє йому нумерувати пакети, підтверджувати їх прийом квитанціями, у разі втрати організовувати повторні передачі, розпізнавати і знищувати дублікати, доставляти прикладного рівня пакети в тому порядку, в якому вони були відправлені. Завдяки цьому протоколу об'єкти на хості-відправника та хості-одержувача можуть підтримувати обмін даними в дуплексному режимі. TCP дає можливість без помилок доставити сформований на одному з комп'ютерів потік байтів на будь-який інший комп'ютер, що входить в складову мережу.

Другий протокол цього рівня, UDP, є найпростішим дейтаграммным протоколом, який використовується тоді, коли завдання надійного обміну даними або взагалі не ставиться, або вирішується засобами більш високого рівня — прикладним рівнем або користувацькими додатками.

Мережевий рівень, званий також рівнем Інтернету, є стрижнем усієї архітектури TCP/IP. Саме цей рівень, функції якого відповідають мережевому рівню моделі OSI, забезпечує переміщення пакетів у межах складовою мережі, утвореної об'єднанням декількох підмереж. Протоколи мережного рівня підтримують інтерфейс з вищерозміщених транспортним рівнем, отримуючи від нього запити на передачу даних по складовою мережі, а також з нижчого рівнем мережевих інтерфейсів [2][1)] .

Основним протоколом мережевого рівня є міжмережевий протокол (Internet Protocol, IP). У його завдання входить просування пакету між

[1)] [2] Мережеві інтерфейси.URL: https://uk.wikipedia.org/wiki/Мережевий_рівень (дата звернення 11.04.2020)

мережами – від одного маршрутизатора до іншого до тих пір, поки пакет не потрапить в мережу призначення. На відміну від протоколів прикладного та транспортного рівнів, протокол IP розгортається не тільки на хостах, але і на всіх маршрутизаторах (шлюзах). Протокол IP – це дейтаграммный протокол, який працює без встановлення з'єднань за принципом доставки з максимальними зусиллями. Такий тип мережного сервісу називають також «ненадійним».

Ідеологічним відзнакою архітектури стеку TCP/IP від багаторівневої архітектури інших стеків є інтерпретація функцій самого нижнього рівня – рівня *мережевих інтерфейсів*.

Завдання нижнього рівня стека TCP/IP – організація взаємодії з підмережами різних технологій, які входять в складену мережу. TCP/IP розглядає будь-яку підмережу, що входить в складену мережу, як засіб транспортування пакетів між двома сусідніми маршрутизаторами.

Кожен комунікаційний протокол оперує певною одиницею переданих даних. Назви цих одиниць інколи закріплюються стандартом, а частіше просто визначаються традицією (рис.1.2). У стеку TCP/IP за багато років його існування утворилася усталена термінологія в цій області.

Рисунок 1.2 – Назви протокольних одиниць даних у TCP/IP

Потоком даних, інформаційним потоком, або просто потоком, називають дані, що надходять від додатків на вхід протоколів транспортного рівня TCP і UDP. Протокол TCP «нарізає» з потоку даних сегменти. Одиницю даних протоколу UDP часто називають дейтаграммой, або датаграммой. Дейтаграма – це загальна назва для одиниць даних, якими оперують протоколи без встановлення з'єднань. До таких протоколів відноситься і протокол IP, тому його одиницю даних іноді теж називають дейтаграммой, хоча досить часто використовується інший термін – пакет.

1.2 Мережеві IP – адреси

Internet Protocol address (IP-адреса) [2][1)] є чисельної міткою, яка присвоюється кожному пристрою, підключеному до комп'ютерної мережі, що використовує протокол Інтернету для зв'язку.

1.2.1 Формат IP-адреси

У заголовку IP-пакета для зберігання IP-адрес відправника і одержувача відводяться два поля, кожне має фіксовану довжину 4 байта (32 біта). IP-адреса складається з двох логічних частин – номера мережі і номера вузла в мережі.

На рис.1.3 показано звернення до протоколу дозволу адрес (ARP).

Найбільш поширеною формою подання IP-адреси є запис у вигляді чотирьох чисел, що представляють значення кожного байта в десятковій формі і розділених крапками, наприклад:

128.10.2.30

[1)] [2] IP – адреса. URL:https://ua.wikipedia.org/wiki/IP-адреса (дата звернення 11.04.2020)

Цю ж адресу може бути представлений у двійковому форматі:

10000000 00001010 00000010 00011110

А також в шістнадцятковому форматі:

80.0 A. 02.1 D

Рисунок 1.3 – Перетворення адрес

Найпростіший варіант вирішення проблеми розбиття IP-адреси на мережеву та хостову часті полягає у використанні фіксованої межі. При цьому всі 32-бітне поле адреси заздалегідь ділиться на дві частини не обов'язково рівні, але фіксованої довжини, в одній із яких завжди буде розміщуватися номер мережі, в іншій – номер вузла. Оскільки поле, яке відводиться для зберігання номера вузла, має фіксовану довжину, всі мережі будуть мати однакову максимальну кількість вузлів.

Другий підхід заснований на використанні маски, яка дозволяє максимально гнучко встановлювати межу між номером мережі і номером вузла. При такому підході адресний простір можна використовувати для створення безлічі мереж різного розміру. Маска – це число, яке застосовується в парі з IP-адресою, причому двійковий запис маски містить безперервну послідовність одиниць в тих розрядах, які повинні в IP-адресі

інтерпретуватися як номер мережі. Межа між послідовностями одиниць і нулів в масці відповідає межі між номером мережі і номер вузла в IP-адресі.

І, нарешті, спосіб, заснований на класах адрес. Цей спосіб являє собою компроміс по відношенню до двох попередніх: розміри мереж хоча і не можуть бути довільними, як при використанні масок, але і не повинні бути однаковими, як при встановленні фіксованих меж. Вводиться п'ять класів адрес: A, B, C, D, E. Три з них – A, B і C – призначені для адресації мереж, а два – D і E – мають спеціальне призначення. Для кожного класу мережевих адрес визначено власне положення кордону між номером мережі і номер вузла.

1.2.2 Класи IP-адрес

Ознакою, на підставі якого IP-адреса відносять до того чи іншого класу, є значення декількох перших бітів адреси. Табл. 1.1 ілюструє структуру IP-адрес різних класів.

Таблиця 1.1 – Характеристики адрес різних класів

Клас	Найменший номер мережі	Найбільший номер мережі	Максимальне число вузлів в мережі	Маска
A	1.0.0.0	126.255.255.255	2^{24}-2	255.0. 0.0
B	128.0.0.0	191.255.255.255	2^{16}-2	255. 255.0.0
C	192.0.0.0	223.255.255.255	2^{8}-2	255. 255.255.0
D	224.0.0.0	239.255.255.255	Групова адреса	
E	240.0.0.0	247.255.255.255	зарезервовано	

До класу A належать адресу, в якому старший біт має значення 0. До класу B відносяться всі адреси, старші два біти яких мають значення 10. До

класу С відносяться всі адреси, старші три біта яких мають значення 110. Якщо адреса починається з послідовності 1110, то вона є адресою класу D і позначає особливий груповий адреса (multicast address).

1.2.3 Особливі IP-адреси

В TCP/IP існують обмеження при призначенні IP-адрес, а саме номери мереж і номери вузлів не можуть складатися з одних двійкових нулів або одиниць. Звідси випливає, що максимальна кількість вузлів, наведене в табл. 1.1 для мереж кожного класу, повинно бути зменшено на 2. Наприклад, в адресах класу С під номер вузла відводиться 8 біт, які дозволяють задати 256 номерів: від 0 до 255. Проте в дійсності максимальне число вузлів в мережі класу С не може перевищувати 254, так як адреси 0 та 255 заборонені для адресації мережевих інтерфейсів. З цих же міркувань випливає, що кінцевий вузол не може мати адресу типу 98.255.255.255, оскільки номер вузла в цій адресі класу А складається з одних двійкових одиниць.

Отже, деякі IP-адреси інтерпретуються особливим чином (табл.1.2).

Таблиця 1.2 – Особливі IP-адреси

Адреси	Призначення
0.0.0.0	Обмежений адреса відправника
255.255.255.255	Обмежений широкомовна адреса
X.X.X.255	Сетевой широковещательный адрес
127.X.X.X, де $0 \le X \le 255$	Зарезервовано для програмного інтерфейсу loopback (lo)
10.0.0.0/8 172.16.0.0./12 192.168.0.0/16	Діапазони приватних (білих) IP адрес

Якщо IP-адреса скаладається тільки з двійковіх нулів, то вона називається навізначеною адресою і позначає адресу того вузла, який згенерував цей пакет.Адреса такого виду в особливих віпадках поміщається в заголовку IP-пакета в поле адреси відправника.

Якщо поле номера мережі коштують лише нулі, то за замовчуванням вважається, що вузол призначення належить тій же самій мережі, що і вузол, який відправив пакет. Така адреса може бути використаний лише у якості адреси відправника.

Якщо всі двійкові розряди IP-адреси дорівнюють 1, то пакет з такою адресою призначення повинен розсилатися всім вузлам, що знаходяться в тій самій мережі, що і джерело цього пакету. Такий адресу називається обмеженим широкомовним (limited broadcast). Обмеженість у даному випадку означає, що пакет не вийде за межі цієї мережі не при яких умовах.

Якщо у полі адреси призначення в розрядах, відповідних номером вузла, стоять лише одиниці, то пакет, який має таку адресу, розсилається всім вузлам мережі, номер якої вказано в адресі призначення. Наприклад, пакет з адресою 192.190.21.255 буде направлений всім вузлам мережі 192.190.21.0. Такий тип адреси називається широкомовним (broadcast).

Особливий сенс має IP-адресу, перший октет якого дорівнює 127. Ця адреса є внутрішнею адресою стека протоколів комп'ютера (або маршрутизатора). Вона використовується для тестування програм, а також для організації роботи клієнтської і серверної частин додатка, встановлених на одному комп'ютері. Коли програма надсилає дані по IP-адресою 127.x.x.x, то дані не передаються в мережу, а повертаються модулями верхнього рівня того ж комп'ютера як тільки що прийняті. Маршрут переміщення даних утворює «петлю», тому ця адреса називається адресою зворотної петлі (loopback).

Вже згадувані групові адреси, що відносяться до класу D, призначені для економічного розповсюдження в Інтернеті або великої корпоративної

мережі аудіо - та відеопрограм, адресованих відразу великої аудиторії слухачів або глядачів. Якщо груповий адресу поміщений в поле адреси призначення IP-пакета, то цей пакет повинен бути доставлений відразу декільком вузлам, які утворюють групу з номером, зазначеним у поле адреси. Один і той же вузол може входити в кілька груп. У загальному випадку члени групи можуть розподілятися по різних мереж, що знаходяться один від одного на довільно великій відстані. Груповий адресу не ділиться на номери мережі і вузла і обробляється маршрутизатором особливим чином. Основне призначення групових адрес – поширення інформації за схемою «один до багатьох».

1.3 Формат IP-пакета

Поле *номера версії* займає 4 біта і ідентифікує версію протоколу IP. Зараз повсюдно використовується версія 4 (IPv4), хоча все частіше зустрічається і нова версія (IPv6) [1][1)].

Значення *довжини заголовка* IP-пакета також займає 4 біта і вимірюється в 32-бітових словах. Зазвичай заголовок має довжину до 20 байт (п'ять 32-бітних слів), але при додаванні певної службової інформації це значення може бути збільшена за рахунок додаткових байтів у полі параметрів. Найбільша довжина заголовка становить 60 байт. На рис 1.4 перераховані поля заголовка

Поле *типу сервісу* (Type of Service, ToS) має й іншу, більш сучасну назву – *байт диференційованого обслуговування*, або *DS-байт*. Цим двом назвами відповідають два варіанти інтерпретації цього поля. В обох випадках це поле служить одній меті – збереженню ознак, які відображають вимоги до

[1)] [1] Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов В.Г. Олифер, Н.А. Олифер. 4-е изд. СПб: Питер, 2010. 944.

якості обслуговування пакету. В попередньому варіанті перші три біти містять значення *пріоритету* пакета: від найнижчого – 0 до найвищого – 7. Маршрутизатори і комп'ютери можуть приймати до уваги пріоритет пакета і обробляти більш важливі пакети в першу чергу. Наступні три біти поля ToS визначають *критерій вибору маршруту*. Якщо біт D (Delay – затримка) встановлений в 1, то маршрут повинен вибиратися для мінімізації затримки доставки даного пакету, встановлений біт T (Throughput – пропускна здатність) – для максимізації пропускної здатності, а біт R (Reliability – надійність) – для максимізації надійності доставки. Два біта мають нульове значення.

Рисунок 1.4 – Структура заголовка IP-пакета

Поле *загальної довжини* займає 2 байти і характеризує загальну довжину пакета з урахуванням заголовка і поля даних. Максимальна довжина пакету обмежена розрядністю поля, що визначає цю величину, і становить 65 535 байт, проте в більшості комп'ютерів і мереж настільки великі пакети не використовуються. При передачі по мережах різного типу довжина пакету вибирається з урахуванням максимальної довжини пакету

протоколу нижнього рівня, що несе IP-пакети. Якщо це кадри Ethernet, то вибираються пакети з максимальною довжиною 1500 байт, що уміщаються в поле даних кадру Ethernet. У стандартах TCP/IP передбачається, що всі хости повинні бути готові приймати пакети довжиною аж до 576 байт (незалежно від того, чи приходять вони цілком або фрагментами)).

Идентифікатор пакету займає 2 байти і використовується для розпізнавання пакетів, що утворилися шляхом розподілу на частини (фрагментації) вихідного пакету. Всі частини (фрагменти) одного пакету повинні мати однакове значення цього поля.

Прапори займають 3 біта і містять ознаки, пов'язані з фрагментацією. Встановлений в 1 біт DF (Do not Fragment – не фрагмент) забороняє маршрутизатора фрагментувати даний пакет, а встановлений в 1 біт MF (More Fragments – більше фрагментів) говорить про те, що даний пакет є проміжним (не останнім) фрагментом. Залишився біт зарезервований.

Поле *зміщення фрагмента* займає 13 біт і задає зсув в байтах поля даних цього фрагмента відносно початку поля даних вихідного (нефрагментированного) пакета. Використовується при збірці/розбиранні фрагментів пакетів. Зміщення повинно бути кратна 8 байт.

Поле *часу життя* (Time To Live, TTL) займає один байт і використовується для завдання граничного строку, протягом якого пакет може переміщуватися по мережі. Час життя пакету вимірюється в секундах і задається джерелом. Після закінчення кожної секунди перебування на кожному з маршрутизаторів, через які проходить пакет під час свого «подорожі» по мережі, з його поточного часу життя віднімається одиниця; віднімається одиниця і в тому випадку, якщо час перебування було менше секунди. Оскільки сучасні маршрутизатори рідко обробляють пакет довше, ніж за одну секунду, то час життя можна інтерпретувати як максимальна кількість транзитних вузлів, які дозволено пройти пакету. Якщо значення поля часу життя стає нульовим до того, як пакет досягає одержувача, пакет

знищується. Таким чином, час життя є свого роду годинниковим механізмом самознищення пакета.

Поле протоколу верхнього рівня займає один байт і містить ідентифікатор, що вказує, протоколом верхнього рівня належить інформація, розміщена у поле даних пакета. Значення ідентифікаторів для різних протоколів наводяться в документі RFC 1700, доступному за адресою http://www.iana.org. Наприклад, 6 означає, що в пакеті знаходиться повідомлення протоколу TCP, 17 – протоколу UDP, 1 – протоколу ICMP.

Контрольна сума заголовка займає 2 байти (16 біт) і розраховується тільки по заголовку. Оскільки деякі поля заголовка змінюють своє значення в процесі передачі пакета по мережі (наприклад, поле часу життя), контрольна сума перевіряється і повторно розраховується на кожному маршрутизаторі і кінцевому вузлі як доповнення до суми всіх 16-бітних слів заголовку. При обчисленні значення контрольної суми самого поляконтрольной суми встановлюється в нуль. Якщо контрольна сума невірна, то пакет відкидається, як тільки виявляється помилка.

Поля *IP-адрес джерела* і *приймача* мають однакову довжину - 32 біта.

Поле параметрів є необов'язковим і використовується зазвичай тільки при налагодженні мережі. Це поле складається з декількох підполів одного з восьми визначених типів. У цих підполях можна вказувати точний маршрут, реєструвати прохідні пакетом маршрутизатори, поміщати дані системи безпеки або тимчасові позначки.

Так як число підполів в поле параметрів може бути довільним, то в кінці заголовка має бути додано кілька нульових байтів для вирівнювання заголовка пакета по 32-бітної кордоні.

Далі приведені значеня полів заголовка одного з реальних IP-пакетів, захоплених в мережі Ethernet засобами аналізатора протоколів мережевого монітора (Network Monitor, NM) компанії Microsoft.

IP: Version – 4 (0x4)

IP: Header Length = 20 (0x14)

IP: Service Type - 0 (0x0)

IP: Precedence = Routine

IP: ...0 = Normal Del ay

IP: ....0... = Normal Throughput

IP: 0.. = Normal Reliability

IP: Total Length = 54 (0x36)

IP: Identification = 31746 (0x7C02)

IP: Flags Summary = 2 (0x2)

IP: 0 = Last fragment in datagram

IP: 1. = Cannot fragment datagram

IP: Fragment Offset = 0 (0x0) bytes

IP: Time to Live = 128 (0x80)

IP: Protocol - TCP - Transmission Control

IP: Checksum = 0xEB86

IP: Source Address = 194.85.135.75

IP: Destination Address - 194.85.135.66

IP: Data: Number of data bytes remaining = 34 (0x0022)

2 ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕРЕЖЕВИХ АНАЛІЗАТОРІВ ТРАФІКУ

2.1 Призначення і принципи роботи пакетних сніфферів

Аналізатори мережевих пакетів, або сніфери, спочатку були розроблені як засіб вирішення мережевих проблем. Вони вміють перехоплювати, інтерпретувати та зберігати для подальшого аналізу пакети, що передаються по мережі. З одного боку, це дозволяє системним адміністраторам і інженерам служби технічної підтримки спостерігати за тим, як дані передаються по мережі, діагностувати і усувати виникаючі проблеми. У цьому сенсі пакетні сніфери представляють собою потужний інструмент діагностики мережевих проблем. З іншого боку, подібно багатьом іншим потужним засобам, спочатку призначався для адміністрування, з плином часу сніфери стали застосовуватися абсолютно для інших цілей. Дійсно, сніффер в руках зловмисника являє собою досить небезпечне засіб і може використовуватися для заволодіння паролями та іншою конфіденційною інформацією. Однак не варто думати, що сніфери – це якийсь магічний інструмент, за допомогою якого будь-який хакер зможе легко проглядати конфіденційну інформацію, що передається по мережі. І перш ніж довести, що небезпека, що виходить від сніффер, не настільки велика, як нерідко підносять, розглянемо більш детально принципи їх функціонування.

Сніффер [4][1)] – це програма, яка працює на рівні мережного адаптера NIC (Network Interface Card) (канальний рівень) і прихованим чином перехоплює весь трафік. Оскільки сніфери працюють на канальному рівні моделі OSI, вони не повинні грати за правилами протоколів більш високого рівня. Сніфери обходять механізми фільтрації (адреси, порти тощо), які

[1)] [4] Аналізатор трафіку.URL: https://ua.wikipedia.org/wiki/Анализатор_трафіку (дата звернення 11.04.2020)

драйвери Ethernet і стек TCP/IP використовують для інтерпретації даних. Пакетні сніфери захоплюють з дроту все, що з нього приходить. Сніфери можуть зберігати кадри в двійковому форматі і пізніше розшифровувати їх, щоб розкрити інформацію більш високого рівня, сховану усередині. На рис. 2.1 показано можливості сніфера

Рисунок 2.1 – Схема роботи сніфера

Для того щоб сніффер міг перехоплювати всі пакети, що проходять через мережевий адаптер, драйвер мережевого адаптера повинен підтримувати режим функціонування promiscuous mode (безладний режим). Саме в цьому режимі роботи мережевого адаптера сніффер здатний перехоплювати всі пакети. Даний режим роботи мережевого адаптера

автоматично активізується при запуску сніфер або встановлюється вручну відповідними налаштуваннями сніфер.

Весь перехоплений трафік передається декодеру пакетів, який ідентифікує і розщеплює пакети за відповідними рівнями ієрархії. Залежно від можливостей конкретного сніферу представлена інформація про пакети може згодом додатково аналізуватися і відфільтруватися.

Сніффер може аналізувати тільки те, що проходить через його мережеву карту. Всередині одного сегмента мережі Ethernet всі пакети розсилаються всім машинам, з-за цього можливо перехоплювати чужу інформацію. Використання комутаторів (switch, switch-hub) і їх грамотна конфігурація вже є захистом від прослуховування. Між сегментами інформація передається через комутатори. Комутація пакетів – форма передачі, при якій дані, розбиті на окремі пакети, можуть пересилатися з вихідного пункту в пункт призначення різними маршрутами.

2.2 Обмеження використання сніфферів

Найбільшу небезпеку сніфери представляли в ті часи, коли інформація передавалася по мережі у відкритому вигляді (без шифрування), а локальні мережі будувалися на основі концентраторів (хабів). Однак ці часи безповоротно пішли, і в даний час використання сніффер для отримання доступу до конфіденційної інформації – завдання зовсім не з простих.

Справа в тому, що при побудові локальних мереж на основі концентраторів існує загальне середовище передачі даних (мережевий кабель) і всі вузли мережі обмінюються пакетами, конкуруючи за доступ до цього середовища (рис. 2.2), причому пакет, посланий одним вузлом мережі, передається на всі порти концентратора і цей пакет прослуховують всі інші

вузли мережі, але приймає його тільки той вузол, якому він адресований. На рис. 2.2 зображений обмін пакетів між вузлами.

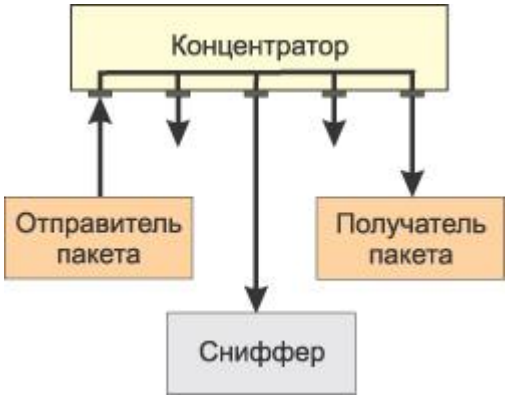


Рисунок 2.2 – При використанні концентраторів сніффер здатний перехоплювати всі пакети мережевого сегменту

При цьому якщо на одному з вузлів мережі встановлений пакетний сніффер, то він може перехоплювати всі мережеві пакети, що належать до даного сегмента мережі (мережі, утвореної концентратором).

Комутатори є більш інтелектуальними пристроями, чим широкомовні концентратори, і ізолюють мережевий трафік. Комутатор знає адреси пристроїв, підключених до кожного порту, і передає пакети між потрібними портами. Це дозволяє розвантажити інші порти, не передаючи на них кожен пакет, як це робить концентратор. Таким чином, посланий певним вузлом мережі пакет передається тільки на той порт комутатора, до якого підключений одержувач пакету, а всі інші вузли мережі не мають можливості виявити цей пакет.

На рис. 2.3 показано, що при використанні комутаторів сніффер здатний перехоплювати тільки вхідні і вихідні пакети одного вузла мережі.

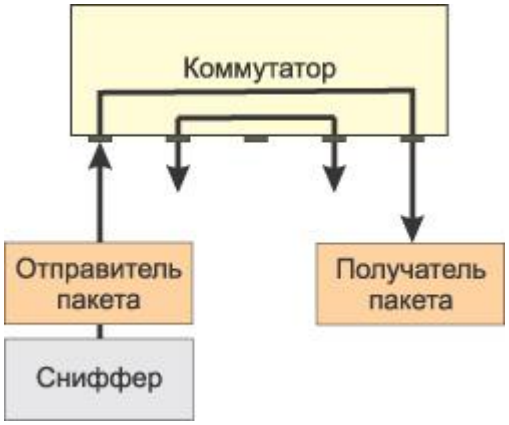


Рисунок 2.3 – Робота сніфера на основі комутатора

Тому якщо мережа побудована на основі комутатора, то сніффер, встановлений на одному з комп'ютерів мережі, здатний перехоплювати тільки ті пакети, якими обмінюється даний комп'ютер з іншими вузлами мережі. У результаті, щоб мати можливість перехоплювати пакети, якими комп'ютер або сервер, що зацікавили зловмисника, обмінюється з іншими вузлами мережі, необхідно встановити сніффер саме на цьому комп'ютері (сервері), що насправді не так-то просто. Правда, слід мати на увазі, що деякі пакетні сніфери запускаються з командного рядка і можуть не мати графічного інтерфейсу. Такі сніфери, в принципі, можна встановлювати і запускати віддалено і непомітно для користувача.

Крім того, необхідно також мати на увазі, що, хоча комутатори ізолюють мережевий трафік, всі керовані комутатори мають функцію перенаправлення або віддзеркалення портів. Тобто порт комутатора можна налаштувати таким чином, щоб на нього дублювалися всі пакети, що приходять на інші порти комутатора. Якщо в цьому випадку до такого порту підключений комп'ютер з пакетним снiффером, то він може перехоплювати всі пакети, якими обмінюються комп'ютери в даному мережевому сегменті. Однак, як правило, можливість конфігурування комутатора доступна тільки адміністратору. Це, звичайно, не означає, що він не може бути

зловмисником, але в адміністратора існує безліч інших способів контролювати всіх користувачів локальної мережі, і навряд чи він буде стежити настільки витонченим способом.

Інша причина, по якій сніфери перестали бути настільки небезпечними, як раніше, полягає в тому, що в даний час найбільш важливі дані передаються в зашифрованому вигляді. Відкриті, незашифровані служби швидко зникають з Інтернету. Наприклад, при відвідуванні web-сайтів все частіше використовується протокол SSL (Secure Sockets Layer); замість відкритого FTP використовується SFTP (Secure FTP), а для інших служб, які не застосовують шифрування за замовчуванням, все частіше використовуються віртуальні приватні мережі (VPN).

Отже, ті, хто турбується про можливості коректного застосування пакетних сніффер, повинні мати на увазі наступне. По-перше, щоб представляти серйозну загрозу для мережі, сніфери повинні знаходитися всередині самої мережі. По-друге, сьогоднішні стандарти шифрування надзвичайно ускладнюють процес перехоплення конфіденційної інформації. Тому в даний час пакетні сніфери поступово втрачають свою актуальність в якості інструментів хакерів, але в той же час залишаються дієвим і потужним засобом для діагностування мереж. Більш того, сніфери можуть з успіхом використовуватися не тільки для діагностики та локалізації мережевих проблем, але і для аудиту мережевої безпеки. Зокрема, застосування пакетних аналізаторів дозволяє виявити несанкціонований трафік, виявити та ідентифікувати несанкціоноване програмне забезпечення, ідентифікувати використовувані протоколи для видалення їх з мережі, здійснювати генерацію трафіку для випробування на вторгнення (penetration test) з метою перевірки системи захисту, працювати з системами виявлення вторгнень (Intrusion Detection System, IDS) [5][1)] .

[1)] [5] Система виявлення вторгнень.URL: https://ua.wikipedia.org/wiki/Система_виявлення_вторгнень (дата звернення 11.04.2020)

2.3 Огляд програмних пакетних сніфферов

Всі програмні сніфери можна умовно розділити на дві категорії: сніфери, що підтримують запуск з командного рядка, і сніфери, що мають графічний інтерфейс. При цьому зазначимо, що існують сніфери, які поєднують в собі обидві ці можливості. Крім того, сніфери відрізняються один від одного протоколами, які вони підтримують, глибиною аналізу перехоплених пакетів, можливостями з налаштування фільтрів, а також можливістю сумісності з іншими програмами.

Зазвичай вікно будь-якого сніффере з графічним інтерфейсом складається з трьох областей. У першій з них відображаються підсумкові дані перехоплених пакетів. Зазвичай в цій області відображається мінімум полів, а саме: час перехоплення пакета; IP-адреси відправника і одержувача пакета; MAC-адреси відправника і одержувача пакета, вихідні та цільові адреси портів; тип протоколу (мережний, транспортний або прикладного рівня); деяка сумарна інформація про перехоплені дані. У другій області виводиться статистична інформація про окремо вибраний пакет, і, нарешті, в третій області пакет представлений в шістнадцятковому вигляді або в символьній формі – ASCII.

Практично всі пакетні сніфери дозволяють проводити аналіз декодованих пакетів (саме тому пакетні сніфери також називають пакетними аналізаторами, або протокольними аналізаторами). Сніффер розподіляє перехоплені пакети за рівнями і протоколам. Деякі аналізатори пакетів здатні розпізнавати протокол і відображати перехоплену інформацію. Цей тип інформації, як правило, відображається у другій області вікна сніферу. Наприклад, будь-сніффер здатний розпізнавати протокол TCP, а просунуті сніфери вміють визначати, яким додатком породжений даний трафік. Більшість аналізаторів протоколів розпізнають понад 500 різних протоколів і вміють описувати і декодувати їх по іменах. Чим більше інформації може

декодувати і представити на екрані сніффер, тим менше доведеться декодувати вручну.

Одна з проблем, з якою можуть стикатися аналізатори пакетів, – неможливість коректної ідентифікації протоколу, що використовує порт, відмінний від порту за замовчуванням. Наприклад, з метою підвищення безпеки деякі відомі програми можуть налаштовуватися на застосування портів, відмінних від портів за замовчуванням. Так, замість традиційного порти 80, зарезервованого для web-сервера, сервер можна примусово переналаштувати на порт 8088 або на будь-який інший. Деякі аналізатори пакетів в подібній ситуації не здатні коректно визначити протокол і відображають лише інформацію про протокол нижнього рівня (TCP або UDP).

Існують програмні сніфери, до яких в якості плагінів або вбудованих модулів додаються аналітичні програмні модулі, що дозволяють створювати звіти з корисною аналітичною інформацією про перехоплений трафік.

Інша характерна риса більшості програмних аналізаторів пакетів – можливість налаштування фільтрів до і після захоплення трафіку. Фільтри виділяють із загального трафіку певні пакети по заданому критерію, що дозволяє при аналізі трафіку позбавитися від зайвої інформації.

Далі розглянемо можливості декількох доступних для скачування сніфферів, які орієнтовані на використання з платформами Windows.

2.3.1 Пакетний сніфер Wireshark

Wireshark — програма-аналізатор трафіку для комп'ютерних мереж Ethernet і деяких інших [6][1)] . Має графічний користувальницький інтерфейс.

[1)] [6] Wireshark. URL: https://en.wikipedia.org/wiki/Wireshark (дата звернення 11.04.2020)

Спочатку проект називався Ethereal, але, з-за проблем з торговою маркою, в червні 2006 року проект був перейменований в Wireshark.

Wireshark дозволяє збирати дані з безперервних потоків мережевого трафіку і конвертувати ці числа в зручні для інтерпретації графіки і таблиці, які точно показують, ким і для чого використовується корпоративна мережа. Wireshark не тільки ідентифікує, які користувачі, додатки та протоколи споживають основну частку смуги пропускання, але й фіксує IP-адреси найбільш активних у чатах співробітників, аналізує Cisco NetFlow, Juniper J-Flow, IPFIX, sFlow, Huawei NetStream та інших потокові дані.

Утиліта WinPcap (www.winpcap.org) представляє собою стандартний інструмент, за допомогою якого Windows-додатка можуть безпосередньо отримувати доступ до мережного адаптера (NIC-рівня) і перехоплювати мережеві пакети. Крім того, драйвер WinPcap має додаткові функціональні можливості, що полягають у фільтрації пакетів, збір мережевої статистики та підтримки можливості віддаленого перехоплення пакетів.

До складу утиліти WinPcap входить драйвер, що забезпечує взаємодію з NIC-рівнем, і бібліотека, яка відповідає за взаємодію з інтерфейсом API.

Wireshark – один з небагатьох сніффер, який підтримує як графічний інтерфейс, так і запуск з командного рядка, що зручно при складанні сценаріїв або активізації функцій перехоплення пакетів у разі виникнення в мережі певних подій.

Функціональні можливості пакету Wireshark дуже великі і виходять далеко за рамки звичайних можливостей інших пакетних сніфферів.

У пакеті Wireshark зустрічаються практично всі функції, які тільки можуть бути у аналізатора пакетів. Програма може декодувати 752 протоколу і підтримує роботу в Wi-Fi мережах.

Графічний інтерфейс пакета Wireshark цілком традиційний і містить три області: відображення перехоплених пакетів, відображення статистичної

інформації про конкретному вибраний пакет, вмісту конкретного пакета (рис. 2.4).

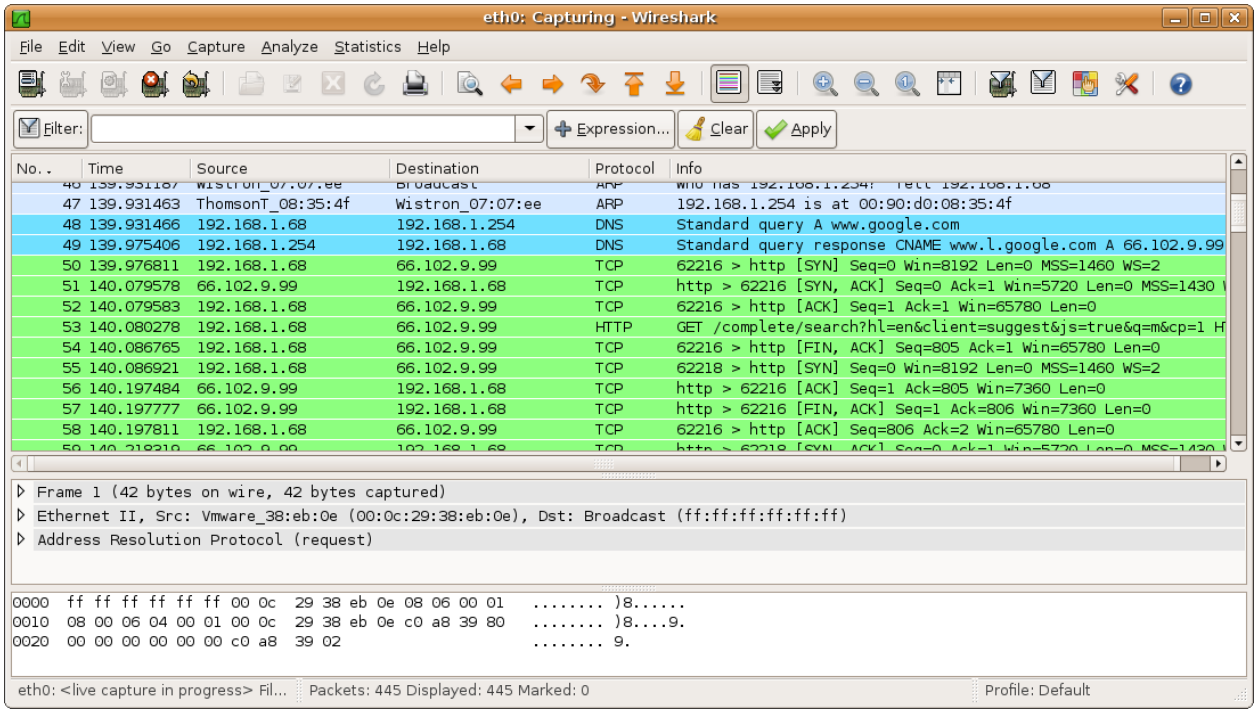


Рисунок 2.4 – Головне вікно програми Wireshark

Графічний інтерфейс програми Wireshark полегшує створення пакетних фільтрів для файлів перехоплених пакетів (фільтри відображення), так і для «живого» перехоплення (фільтри перехоплення). Після вивчення синтаксису фільтрів програми Wireshark можна створювати фільтри самостійно, присвоювати їм імена і зберігати їх для подальшого використання (рис. 2.5). Програма Wireshark має досить зручним інтерфейсом для створення фільтрів. Досить ввести необхідний критерій пошуку в рядки фільтра.

Варто відзначити, що програма Wireshark має дуже гнучкі можливості по створенню фільтрів. Програма здатна здійснювати фільтрацію практично за будь-якою характеристикою пакета.

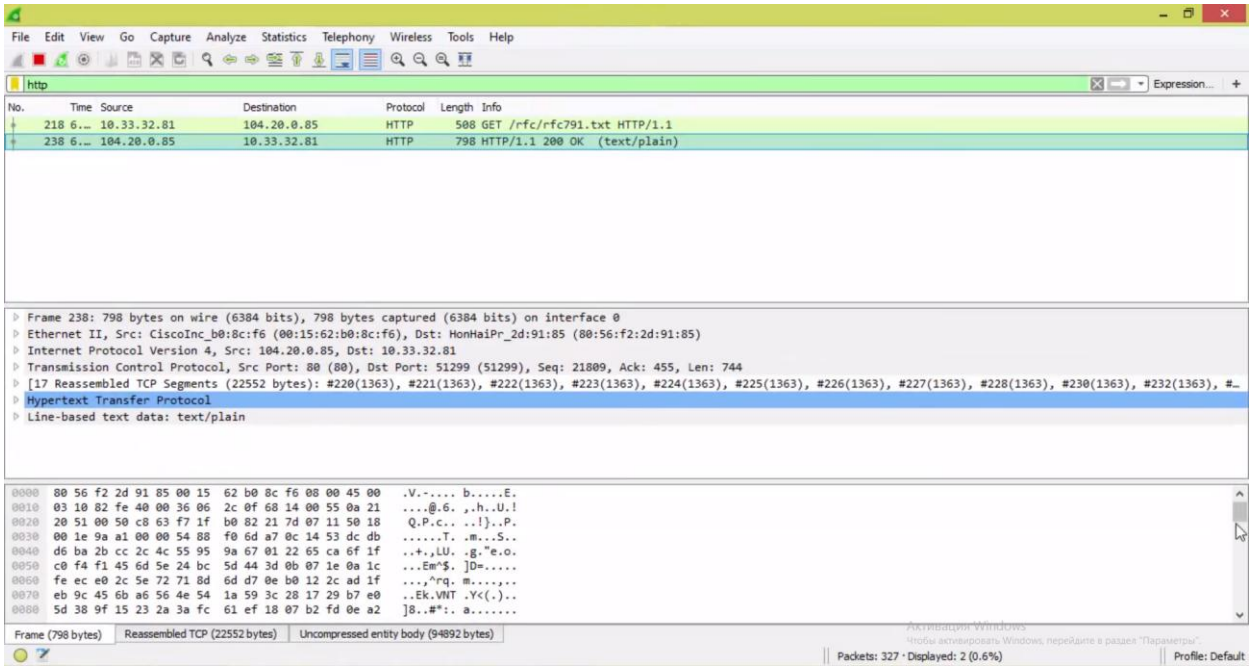


Рисунок 2.5 – Створення фільтра у програмі Wireshark

Застосування фільтрів в аналізаторі Wireshark дозволяє легко виділити із загального потоку перехопленої інформації саме ті або навіть той єдиний кадр, який потрібно. Розглянемо, наприклад, як знайти за допомогою фільтра пакет при переході користувача на веб сторінку.

Насамперед треба запустити сніффер (це можна зробити і з командного рядка, причому віддалено і непомітно для користувача) і накопичувати інформацію до тих пір, поки користувач не встановить з'єднання з Інтернетом.

Далі необхідно налаштувати фільтр, що дозволяє знайти потрібний пакет. Після досить ввести в рядок интерисующий користувача филтр і спостерігати результат.

Ще один приклад ефективного використання програми Wireshark в мирних цілях – це точна настройка розміру TCP-вікна. Щоб оптимальним чином налаштувати розмір TCP-вікна, необхідно запустити сніффер в процесі скачування файлу по мережі і потім, настроївши відповідним чином

фільтр, переглянути кількість запитів на повторну передачу пакетів, кількість пакетів-підтверджень і помилкових пакетів. Маніпулюючи з розміром TCP-вікна, можна добитися максимально можливої швидкості передачі, знизивши кількість підтверджень при гарноій якості зв'язку або зменшивши кількість запитів на повторну передачу, – при не дуже гарній якості зв'язку.

Крім прекрасних можливостей щодо створення різного роду фільтрів, програма Wireshark дозволяє виконувати всебічний аналіз трафіку, представляючи його у графічній формі або у формі статистичного звіту. Наприклад, можна виконати аналіз TCP трафіку по пропускній здатності, з часу передачі туди і назад і за номерами пакетів. Результати аналізу представляються у вигляді графіків. Так, аналіз, який використовує порядкові номери пакетів і час, дозволяє отримати уявлення про те, який обсяг даних був посланий в різні моменти часу, оскільки порядкові номери пакетів збільшуються на розмір пакету даних.

В цілому можна сказати, що пакетний аналізатор Wireshark є дуже потужним інструментальним засобом діагностики мереж.

2.3.2 Пакетний сніффер Analyzer v.2.2

Утиліта Analyzer v.2.2 компанії NetGroup – ще один невеликий за обсягом пакетний аналізатор, що розповсюджується на безкоштовній основі [7][1)] . До переваг даної утиліти можна віднести те, що вона не вимагає інсталяції на комп'ютер. Єдине, що необхідно, — наявність встановленої утиліти WinPcap, яка використовується сниффером Analyzer v.2.2. Крім того, пакетний аналізатор Analyzer v.2.2. дуже простий у використанні і може бути рекомендований для початківців користувачів. Недоліки цього аналізатора випливають з його переваг – простота в обігу не дозволяє проводити глибокий аналіз пакетів та створювати фільтри за будь характеристиці пакета

[1)] [7] Analyzer v.2.2. URL: https://compress.ru/article.aspx?id=16244#Analyzer%20v.2.2 (дата звернення 11.04.2020)

Утиліта Analyzer v.2.2 підтримує вибір інтерфейсу: мережевий адаптер або аналоговий модем. Робота в бездротових мережах не передбачена.

Графічний інтерфейс пакетного аналізатора Analyzer v.2.2 містить три традиційних вікна (рис. 2.6).

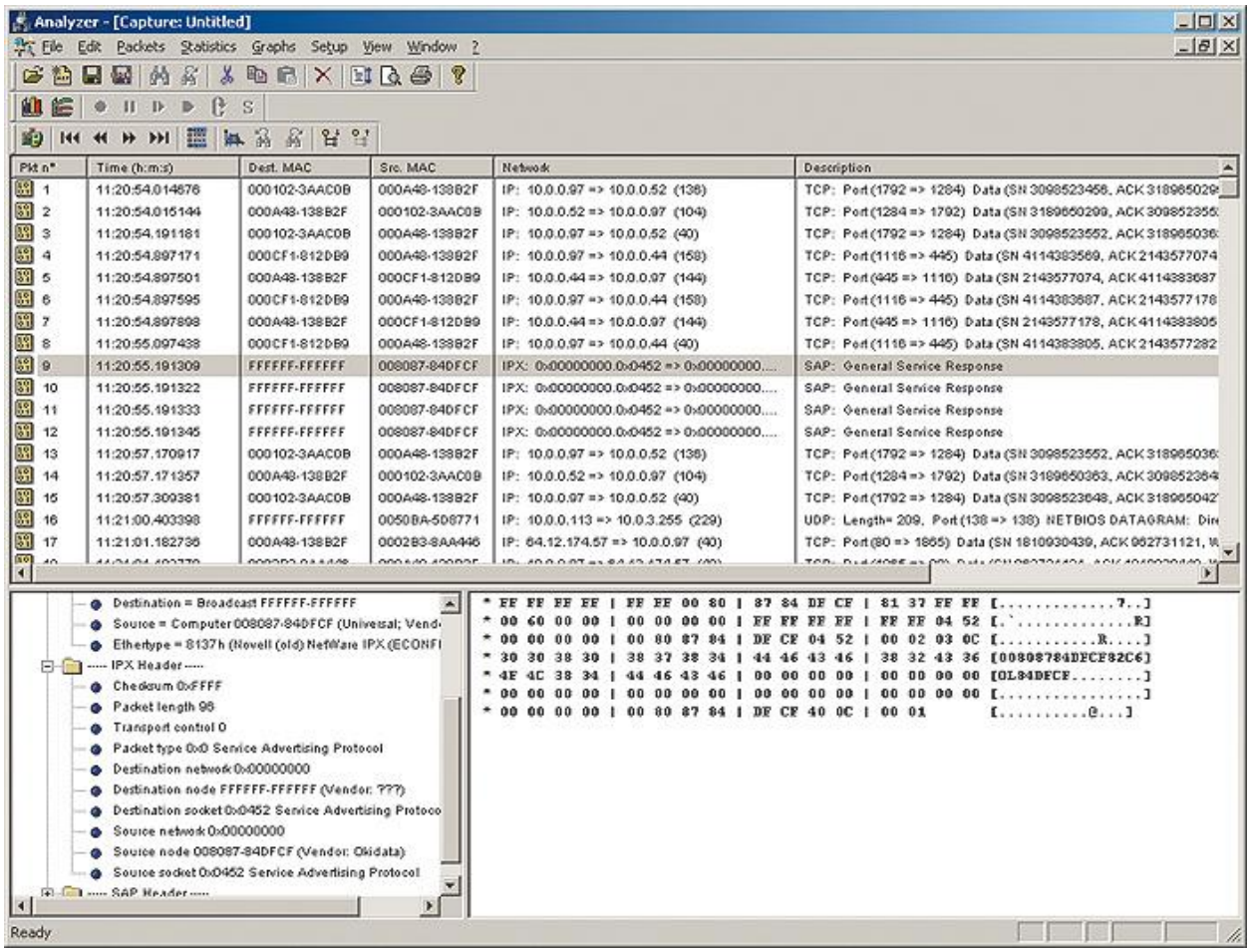


Рисунок 2.6 – Головне вікно пакетного аналізатора Analyzer v.2.2

У першому вікні відображаються перехоплені пакети з інформацією про час одержання пакета, MAC-адреси відправника і одержувача, IP-адреси відправника і одержувача, протоколі передачі, портах джерела і одержувача пакетів, розмір TCP-вікна, номер послідовності пакету і номер підтвердження цього пакету. У другому вікні виводиться розкодована

інформація про окремих полях пакету, а третє вікно відображає вміст самого пакета.

З недоліків даного пакету можна відзначити неможливість використання фільтрів після збору інформації. Єдине, що можна зробити в даному випадку, – це виділити по заданому фільтру пакети. Самі фільтри не надають гнучкого механізму (у порівнянні з пакетом Wireshark) збору необхідної інформації.

Тому ще раз підкреслимо, що програма Analyzer v.2.2 представляє інтерес тільки для початківців користувачів і як засіб ознайомлення з принципами функціонування мереж та структурами пакетів різних протоколів.

2.3.3 Пакетний сніффер CommView 5.0

На відміну від усіх розглянутих вище аналізаторів, програма CommView 5.0 поширюється на комерційній основі і для вільного скачування доступна лише її демонстраційна версія з урізаною функціональністю.

Цей пакетний аналізатор призначений для моніторингу локальної мережі та з'єднання з Інтернетом. Він здатний захоплювати пакети, що проходять через мережевий адаптер або модем, декодувати їх і представляти досить детальну інформацію в зручному для сприйняття вигляді. На відміну від більшості сніффер, програма CommView 5.0 не вимагає попередньої установки на ПК WinPcap.

Перелік підтримуваних протоколів CommView 5.0 досить великий, і в цьому плані можна розраховувати на надання достатньо докладної інформації про перехоплені пакети.

Пакетний аналізатор CommView 5.0 підтримує можливість створення фільтрів (правил), що гнучко настроюються, проте недоліком тут є те, що ці

фільтри можна застосувати до вже наявних зібраних пакетів (створювані правила поширюються тільки на захоплення пакетів).

Графічний інтерфейс програми CommView 5.0 традиційний – три вікна, в першому з яких відображаються захоплені пакети, у другому – розкодована інформація про окремому пакеті, а в третьому – вміст самого пакету (рис. 2.7).

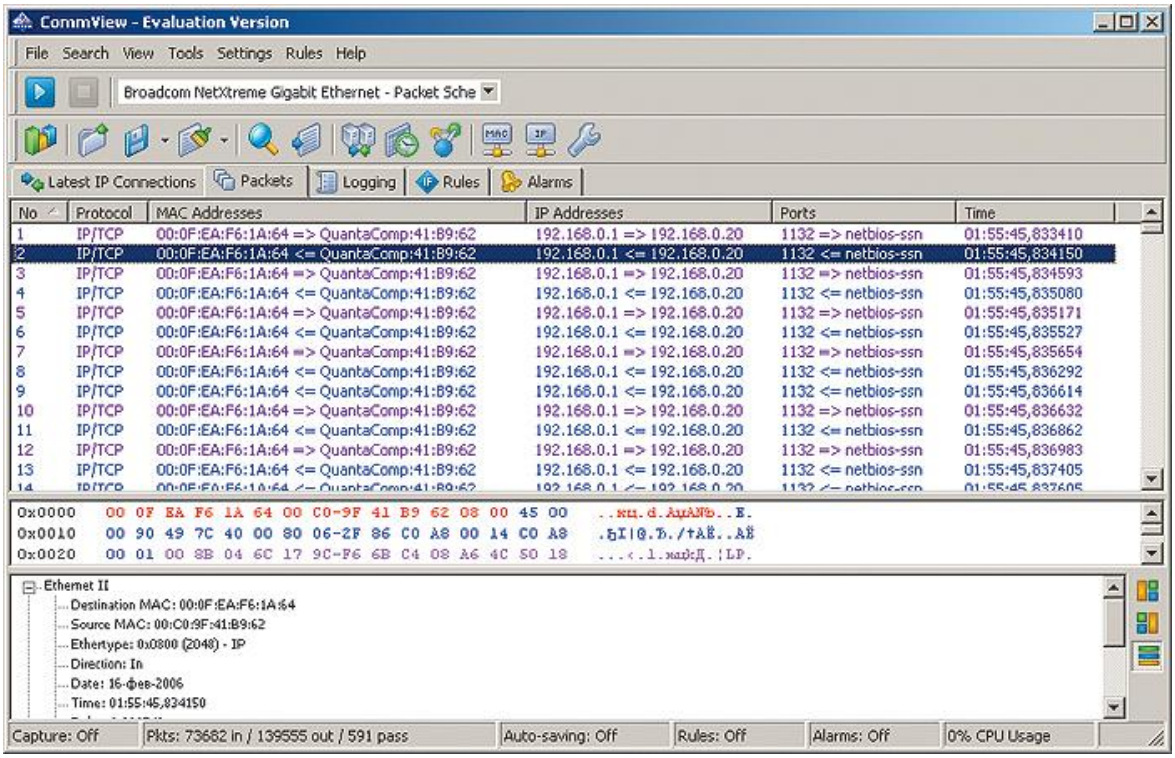


Рисунок 2.7 – Головне вікно пакетного аналізатора CommView 5.0

Інформація у вікні з захопленими пакетами досить мала. Відображаються лише тип протоколу, IP - і MAC-адреси джерела й одержувача пакета, час і порт призначення і відправлення.

Вікно з декодированной інформацією про окремому пакету значно більш інформативно і по деталізації інформації, що надається, не поступається аналізатору Ethereal (в усякому разі це стосується протоколу TCP).

До переваг аналізатора CommView 5.0 можна віднести можливість перегляду детальної статистичної інформації про сеанс перехоплення, яка подається в окремому вікні в зручному графічному вигляді (рис. 2.8), а також складання звіту в окремому файлі. Ще однією відмінністю аналізатора CommView 5.0 є можливість налаштування сигналу тривоги по визначеним подіям. Зокрема, можна задати правила, при виконанні яких буде надіслано повідомлення по електронній пошті.

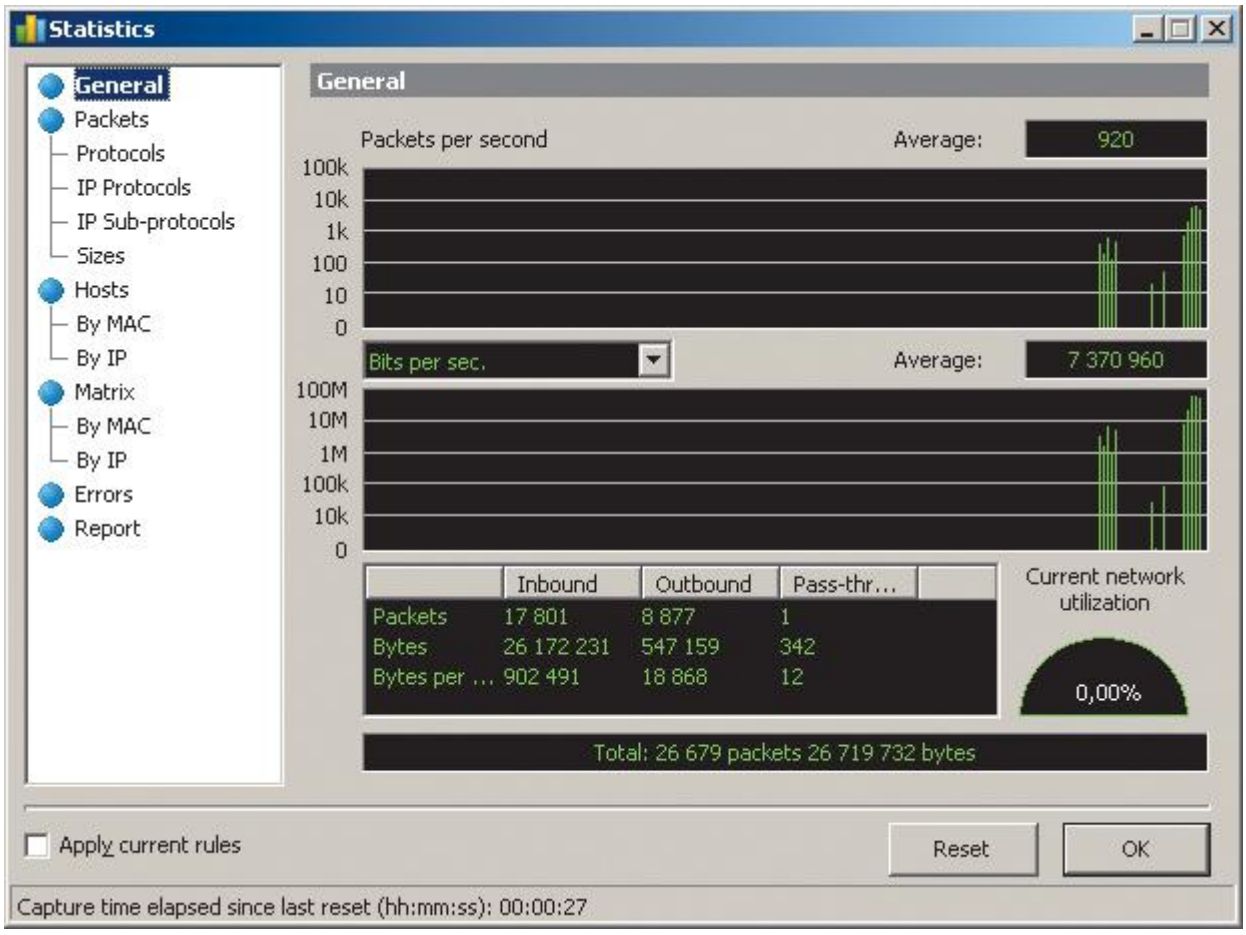


Рисунок 2.8 – Статистична інформація про сеанс перехоплення в програмі CommView 5.0

Крім того, сніффер CommView 5.0 дозволяє замінювати IP - і MAC-адреси мережевого адаптера на імена користувачів, що спрощує моніторинг

мережі. З метою діагностики мережі даний сніффер має вбудований генератор трафіку з можливістю налаштування розміру переданого пакета і швидкості генерації пакетів.

І нарешті, остання особливість сніфер CommView 5.0 – можливість створення віддаленого агента, що дозволяє здійснювати віддалений моніторинг мережі. Для реалізації даної функції на віддаленому ПК необхідно інсталювати програму Remote Agent, а використовуючи консоль CommView 5.0, можна встановлювати з'єднання з комп'ютером, на якому інстальовано утиліта Remote Agent [8][1)].

[1)] [8] Remote Agent. URL: https://www.tamos.ru/htmlhelp/commview/remote.htm (дата звернення 11.04.2020)

3 ОБҐРУНТУВАННЯ ВИБОРУ ІНТЕРФЕЙСУ ПРИКЛАДНОГО МЕРЕЖЕВОГО ПРОГРАМУВАННЯ

В сучасних Windows-орієнтованих середовищах програмування існує великий набір вбудованих компонентів і велика кількість безкоштовно розповсюджуваних класів для прикладного мережевого програмування. Так, в середовищі програмування Delphi [9][1)] існують вбудовані класи для роботи з мережею – це компоненти Delphi 6 на закладці Internet (TServerSocket і TClientCocket) і компоненти FastNet, або компоненти Indy в Delphi 7. В Java є пакет java.net. Додамо до цього також бібліотеку WINHTTP (раніше-WinInet), з допомогою якої можна програмувати клієнтські програми і MFC-класи. Однак, всі згадані компоненти, класи і бібліотеки в своїй основі спираються саме на інтерфейс сокетів. Тому в дипломній роботі вирішено використовувати саме цей інтерфейс. Крім цього, можна вказати ще на наступні причини:

— у разі, якщо для програми критичний розмір (це найбільше відноситься до серверним додаткам, які найчастіше не мають ніякої візуальної частини), то використання будь-яких VCL-компонентів вкрай небажано;

— програмування на більш низькому рівні традиційно вимагає великих витрат, але надає більш повний контроль за роботою програми, що дуже важливо для побудови гнучких і різноманітних мережевих додатків;

— якщо говорити про знання специфікації Windows Socket, то її останні нововведення (в Winsock2.0) доступні тільки на її рівні, і побудова ефективних і високопродуктивних клієнт/серверних Windows-додатків неможливо без гарного знання всіх особливостей сокетного інтерфейсу.

[1)] [9] Delphi.URL: https://ru.wikipedia.org/wiki/Delphi_(среда_разработки) (дата звернення 11.04.2020)

Два популярних інтерфейсу прикладного мережевого програмування (API – Application Programming Interface) для додатків, що використовують протоколи стека TCP/IP, називаються:

— інтерфейс сокетів (sockets API) ;

— інтерфейс транспортного рівня (TLI – Transport Layer Interface).

Останній, початково розроблений в компанії AT&T, іноді називається XTI (X/Open Transport Interface), це робота, вироблена міжнародною групою постачальників комп'ютерів X/Open, які створили свій власний набір стандартів. XTI насправді є розширенням TLI. В додатку А представлені основні програмні інтерфейси, що представляють практичний інтерес для творців мережевого програмного забезпечення [10][1)].

Відомо, що в локальних і глобальних мережах існує два принципово різних способу передачі даних. Дейтаграммный, приклад такого протоколу – протокол UDP (User Datagram Protocol), який використовується в мережах TCP/IP, або протокол IPX, який раніше був базовим в мережах Novell NetWare. Такий вид передачі даних називається «не орієнтованих на з'єднання» – (connectionless). І з встановленням логічного з'єднання. Прикладами протоколів, що використовують канали зв'язку, можуть служити протоколи TCP і SPX. Для передачі даних з використанням будь-якого з перерахованих вище протоколів додаток верхнього рівня повинна створити деякий абстрактний об'єкт, який називається сокетом і являє собою кінцеву точку з'єднання. Для створення, обслуговування і маніпулювання цим об'єктом необхідний спеціальний програмний інтерфейс.

Інтерфейс прикладної програми (Application Programm Interface – API) являє собою набір функцій, що використовується програмістами для розробки прикладних програм у визначених операційних середовищах. У вісімдесятих роках американське урядове агентство з підтримки

[1)] [10]XTI.URL: https://ru.qwe.wiki/wiki/X/Open_Transport_Interface (дата звернення 11.04.2020)

дослідницьких проектів (ARPA) фінансував реалізацію протоколів TCP/IP для UNIX-систем в Каліфорнійському університеті в р. Берклі. В ході цього проекту група дослідників-програмістів розробила інтерфейс прикладного програмування для мережевих додатків TCP/IP (TCP/IP API). Цей інтерфейс був названий інтерфейс *сокетів* TCP/IP (TCP/IP sockets) або інтерфейс сокетів Берклі (Berkeley sockets).

Крім цього інтерфейсу, додатку А зображено інтерфейс TLI/XTI, що застосовується в деяких *nix ОС, а також добре відомий інтерфейс NetBIOS, з допомогою якого досі створюються багато програми, що обслуговують локальні мережі Windows – цей інтерфейс працює за зовсім іншої ідеології, ніж інтерфейс сокетів. Як видно з додатку А, NetBIOS може взаємодіяти або безпосередньо зі своїм мережевим рівнем NetBEUI, або працювати «поверх» стека TCP/IP.

Межсетевое взаємодія в основних операційних середовищах (UNIX, LINUX, Free BSD, AIX, MACOS, Windows 9x-NT-2K або XP) базуються в даний час на ідеології сокетів. Сокети реалізують механізм взаємодії не тільки партнерів по телекомунікаціям, але і межпроцессных взаємодіях на одному комп'ютері – IPC, InterProcess Communication.

При використанні сокетів виділяються наступні етапи:

- створюється сокет;
- настроюється на заданий режим роботи;
- використовується для организації обміну;
- ліквідується.

В даний час технологія сокетів підтримує роботу практично з будь-якими стеками протоколів, суміщені (overlapped – Windows) процедури вводу/виводу, використання великої кількості сервіс-провайдерів (серверів послуг) різних виробників, можливість групування сокетів, що дозволяє реалізувати їх пріоритетне обслуговування, і багато іншого.

Спочатку інтерфейс сокетів був вбудований в ядро операційної системи UNIX. Однак багато інші операційні системи, наприклад Microsoft Windows, також реалізували інтерфейс сокетів, але у вигляді бібліотек програм (деяка модернізація концепції мережевого інтерфейсу була зроблена, починаючи з WindowsMe).

Іншими словами, для того щоб виконати мережеву функцію в операційній системі Windows, необхідно викликати відповідну функцію з бібліотеки сокетів. Незважаючи на відмінності, наявні в цих операційних системах, вихідні тексти найпростіших програм, написаних у них, трохи відрізняються один від одного.

3.1 Огляд інтерфейсу Windows Sockets

WinSock або Windows Sockets [11–13][1)] – це інтерфейс прикладного програмування (API) створений для реалізації мережевих додатків MS Windows на основі протоколу TCP/IP. Практично – це Windows-інтерфейс до стеку TCP/IP (старші версії специфікації WinSock підтримують і інші стеки мережевих протоколів). Вище згадувалося про те, що спочатку інтерфейс Берклі-сокетів був вбудований в ядро UNIX, тобто API був інтегрований в операційну систему. Інтерфейс сокетів Windows не входить в склад версій типу Windows 9x, а реалізований у вигляді однієї або декількох динамічно завантажуваних бібліотек (DLL). Для роботи використовуються бібліотеки winsock.dll (16 біт, WinSock 1.1) або wsock32.dll (підтримка BSD-sockets) або ws2_32.dll (32 біта, WinSock 2.0).

[1)] [11] WinSock.URL: https://uk.wikipedia.org/wiki/WinSock (дата звернення 11.04.2020)

[12] Windows Sockets. An Open Interface for Network Programming under Microsoft Windows Version 1.1

[13] Roberts D. Developing for the Internet with WinSock, www.itknowledge.com. Publication Date: 09/01/95.

Програмування сокетів в UNIX і Windows схоже, так як Windows Sockets розроблявся на основі сокетів Берклі, але в WinSock API до них додані функції підтримки подій Windows, а також деякі «свої», специфічні для Windows функції.

На сьогодняшній день існують три основні версії специфікації WinSock [14,15][1)]:

– WinSock 1.0 – підтримка тільки TCP/IP, 1992 г.

– WinSock 1.1 – підиримка тільки TCP/IP, 1993 г.

– WinSock 1.1 – має тільки одне сімейство адрес – AF_INET

– WinSock 2.0 – підтримка TCP/IP і додаткового програмного звбезпечення, 1994 h.

– WinSock 2.2.0 – підтримка TCP/IP і додаткового програмного забезпечення, 1996 h.

Версії WinSock2-2.2.0 об'єднуються під одним ім'ям WinSock2. Реалізація специфікації WinSock 2.0 практично не впроваджувалася в склад ОС, і тому основною версією реалізації вважається специфікація WinSock 2.2.

В додатку Б показано, як інтерфейс WinSock вписується в загальну схему розробки TCP/IP-програм для Windows. Список всіх основних динамічних бібліотек, так або інакше пов'язаних з WinSock, наведено в таблиці 3.1.

У специфікації WinSock визначається новий тип даних, SOCKET, який є дескриптором саме сокета. Дескриптор сокета в середовищі UNIX (як і інші дескриптори) є додатним цілим числом. Тому багато програм, перенесені в Windows, UNIX, припускають, що дескриптор є додатним цілим числом. У

[1)] [14] Оланд Д., Джонс Э. Программирование в сетях Microsoft Windows , ISBN: 5-318-00725-2 издательство "Питер", 2002

[15] Снейдер Й. Эффективное программирование TCP/IP. Библиотека программиста. СПб: Питер, 2001 ISBN 5-318-00453-9

загальному випадку, це не так. Дескриптор сокета WinSock не відповідає дескриптору сокета Берклі і не зобов'язаний мати позитивне значення.

Таблиця 3.1 – Опис основних динамічних бібліотек WinSock

Назва бібліотеки	Короткий опис
Winsock.dll	16-бит Winsock 1.1
Wsock32.dll	32-бит Winsock 1.1
Ws2_32.dll	Головна DLL Winsock 2.0
Mswsock.dll	Розширення Microsoft. API Mswsock.dll постачає додаткові сервіси
Ws2help.dll	Платформо-залежні доповнення, які залежать від конкретної версії ОС і не є складовою частиною WinSock
Wshtcpip.dll	Допоміжні функції для TCP
Wshnetbs.dll	Допоміжні функції для NetBT
Wshirda.dll	Допоміжні функції для IrDA
Wshatm.dll	Допоміжні функції для ATM
Wshisn.dll	Допоміжні функції для Netware
Wshisotp.dll	Допоміжні функції для OSI-транспорта
Sfmwshat.dll	Допоміжні функції для Macintosh
Nwprovau.dll	Сервіс дозволу імен для IPX
Rnr20.dll	Основний сервіс дозволу імен
Winrnr.dll	Сервіс дозволу імен для LDAP
Msafd.dll	Інтерфейс WinSock для доступу до ядра
Afd.sys	Winsock-інтерфейс ядра для доступу до TDI (Transport driver interface)

У специфікації WinSock тип SOCKET визначено як беззнаковий (unsigned). Недійсні сокети позначаються символьною константою INVALID_SOCKET. У специфікації це значення прирівнюється до виразу

(SOCKET)(~0) – від'ємне число (-1) беззнаковом вигляді. Далі специфікація обумовлює, що дійсний сокет може мати дескриптор з діапазону від нуля до INVALID_SOCKET-1. Таким чином, однина, яким можна позначити недействительный сокет – найбільше можливе беззнакове ціле, 0xFFFF .

Функції read() і write(). Вони не входять до складу WinSock, тому потребують заміни відповідно на recv() і send(). Замість функції close() необхідно використовувати closesocket(), а замість ioctl() — ioctlsocket().

3.2 Специфікація Winsock 2.0

На сьогодняшний день для створення мережевих додатків в середовищі Windows, орієнтованих на специфікацію Windows Sockets, використовується більш сучасна версія інтерфейсу прикладного програмування – WinSock 2.0 – 2.2.

Інтерфейс прикладного програмування (API) WinSock 2.0 являє собою повністю перероблену версію WinSock 1.1. Він створює основу для розробки програмних продуктів для моделі "клієнт-сервер", незалежних від транспортного середовища і забезпечує комутацію каналу зв'язку (TCP) або пакетів (UDP) для обміну даними. В результаті можна розробляти і встановлювати програми без урахування типу мережі, в якій їм доведеться працювати.

За допомогою WinSock API версії 2.0 забезпечується можливість одночасного доступу до протоколів TCP/IP та IPX/SPX, що дозволяє створювати додатки, що працюють на різних локальних мережевих операційних системах і мають доступ до TCP/IP. У специфікацію WinSock 2.0 включена реалізація нових функцій для протоколів, що підтримують Quality of Service (QoS – якість обслуговування), що забезпечує необхідну продуктивність мережі. Завдяки цьому з'явилася можливість резервувати

необхідну для конкретного додатка пропускну здатність і враховувати зміни в мережі, наприклад, зниженне або підвищенне навантаження та розрив з'єднання.

WinSock 2.0 API добре підходить для мультимедійних додатків, які вимагають гарантованої мінімальної швидкості передачі даних. WinSock 2.0 дозволяє передавати дані в таких сучасних транспортних середовищах, як ATM, ISDN і бездротові системи.

Сучасні версії WinSock API на базі специфікації WinSock 2.0:

— забезпечують протокольно-незалежний доступ до всіх ресурсів мережі, включаючи такі стандартні додатки, як DNS, SAP, X. 500 і т. д.;

— дозволяють реалізувати операції введення / виведення в режимі перекриття;

— підтримують будь-які стандартні рівні сервісу (QoS – Quality of Service);

— здійснюють об'єднання сокетов по групах відповідно до їх параметрами;

— працюють з многопротокольною широкомовною або мультікастінг-адресацією;

— передбачають спільне використання сокетів декількома процесами, умовне створення сокетів і об'єднання їх в групи;

— підтримують ідеологію Scatter and Gather (безліч буферів прийому / відправки);

— підтримують багато протокольний дозвіл імен;

— підтримують механізм даних з'єднання / роз'єднання;

— підтримують механізм умовної обробки;

— підтримують ідеологію багаторівневих провайдерів (LSP и NSP);

— надають можливості многопротокольной підтримки з інтеграцією різнопланових API.

У кваліфікаційній роботі програмний мережевий додаток, що реалізує пакетний сниффер, будемо розробляти з використанням бібліотеки Windows Sockets версії 2.2 в середовищі Visual Studio.

4 РОЗРОБКА ПРОГРАМИ АНАЛІЗАТОРА МЕРЕЖЕВОГО ТРАФІКУ

Мета кваліфікаційної роботи – розробити програму, яка буде захоплювати мережевий трафік (Ethernet, WiFi), що передається по протоколу IP, на базі бібліотеки WinSock2.2 в інтегрованому середовищі розробки Visual Studio.

Підхід, який будемо використовувати, успішно застосовується в багатьох комерційних, а також безкоштовних програмах.

У даний момент переважна більшість сучасних інформаційних мереж базуються на стеці протоколів TCP/IP. Стек протоколів TCP/IP (англ. Transmission Control Protocol/Internet Protocol) – збірна назва для мережевих протоколів різних рівнів, використовуваних у мережах. Нас буде цікавити в основному протокол IP – мережевий протокол, що використовується для негарантованої доставки даних, які розділяються на так звані пакети (більш правильний термін – дейтаграма) від одного вузла мережі до іншого.

Особливий інтерес для нас представляють IP-пакети, призначені для передачі інформації. Це досить високий рівень мережевої OSI-моделі даних, коли можна обстрагироваться від пристрою та середовища передачі даних, оперуючи лише логічним поданням. Абсолютно логічним є та обставина, що рано чи пізно повинні були з'явиться інструменти для перехоплення, контролю, обліку і аналізу мережевого трафіку. Такі засоби зазвичай називається аналізатором трафіку, пакетними анализаторыми або снифферами (від англ. to sniff — нюхати). Це – мережевий аналізатор трафіку, програма або програмно-апаратний пристрій, призначений для перехоплення і подальшого аналізу, або аналізу мережевого трафіку, призначеного для інших вузлів.

На даний момент створено досить багато програмного забезпечення для прослуховування трафіку. Найбільш відомий з них: Wireshark. В даній

роботі завдання перехоплення трафіку буде реалізована методом звичайного «прослуховування» мережевого інтерфейсу. Програма не призначена для зломом і перехоплювання чужого трафіку. Потрібно всього лише переглядати та аналізувати трафік, який проходить через хост.

Для чого це може знадобитися:

1) дивитися поточний потік трафіку через мережеве з'єднання (вхідний/вихідний/всього);

2) перенаправляти трафік для подальшого аналізу на інший хост;

3) теоретично, можна спробувати застосувати його для злому WiFi-мережі (але ми не збираємося цим займатися).

На відміну від Wireshark, який базується на бібліотеці libpcap/WinPcap, даний аналізатор не буде використовувати цей драйвер. Він буде просто пасивним спостерігачем, використовують тільки бібліотеки WinSock. Використання драйвера в даному випадку зайві. Ключовим кроком у перетворенні простого мережевого додатку в мережевий аналізатор є перемикання мережевого інтерфейсу в режим прослуховування (promiscuous mode), що і дозволить йому отримувати пакети, адресовані іншим інтерфейсам в мережі. Цей режим змушує мережеву плату приймати всі кадри, незалежно від того, кому вони адресовані в мережі.

Починаючи з Windows 2000 (NT 5.0) створити програму для прослуховування сегмента мережі стало просто, тому що її мережевий драйвер дозволяє перевести сокет в режим прийому всіх пакетів.

Включення нерозбірливого режиму:

```
long flag = 1;
SOCKET socket;
#define SIO_RCVALL 0x98000001
ioctlsocket(socket, SIO_RCVALL, &RS_Flag);
```

Наша програма оперує IP-пакетами, і використовує бібліотеку Windows Sockets версії 2.2 і «сирі» сокети (raw sockets). Для того щоб отримати прямий доступ до IP-пакету, сокет потрібно створювати наступним чином:

```
s = socket(AF_INET, SOCK_RAW, IPPROTO_IP);
```

Тут замість константи SOCK_STREAM (TCP) або SOCK_DGRAM (протокол UDP), ми використовуємо значення SOCK_RAW. Взагалі кажучи, робота з raw sockets цікава не тільки з точки зору захоплення трафіку. Фактично, ми отримуємо повний контроль за формуванням пакету. Вірніше, формуємо його вручну, що дозволяє, наприклад, послати специфічний ICMP-пакет.

Відомо, що IP-пакет складається з заголовка, службової інформації і, власне, даних. Опишемо у вигляді структури IP-заголовок.

Опис структури IP-пакета::

```
typedef struct _IPHeader
{
  unsigned char  ver_len;  // версия и длина заголовка
  unsigned char  tos;      // тип сервиса
  unsigned short length;         // длина всего пакета
  unsigned short id;       // Id
  unsigned short flgs_offset;   // флаги и смещение
  unsigned char  ttl;      // время жизни
  unsigned char  protocol; // протокол
  unsigned short xsum;           // контрольная сумма
  unsigned long  src;      // IP-адрес отправителя
  unsigned long  dest;           // IP-адрес назначения
  unsigned short *params;  // параметры (до 320 бит)
  unsigned char  *data;          // данные (до 65535 октетов)
}IPHeader;
```

Головна функція алгоритму прослуховування (захвату одного пакета) буде виглядати наступним чином:

```
IPHeader* RS_Sniff()
```

```
{
    IPHeader *hdr;
    int count = 0;
    count = recv(RS_SSocket, (char*)&RS_Buffer[0], sizeof(RS_Buffer),
0);
    if (count >= sizeof(IPHeader))
    {
        hdr = (LPIPHeader)malloc(MAX_PACKET_SIZE);
        memcpy(hdr, RS_Buffer, MAX_PACKET_SIZE);
        RS_UpdateNetStat(count, hdr);
        return hdr;
    }
    else
        return 0;
}
```

Тут отримуємо порцію даних за допомогою стандартної функції socket-функції recv, а потім копіюємо їх в структуру типу IPHeader.

І, нарешті, запускаємо нескінченний цикл захоплення пакетів. Захоплюємо всі пакети, які потраплять на наш мережевий інтерфейс:

```
while (true)
{
    IPHeader* hdr = RS_Sniff();
    // обработка IP-пакета
    if (hdr)
    {
        // печатаем заголовок в консоли
    }
}
```

Далі описуються заголовки всіх наступних протоколів, що знаходяться вище. Для цього необхідно аналізувати поле protocol в структурі IPHeader.У наступному програмному коді, відбувається розфарбовування заголовка в залежності від того, який протокол має пакет, інкапсульований в IP:

```
/*
*  Виділення пакету кольором
*/
void ColorPacket(const IPHeader *h, const u_long haddr, const u_long
whost = 0)
{
    if (h->xsum)
        SetConsoleTextColor(0x17); // если пакет не пустой
    else
        SetConsoleTextColor(0x07); // пустой пакет
     if (haddr == h->src)
         {
         SetConsoleTextColor(BACKGROUND_BLUE  |  /*BACKGROUND_INTENSITY
|*/
     FOREGROUND_RED | FOREGROUND_INTENSITY);
         }
         else if (haddr == h->dest)
         {
         SetConsoleTextColor(BACKGROUND_BLUE  |  /*BACKGROUND_INTENSITY
|*/
      FOREGROUND_GREEN | FOREGROUND_INTENSITY);    }
       if (h->protocol == PROT_ICMP || h->protocol == PROT_IGMP)
         {
         SetConsoleTextColor(0x70); // ICMP-пакет
         }
         else if(h->protocol == PROT_IP || h->protocol == 115)
         {
         SetConsoleTextColor(0x4F); // IP-in-IP-пакет, L2TP
         }
         else if(h->protocol == 53 || h->protocol == 56)
         {
         SetConsoleTextColor(0x4C); // TLS, IP with Encryption
         }
```

```
          if(whost == h->dest || whost == h->src)
          {
         SetConsoleTextColor(0x0A);
    }
}
```

Для розроблюваної програми цілком достатньо буде подивитися IP-адреси хостів, з яких і на які йде трафік, і порахувати його кількість в одиницю часу.

Для того, щоб відобразити дані IP-заголовка, необхідно реалізувати функцію перетворення заголовка (але не даних) дейтаграми в рядок. Як приклад реалізації, можна запропонувати такий варіант:

```
inline char* iph2str(IPHeader *iph)
{
    const int BUF_SIZE = 1024;
    char *r = (char*)malloc(BUF_SIZE);
    memset((void*)r, 0, BUF_SIZE);
sprintf(r, "ver=%d hlen=%d tos=%d len=%d id=%d flags=0x%X
offset=%d ttl=%dms prot=%d crc=0x%X src=%s dest=%s",
            BYTE_H(iph->ver_len),
            BYTE_L(iph->ver_len)*4,
            iph->tos,
            ntohs(iph->length),
            ntohs(iph->id),
            IP_FLAGS(ntohs(iph->flgs_offset)),
            IP_OFFSET(ntohs(iph->flgs_offset)),
            iph->ttl, iph->protocol,
            ntohs(iph->xsum), nethost2str(iph->src),
            nethost2str(iph->dest)
       );
    return r;
}
```

На підставі наведених вище базових відомостей, була написана програма-аналізатор мережевого трафіку, що реалізує локальне прослуховування IP-трафіку.

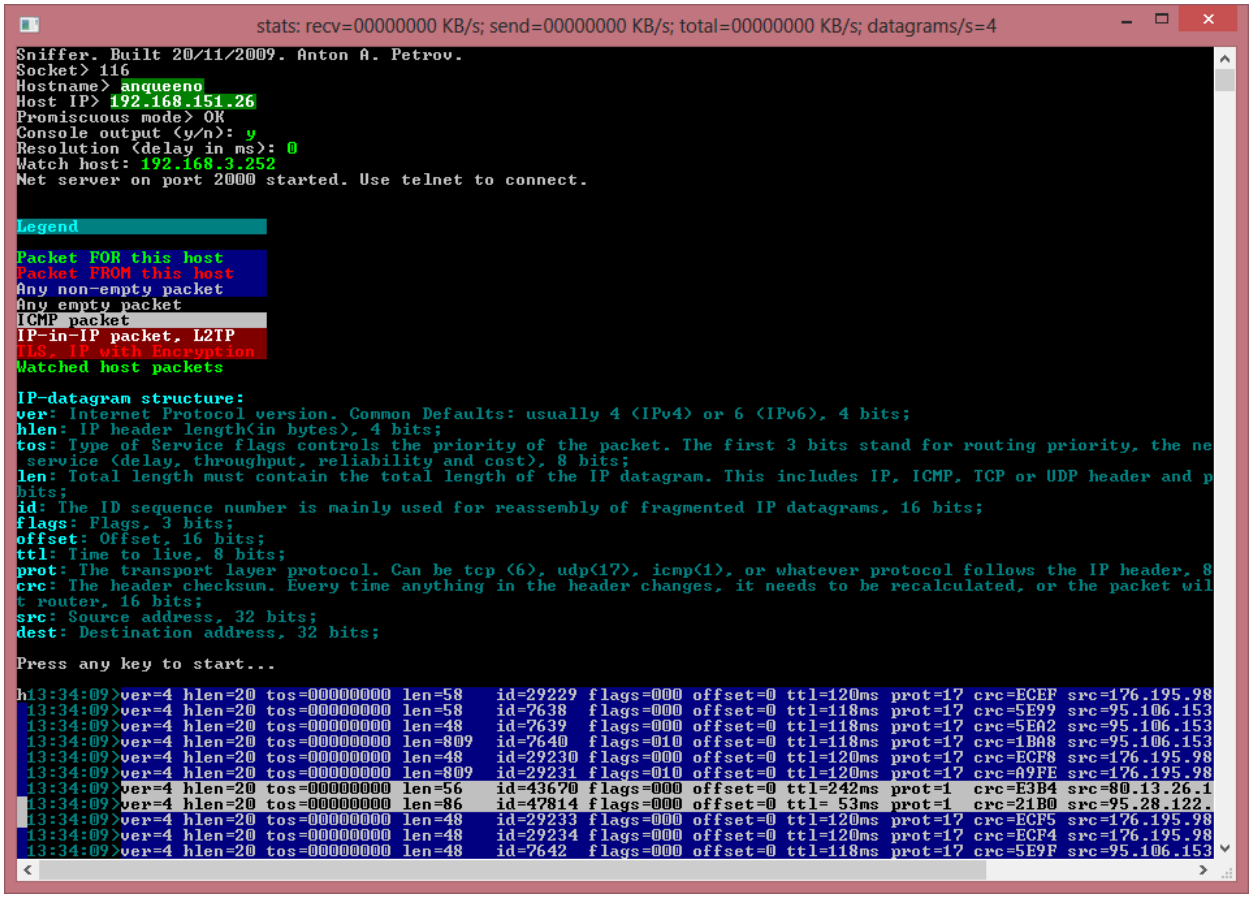


Рисунок 4.1 – Інтерфейс програма-аналізатор мережевого трафіку

Програма володіє наступною функціональністю:

– сніфер працює в режимі користувача, однак вимагає привілеї адміністраторів;

– пакети не фільтруються вілображаючиьст як є (є можливість додати настроювані фільтри);

– є можливість захоплення WiFi-трафіку;

– весь потік дейтаграм логірується в файл;

– програма працює в якості сервера на проту 2000. Можна підключитися за допомогою утиліти telnet до хосту і провести моніторинг потоків трафіку;

– кількість підключень обмежена двадцятьма.

ВИСНОВКИ

В рамках кваліфікаційної роботи було розроблено прикладне забезпечення, що реалізує функції аналізатора мережевого трафіку, який прослуховує та візуалізує мережевий трафік (Ethernet, WiFi), що передається по протоколу IP. Додаток реалізовано на базі бібліотеки WinSock 2.2 в інтегрованому середовищі розробки Visual Studio 2017.

В роботі проведено обґрунтування вибору бібліотеки WinSock 2.2 в якості бази для розробки програми. Більшість мов програмування високого рівня містять вбудовані компоненти, класи та бібліотеки для сокетного програмування, проте в роботі вибір був зупинений саме на sockets API, так як всі згадані бібліотеки в своїй основі спираються саме на інтерфейс сокетів.

В процесі розробки був проведений порівняльний аналіз подібних сучасних програм-аналізаторів мережевого трафіку. Досліджено їх структура та принципи роботи. Окремі функції програм-аналогів були взяті на озброєння при розробці власного сніферу.

Програма володіє наступною функціональністю:

– сніфер працює в режимі користувача, однак вимагає привілеї адміністратора;

– пакети не фільтруються, відображаючись як є (є можливість додати настроювані фільтри);

– є можливість захоплення WiFi-трафіку;

– весь потік дейтаграм логується в файл;

– програма працює в якості сервера на порту 2000; можна підключитися за допомогою утиліти telnet до хосту і провести моніторінг потоків трафіку;

– кількість підключень обмежена двадцятьма.

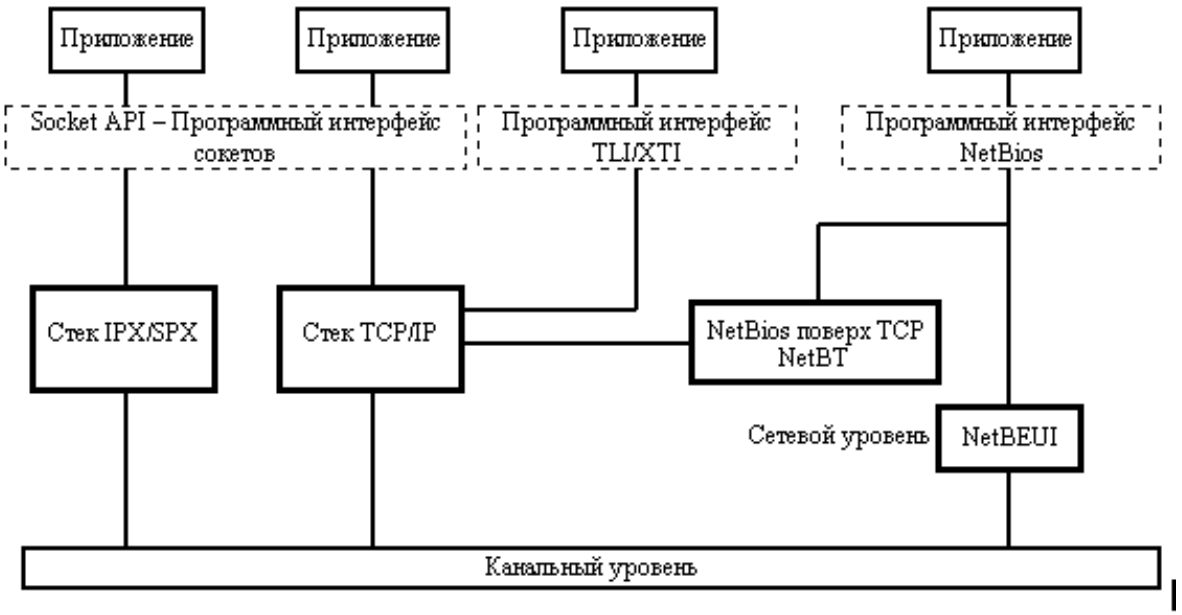
Розроблений мережевий застосунок успішно протестовано і планується до подальшого використання виключно для дослідження поведінки мережевих вузлів і виявлення неполадок у роботі власної мережі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов В.Г. Олифер, Н.А. Олифер. 4-е изд. СПб: Питер, 2010. 944.
2. Мережеві інтерфейси.URL: https://uk.wikipedia.org/wiki/ Мережевий_рівень (дата звернення 11.04.2020)
3. IP – адреса. URL:https://ua.wikipedia.org/wiki/IP-адреса (дата звернення 11.04.2020)
4. Аналізатор трафіку. URL: https://ua.wikipedia.org/wiki/ Анализатор_трафіку (дата звернення 11.04.2020)
5. Система виявлення вторгнень.URL: https://ua.wikipedia.org/wiki/ Система_виявлення_вторгнень (дата звернення 11.04.2020)
6. Wireshark. URL: https://en.wikipedia.org/wiki/Wireshark (дата звернення 11.04.2020)
7. Analyzer v.2.2. URL: https://compress.ru/ article.aspx?id=16244#Analyzer%20v.2.2 (дата звернення 11.04.2020)
8. Remote Agent. URL: https://www.tamos.ru/htmlhelp/commview/ remote.htm (дата звернення 11.04.2020)
9. Delphi.URL: https://ru.wikipedia.org/wiki/Delphi_(среда_разработки) (дата звернення 11.04.2020)
10. XTI.URL: https://ru.qwe.wiki/wiki/X/Open_Transport_Interface (дата звернення 11.04.2020)
11. WinSock.URL: https://uk.wikipedia.org/wiki/WinSock (дата звернення 11.04.2020)
12. Windows Sockets. An Open Interface for Network Programming under Microsoft Windows Version 1.1
13. Roberts D. Developing for the Internet with WinSock, www.itknowledge.com. Publication Date: 09/01/95.

14. Оланд Д., Джонс Э. Программирование в сетях Microsoft Windows , ISBN: 5-318-00725-2 издательство "Питер", 2002
15. Снейдер Й. Эффективное программирование TCP/IP. Библиотека программиста. СПб: Питер, 2001 ISBN 5-318-00453-9

ДОДАТОК А
ОСНОВНІ ПРОГРАМНІ ІНТЕРФЕЙСИ



ДОДАТОК Б

БІБЛІОТЕКИ СІМЕЙСТВА WINSOCK МЕРЕЖЕВИЙ ІЄРАРХІЇ WINDOWS