МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Фізико-технічний інститут

ОСНОВИ КОМП'ЮТЕРНИХ МЕРЕЖ

методичні вказівки до виконання комп'ютерного практикуму

версія тестової лабораторії: v1.0

Київ – 2021

Зміст

1 Тестова лабораторія

1.1 Архітектура тестової лабораторії

Робота базується на симуляції мережної архітектури на основі ізольованих середовищ у вигляді docker контейнерів. Варто зазначити, що така симуляція має багато обмежень порівняно з фізичною архітектурою. Наприклад, за необхідності створення мережного інтерфейсу відбувається підняття мосту між контейнером і хостом, причому хост стає шлюзом за замовчуванням (для повноцінної маршрутизації, ця проблема вирішується перепризначенням шлюзу за замовчуванням під час розгортання контейнеру).

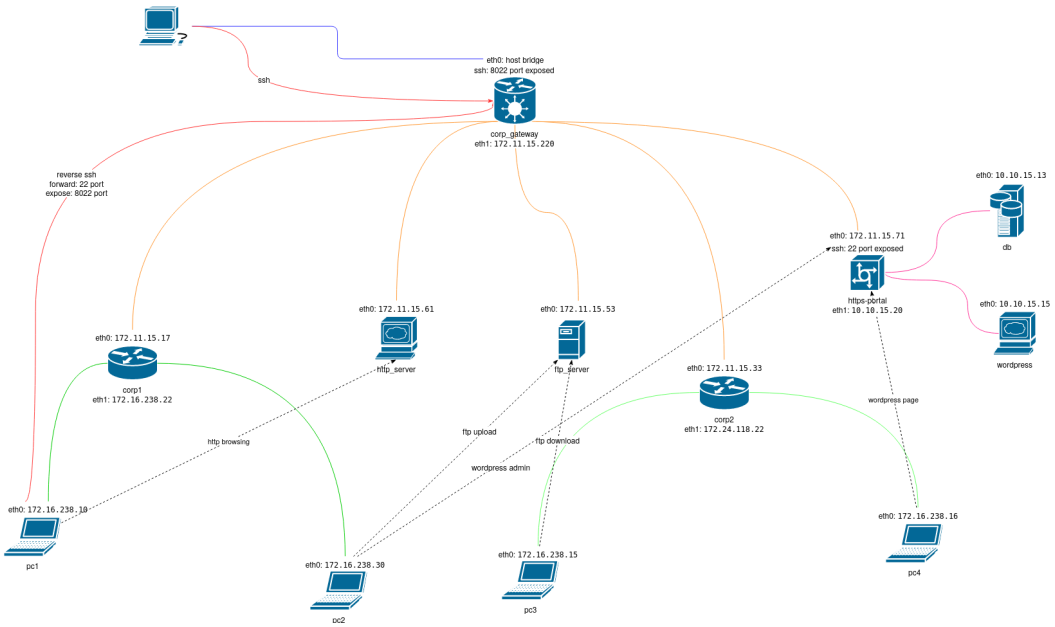
Проте така архітектура доволі адекватно симулює більшість мережних технологій, в тому числі протоколи мережного рівня та вище, технологію трансляції адрес NAT та протоколи канального рівня (з певними обмеженнями).

1.1.1 Топологія мережі

Тестова лабораторія складається з вузлів різного призначення:

- маршрутизатори локальних підмереж (`corp1` та `corp2`): призначені для логічної сегментації мережі на дві внутрішні підмережі з групами робочих станцій `pc1,pc2` та `pc3,pc4`.
- WAN шлюз `corp_gateway`, який є виходом корпоративної мережі на WAN.
- Внутрішньо корпоративні сервіси: http сервер, ftp сервер.
- Внутрішній https зворотній проксі для веб сервісу Wordpress (з окремим сервером бази даних). Цей сервіс також має внутрішній SSH доступ.
- Робочі станції `pc1`-`pc4`, які періодично (автоматично) симулюють роботу з корпоративними сервісами. При цьому, на станції `pc1` відкритий прокидання портів (port forwarding) на шлюз `corp_gateway`, на якому піднятий зворотній SSH, що дає змогу отримати SSH доступ до робочої станції ззовні корпоративної мережі.

Топологія мережі:



Варто зазначити, що завдяки вбудованому DNS в Docker контейнерах, до вузлів всередині однієї підмережі можна отримати як за IP (`172.16.238.10, 172.16.238.30`) адресою так і за хостнеймом (`client_pc1, client_pc2`)

1.1.2 Розгортання лабораторії

Лабораторія базується на технології ізольованих контейнерів Docker з засобами керування розгортанням Docker-compose, під управлінням Лінукс дистрибутиву Xubuntu. Практично весь процес розгортання та налаштування середовища автоматизований.

Для розгортання лабораторії необхідно:

1. Розгорнути віртуальну машину (Virtualbox, Vmware) з образом Лінукс дистрибутиву Xubuntu 20.04 (на інших дистрибутивах Ubuntu теж, ймовірно, можна, але не тестувалося. На інших лінукс дистрибутивах, ймовірно, можна, але можуть бути нюанси).
2. Інсталювати `git`:

```
$ sudo apt update
$ sudo apt install -y git
```

3. Клонувати репозиторій зі скриптами розгортання та конфігурацією архітектури:

```
$ git clone https://github.com/animant/ftilabs_comnetworks.git
```

4. Налаштувати середовище:

```
$ cd ftilabs_comnetworks
$ ./setup_env.sh
```

5. Розгорнути мережну архітектуру:

```
$ ./start_lab.sh
```

   Після закінчення розгортання, в терміналі виводиться версія лабораторії – її необхідно вказувати у звіті, щоб точно розуміти, на якій версії інфраструктури виконується робота.
6. Почекати 1-2 хвилини для ініціалізації сервісів.
7. Отримати доступ до робочої станції `pc1` можна через зворотний SSH:

```
$ ssh -i ./assets/pc1_access root@127.0.0.1 -p 8022
```

8. Для зупинки (згортання) лабораторії є відповідний скрипт:

```
$ ./stop_lab.sh
```

2 Вимоги до звіту

1. Звіт має містити титульну сторінку (щоб розуміти, хто його написав). Вона має включати щонайменше Ім'я виконавця, групу, дату виконання, емейл, за яким підключаєтесь до конференцій.
2. Титулка має також включати номер лабораторної роботи та версію тестової лабораторії. Це важливо, бо лабораторія буде змінюватись, іноді суттєво і потрібно буде розуміти в якому середовищі зроблені старі звіти.
3. Звіт має містити детальний опис ходу роботу із підкріпленням результів скріншотами. В т.ч., скріншоти повинні захоплювати, по можливості, ідентифікаційну інформацію – MAC-IP адреси, доменні імена, порти, ідентифікатори сесій тощо.

3 Лабораторна робота №1

3.1 Мета роботи

Метою даної лабораторної роботи є отриманя розуміння роботи базових мережних протоколів канального та мережного рівня (Ethernet, ARP, ICMP, IP), отримання розуміння роботи атаки ARP-spoofing, а також отримання навичок роботи з розповсюджении утилітами для аналізу і конфігурування у комп'ютерних мережах: `ifconfig, ip, netstat, ping, tcpdump, wireshark`.

3.2 Теоретичний матеріал

3.2.1 Збір інформації по архітектурі

1. Спершу, необхідно розгорнути лабораторію та отримати SSH доступ до робочої станції pc1:

```
$ ssh -i ./assets/pc1_access root@127.0.0.1 -p 8022
```

2. Отримайте інформацію про налаштування інтерфейсів (ip адресу, мак адресу, broadcast адресу) за допомогою команди

```
$ ifconfig -a
```

3. Канальний рівень передачі даних відбувається між сусідніми вузлами, тобто вузлами, що знаходяться в одній локальній підмережі. Оскільки ідентифікаторами сторін в цьому випадку є MAC адреси, то адреси сусідів мають зберігатися на кожному вузлі. Перевірити збережені адреси для сусідніх вузлів можна за допомогою команди arp:

```
$ arp -e
```

Отримуємо мак адресу шлюзу за замовченням 172.16.238.22 (corp1):

```
root@f2ec0f8bbc3b:~# arp -e
Address                  HWtype  HWaddress           Flags Mask            Iface
corp1.ftilabs_comnetwor  ether   02:42:ac:10:ee:16   C                     eth0
root@f2ec0f8bbc3b:~#
```

Якщо спробувати знайти маршрут до іншого вузлу цієї підмережі, то після відпрацювання протоколу ARP client_pc1 отримає та збереже в arp кеші відповідний мак:

```
$ ping -c 4 172.16.238.30
$ arp -e
```

```
root@f2ec0f8bbc3b:~# ping client_pc2
PING client_pc2 (172.16.238.30): 56 data bytes
64 bytes from 172.16.238.30: icmp_seq=0 ttl=64 time=0.179 ms
64 bytes from 172.16.238.30: icmp_seq=1 ttl=64 time=0.123 ms
64 bytes from 172.16.238.30: icmp_seq=2 ttl=64 time=0.124 ms
^C--- client_pc2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.123/0.142/0.179/0.026 ms
root@f2ec0f8bbc3b:~# arp -e
Address                  HWtype  HWaddress           Flags Mask            Iface
corp1.ftilabs_comnetwor  ether   02:42:ac:10:ee:16   C                     eth0
client_pc2.ftilabs_comn  ether   36:3b:86:f7:9d:5e   C                     eth0
root@f2ec0f8bbc3b:~#
```

Інший спосіб дізнатися мак адреси сусідніх вузлів – через команду ip:

```
$ ip neigh show
```

```
root@f2ec0f8bbc3b:~# ip neigh show
172.16.238.22 dev eth0 lladdr 02:42:ac:10:ee:16 REACHABLE
172.16.238.30 dev eth0 lladdr 36:3b:86:f7:9d:5e STALE
root@f2ec0f8bbc3b:~#
```

4. Перейдіть по SSH (пароль SSH з'єднання – P@ssw0rd) на https portal та очистіть ARP кеш:

```
$ ssh root@172.11.15.71
$ ip -s neigh flush all
```

5. Шлюз за замовчуванням – це машина в тій самій локальній підмережі через яку даний хост буде встановлювати IP шлях до машин з інших підмереж. Дізнатися про свій шлюз за замовчуванням можна за допомогою команди route -n:

```
$ route -n
```

```
root@f2ec0f8bbc3b:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         172.16.238.22   0.0.0.0         UG    0      0        0 eth0
172.16.238.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
root@f2ec0f8bbc3b:~#
```

Окрім шлюзу, в таблиці маршрутизації відображаються поточні підмережі для всіх мережних інтерфейсів.
Примітка: під час розгортання docker контейнеру, шлюзом за замовчуванням є фізичний хост. Для коректного перенаправлення трафіку, при розгортанні інфраструктури лабораторії, видаляється встановлений шлюз, а сусідній роутер призначається новим шлюзом за замовчуванням:

```
$ route add default gw 172.16.238.22
$ route del default gw 172.16.238.1
```

3.2.2 Аналіз трафіку через tcpdump і wireshark

Wireshark – це розповсюджена утиліта для аналізу мережного трафіку зі зручним GUI і доступна для різних платформ. Проте в реальних ситуаціях, часто ми маємо лише віддалений доступ через термінал чи SSH, без підтримки графічної оболонки та з обмеженими ресурсами. У таких випадках є сенс використовувати консольні утиліти tcpdump (уже встановлений на робочі станції лабораторії) та tshark (консольна версія wireshark).

1. Для початку перехоплення трафіку, на цільовій машині використовуйте команди:

```
$ tcpdump -i eth0 # прослуховувати трафік на інтерфейсі eth0
$ tcpdump -i eth0 port 80 # прослуховувати http трафік (порт 80)
$ tcpdump -i eth0 dst 10.10.13.15 # фільтрувати трафік за адресою одержувача
$ tcpdump -i eth0 -w /tmp/out.cap # зберегти трафік в /tmp/out.cap
```

   Примітка: більше інформації про утиліту можна знайти, наприклад, тут: `https://danielmiessler.com/study/tcpdump/`

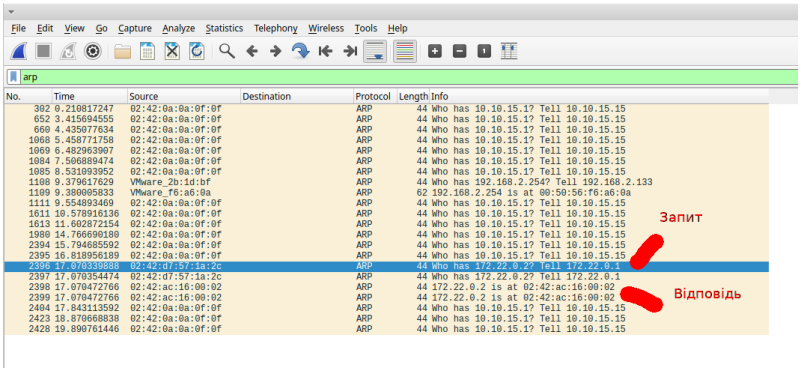
2. Для того, щоб вивантажити збережений дамп по SSH можна користуватися командою:

```
$ scp -i assets/pc1_access -P8022 root@127.0.0.1:/tmp/out.cap .
```

3. Далі, можна проаналізувати отриманий дамп більш зручним в користуванні wireshark. Документацію по wireshark можна знайти за посиланнями: `https://www.cise.ufl.edu/~helmy/F14/Wireshark_Intro_v6.0.pdf` та `https://www.wireshark.org/docs/`

3.2.3 ARP протокол, атака ARP-spoofing

Задача протоколу ARP – отримання MAC адрес вузлів, за відомою IP адресою, для встановлення каналів зв'язку всередині логічного сегменту (тому це протокол канального рівня). Нехай вузол A з адресою 10.10.15.a намагається встановити канал з вузлом B з адресою 10.10.15.b. В такому випадку, A розсилає на широкомовну адресу 10.10.15.255 ARP запит, що містить IP адресу та MAC вузла A, IP адресу вузла B. Коли запит потрапляє до вузла B, він повертає відповідь на 10.10.15.a зі своєю MAC адресою.



Протокол доволі простий, проте він є незахищеним: будь який вузол підмережі може видати себе за цільовий, повернувши ARP відповідь із своєю MAC адресою і цільовою IP. Більш того, зловмисному вузлу навіть не треба чекати запит – він сам може сформувати ARP запит зі своєю MAC та IP жертви A і відправити за адресою жертви B. Таким чином, тепер трафік від B йтиме через зловмисника і буде йому доступний. Цю атаку можно провести за допомогою утиліти arpspoof:

1. Перше, що потрібно зробити, це включити на машині атакуючого просування трафіку (ip forwarding):

```
$ echo 1 > /proc/sys/net/ipv4/ip_forward
```

2. Зазвичай, атакують одразу два вузли: жертву та шлюз, щоб і прямий і зворотній трафік проходив через атакуючого. Нехай, адреса шлюзу 10.10.10.1, а жертви – 10.10.10.3. Тоді атака виглядатиме таким чином:

```
$ arpspoof -i eth0 -t 10.10.10.1 10.10.10.3 2> /dev/null &
$ arpspoof -i eth0 -t 10.10.10.3 10.10.10.1 2> /dev/null &
```

   Після початку отруєння arp таблиці (arp poisoning), трафік між цими вузлами починає проходити через атакуючого.

3. Щоб припинити атаку, потрібно знайти PID відповідних процесів та вбити їх:

```
$ kill 50 53 # 50, 53 - arpspoof process ids
```

3.2.4 Трансляція мережних адрес (NAT)

Механізм трансляції мережних адрес використовується для створення двостороннього каналу передачі між вузлом, що знаходиться у підмережі з вузлом, що знаходиться за межами цієї підмережі. Суть такого механізму в тому, що IP адреса відправника (до якої отримувач не має маршруту) змінюється на IP адресу шлюза (який доступний отримувачу), а на зворотньому шляху IP адреси знову замінюються на оригінальні. Таким чином, отримувач бачить лише шлюз, але при цьому IP пакети передаються до вузла, що знаходиться за NAT.

3.3 Хід роботи

1. Отримайте SSH доступ до машини `client_pc1`.
2. Зберіть інформацію про налаштування мережі, адреси, маршрути, тощо.
3. Перейдіть за SSH на вузол `https-portal` (172.11.15.71). Очистіть arp кеш (та упевніться в цьому) та спробуйте пінгувати вузли `corp_gateway, corp1, corp2, http_server, ftp_server`, проаналізуйте зміни в arp кеші та дайте оцінку отриманому результату.
4. Відкрийте дві SSH сесії на `client_pc1`. В одній з них запустіть `tcpdump`, а у іншій – очистіть arp кеш та спробуйте пінгувати сусідні машини `client_pc2`-`client_pc4`. Вивантажте дамп трафіку та проаналізуйте за допомогою `wireshark`. Відфільтруйте трафік за ICMP та ARP. Опишіть функціонування цих протоколів на прикладі перехоплених пакетів.
5. Відкрийте у дампі будь який TCP пакет та проаналізуйте кожен рівень стеку TCP/IP: опишіть на прикладі перехопленого пакету, які поля містяться у заголовках кожного рівня, яке їх призначення.
6. Знову відкрийте дві сесії на `client_pc1` і запустіть `tcpdump`. Одночасно, запустіть `wireshark` для перехопленя трафіку зі сторони хосту, на мостовому інтерфейсі з контейнером `corp_gateway`. У другій SSH сесії спробуйте доступитися до зовнішнього ресурсу (наприклад, через команду `curl google.com`). Поясніть роботу механізму NAT на прикладі порівняння двох дампів трафіків – зовнішнього і внутрішнього.
7. На вузлі `client_pc1` запустіть `tcpdump` та збирайте трафік впродовж 1-2 хвилин. Проведіть arp spoofing атаку на вузли `client_pc2 client_pc3`, `ftp_server`, під час кожної з них зберіть трафік впродовж 1-2 хвилин. Порівняйте отримані дампи та надайте їм оцінку, поясніть, який результат дала атака в кожному випадку і чому саме такий.

3.4 Додаткові питання

1. Що таке маска підмережі (netmask)
2. Що таке broadcast адреса, як вона використовується
3. Що відбудеться, якщо шлюзом за замовчуванням призначити сам вузол, на якому проводяться налаштування
4. Для чого потрібен протокол ARP і як він працює
5. Які задачі виконують протоколи канального та мережного рівня
6. Як вирішується проблема у випадку, коли розмір пакету вищого рівня (наприклад, TCP) перевищує розмір тіла повідомлення (payload) нижчого (наприклад, IP)
7. Як працює механізм NAT
8. Чи можливо провести arp-spoofing атаку на вузли з іншої логічної підмережі
9. Чи можливо провести arp-spoofing атаку на вузли, що знаходяться за межами NAT атакуючого

4 Лабораторна робота №2

4.1 Мета роботи

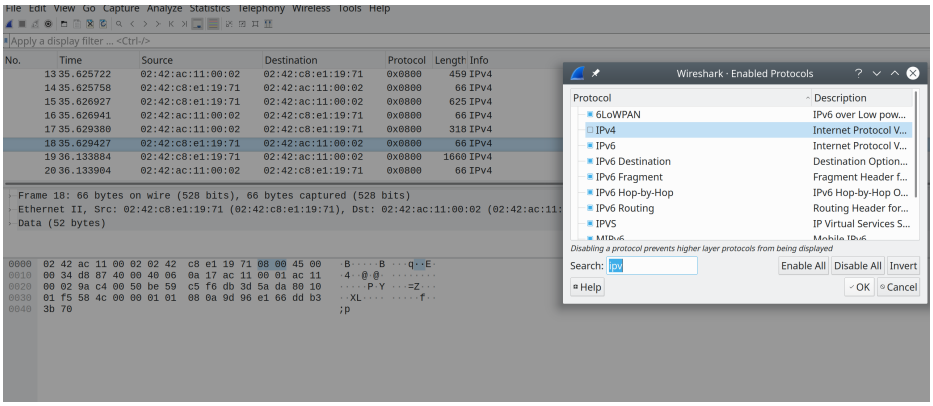
Метою даної лабораторної роботи є отриманя розуміння роботи базових мережних протоколів (Ethernet, HTTP, FTP), протоколу сеансового рівня SSL/TLS, а також поглиблення навичок роботи утилітами для аналізу мережного трафіку: `tcpdump, wireshark`.

4.2 Теоретичний матеріал

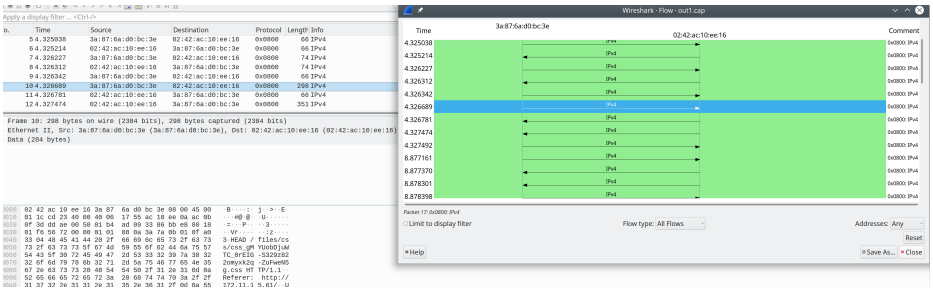
4.2.1 Протокол Ethernet

Ethernet – родина технологій передачі даних на фізичному та канальному рівні моделі OSI. В рамках роботи нас цікавить канальний рівень, а саме, Ethernet II фрейми (бо це все, що ми можемо перехопити сніфером).

Для того, щоб проаналізувати вміст Ethernet II кадру, потрібно вимкнути аналіз рівнів вище канального. Це можна зробити в меню `Analyze→Enabled protocols`, де потрібно, у більшості випадків, вимкнути IPv4 (або все, що починається на IP – про всяк випадок):



В `wireshark` є ще можливість візуалізувати шляхи пересилання повідомлень на різних рівнях між вузлами через діаграму послідовностей. Це можна зробити через меню `Statistics→Flow graph` (залишаючи необхідний рівень аналізу, наприклад, вимикаючи IPv4 протокол):

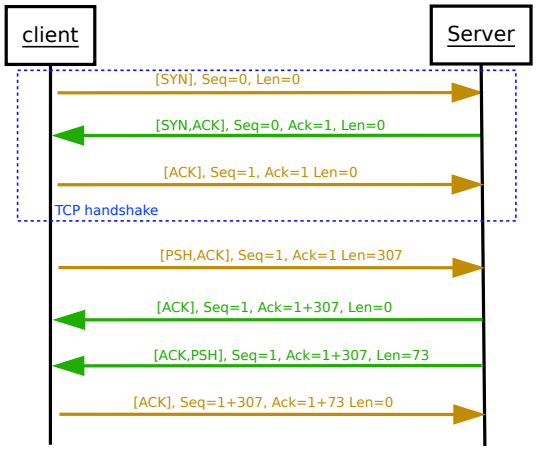


4.2.2 Протокол TCP

Розглянемо роботу протоколу TCP. Найцікавішими полями в TCP є:

- `порти призначення/відправника`
- `sequence number` – акумулює довжини переданих даних
- `acknowledge number` – акумулює довжини прийнятих даних
- `flags` – прапори типу пакету (SYN, ACK, PSH, FIN, RST та інші)
- `window size` – довжина даних в пакеті

TCP сесія починається з 3-раундового узгодження (TCP handshake), що ініціює одна із сторін пакетом типу `SYN` без тіла повідомлення. При цьому значення `sequence number` ініціалізується нулем. Інша сторона відповідає пакетом типу `SYN|ACK` та ініціалізує свій лічильник `sequence number` нулем, а лічильник `acknowledge number` – одиницею. Після цього, ініціатор посилає пакет типу `ACK` з `sequence number` = 1, `acknowledge number` = 1. На цьому протокол узгодження завершується і починається власне транспорт.



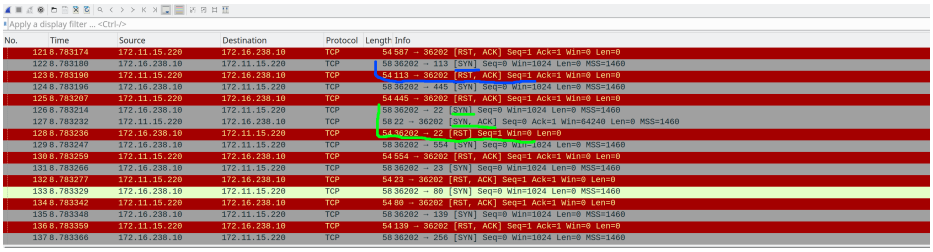
Під час транспорту, типовими пакетами є PSH|ACK при передачі даних та ACK при підтверджені. Для першого інформаційного пакету лічильники sequence і acknowledge встановлюється в 1. Далі, при передачі, в sequence додаються довжини переданих пакетів, а в acknowledge – довжини отриманих.

Оскільки до протоколу узгодження складно провести хоч якусь надійну авторизацію клієнта, то його можна використовувати для перевірки відкритих портів сервера, тобто проводити їх сканування. Є декілька типів TCP сканування, основні з яких SYN сканування та CONNECT сканування. Для SYN скану, клієнт (сканер) відправляє SYN пакет на сервер на необхідний порт. Якщо порт не прослуховується, то сервер повертає пакет RST|ACK (замість SYN|ACK) і сканер розуміє, що порт закритий. Інакше, сервер повертає SYN|ACK, але клієнт відправляє пакет RST, перериваючи протокол узгодження. Таким чином, у будь якому випадку протокол узгодження переривається і TCP сесія не створюється. Це корисно оскільки таке сканування не можна виявити з непривілейованого рівня.

Просканувати перші 1000 портів вузла (наприклад, 172.11.15.220), можна через команду:

```
$ sudo nmap -sS -p1-1000 172.11.15.220
```

В результаті отримуємо:



Бачимо, що для порту 113 сервер одразу повертає RST|ACK пакет, що означає, що цей порт не прослуховується. Але для порту 22 (SSH) повертається звичайний SYN|ACK, що означає, що порт, ймовірно, відкритий і прослуховується.

4.2.3 Протоколи прикладного рівня

Першим з протоколів прикладного рівня розглянемо протокол HTTP. Він призначений для передачі даних у клієнт-серверній архітектурі, зокрема (але не виключно), для даних, що відображаються браузерами.

Повідомнення HTTP протоколу містить заголовок, де містяться:

- метод передачі (GET, POST, OPTIONS, HEAD, PUT, DELETE, ...) – вказує серверу на спосіб обробки запиту, наприклад, GET – повернути ресурс за вказаним URL, POST – обробити дані з тіла запиту та повернути ресурс, DELETE – видалити ресурс. Поділ на методи є доволі умовним і їх обробка залежить від реалізації серверної частини. Наприклад, замість усіх методів може використовуватися лише один метод (POST), а необхідні дії вказуватися в тілі запиту чи у заголовку.

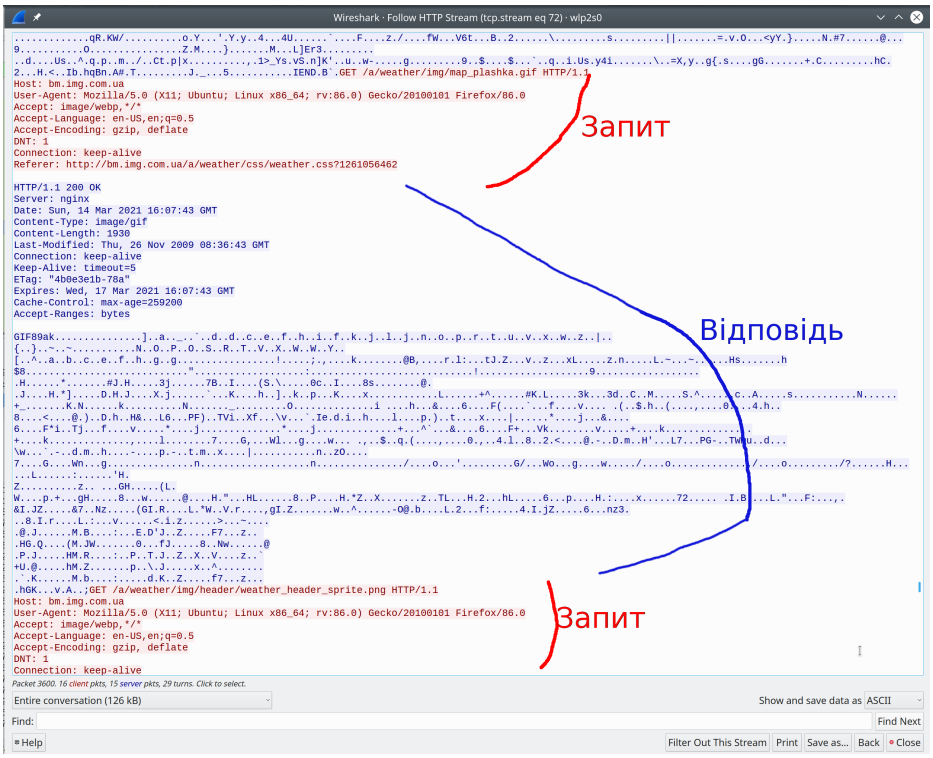
- URL – unified resource locator: уніфікована адреса ресурсу. Повна адреса файлу/скрипта, який необхідно повернути/виконати. Наприклад, `http://www-net.cs.umass.edu:80/wireshark-labs/Wireshark_HTTP_v8.0.pdf` означає повернути ресурс Wireshark_HTTP_v8.0.pdf, що знаходиться у директорії /wireshark-labs/ веб серверу www-net.cs.umass.edu, з доступом за портом 80.

- Cookie/Set-cookie – дані, що зберігаються в браузері після закриття веб-сторінки, що часто використовується як механізм керування HTTP сесією.

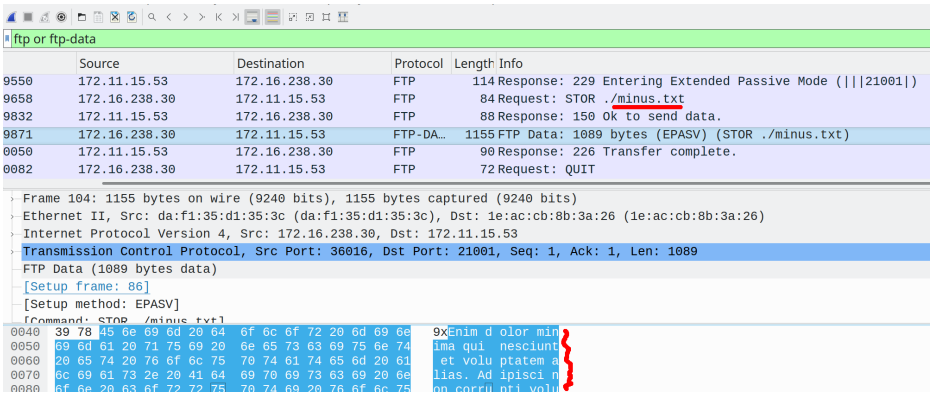
- `Інші поля` – за необхідності. Наприклад, `User-Agent` використовується для передачі серверній логіці інформації про браузер, що використовується, поле `Content-Type` потрібен для інформування про тип тіла запиту (XML, json, x-www-form-url-encoded, тощо).

- `Тіло запиту/відповіді` – дані, що передаються для виконання скриптом, або інші параметри запиту (для запиту), або дані, що повертаються сервером (для відповіді). В GET запиті зазвичай відсутнє.

У wireshark є зручна функція, яка дозволяє відображати послідовно запити і відповіді при взаємодії із сервером. Для цього, виділіть будь-який HTTP запит (з якого починатиметься відслідковування) і перейдіть в контекстному меню `Follow→HTTP stream`. Результат виглядатиме приблизно так:



Іншим прикладом протоколу прикладного рівня, який ми розглянемо є FTP (file transfer protocol), задачею якого є завантаження та вивантаження файлів. Для його аналізу необхідно фільтрувати трафік за `ftp` та `ftp-data`:



Тут ми бачимо, що завантажується файл 'minus.txt', а вміст файлу передається пакетом номер 9871.

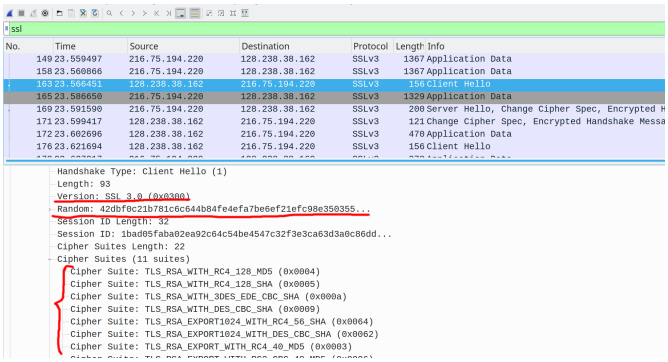
4.2.4 SSL/TLS

Протоколи транспортного та прикладного рівнів, що розглядалися вище мають один суттєвий недолік – вони не гарантують ані цілісності ані конфіденційності даних, що передаються. Одним із найрозповсюджених способів забезпечити шифрування даних та гарантування аутентичності сервера є протоколи сеансового рівня SSL(застарів)/TLS.

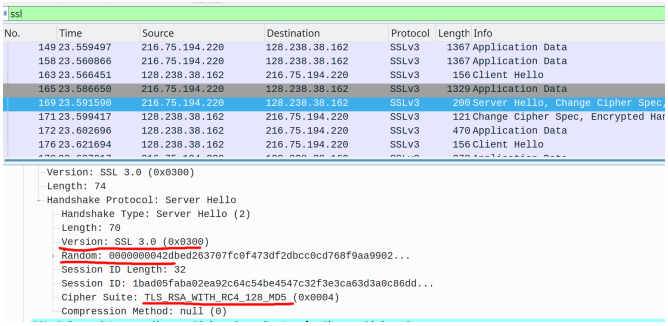
`Wireshark` дозволяє фільтрувати SSL/TLS трафік та отримувати деяку інформацію про нього.

В перехоплених пакетах виділяються такі основні:

- `Client hello` – передається на сервер список протоколів і шифрів, що підтримуються клієнтом. Також передається рандомізатор клієнта.



- `Server hello` – повертається на клієнт версію найсильнішого протокола і шифру, що підтримується і клієнтом і сервером. Також передається рандомізатор сервера та сертифікат публічного ключа.



- `Application data` – обмін зашифрованими даними.

Більше інформації про роботу TLS можна отримати тут: RFC TLS1.2, RFC TLS 1.3. На жаль, адекватної альтернативи офіційним специфікаціям немає.

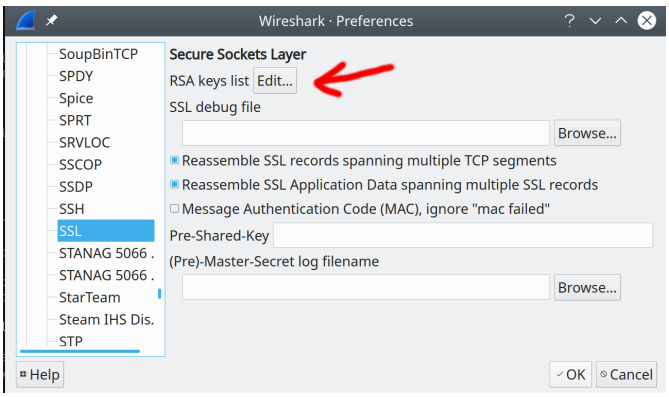
SSL/TLS сесія починається з протоколу рукостискання (handshake protocol), основною задачею якого є узгодження спільного секретного ключа між клієнтом і сервером. Є два шляхи такого узгодження:

- `Криптографічний протокол передачі ключа` – ідея підходу в тому, що кожна із сторін генерує власну частину секретного ключа і передає іншій стороні через механізм направленного шифрування, після чого, кожна із сторін обчислює спільний секрет на базі обох частин. В протоколах SSL/TLS протоколи передачі ключа базуються на алгоритмі направленого шифрування RSA.

- `Криптографічний протокол узгодження ключа` – ідея підходу в тому, що обидві сторони узгоджують спільний ключ у ході роботи протоколу Діффі-Хеллмана (на базі поля лишків або групи точок еліптичної кривої).

Важливою властивістю прикладних криптографічних протоколів є `зворотня секретність (backward secrecy)` – неможливість відновити попередні сеансові ключі після компрометації довготривалого секретного ключа. Криптографічні протоколи передачі ключа не мають такої властивості і тому вважаються недостатньо безпечними. Починаючи з версії TLS v1.3 використання протоколів без зворотньої секретності неможливе.

Але для протоколів попередніх версій, у випадку використання RSA та за умови компрометації ключа шифрування `wireshark` дозволяє розшифрувати повідомлення. Для цього потрібно зайти в `Edit→Preferences→Protocols`, вибрати протокол `SSL` та завантажити секретний ключ у відповідне поле:



4.3 Хід роботи

- За допомогою `tcpdump` перехопіть трафік з вузла `client_pc1` впродовж 1-2 хвилин. Вимкніть аналіз IPv4. Виберіть один з пакетів та опишіть на прикладі перехопленого пакету структуру Ethernet II фрейму.
- Проведіть атаку `arp spoof` на вузол `client_pc2` та перехоплюйте трафік впродовж 1-2 хвилин. Проаналізуйте довільний перехоплений пакет, визначте і опишать його шлях на канальному рівні, яким чином модифікується ethernet фрейм пд час форвардінгу.
- Ознайомтесь з CONNECT скануванням за допомогою nmap (наприклад, тут: `https://nmap.org/book/man-port-scanning-techniques.html`), як воно відбувається і в яких випадках застосовується? Проскануйте методом CONNECT вузол https_portal (172.11.15.71) та перехопіть процес мережним сніфером (tcpdump). На прикладі перехопленого скану, опишіть, як відрізняється реакція закритого і відкритого портів та яка відмінність від SYN сканування.
- Відкрийте у `wireshark` дамп з машини `client_pc1` і пофільтруйте трафік за `http`. Опишіть на прикладі перехоплених пакетів: які HTTP методи тут використовуються, які використовуються поля заголовків і для чого кожне з них призначений.
- Відкрийте у перехоплений трафік з машини `client_pc2` (отриманий в результаті arp spoof) та проаналізуйте ftp трафік. На прикладі перехоплених пакетів, опишіть процедуру аутентифікації та витягніть логін і пароль користувача, під яким передаються файли.
- Проаналізуйте SSL трафік в цьому ж дампі. Визначте та опишіть, які версії протоколів та які набори шифрів підтримуються клієнтом. Які з цих наборів мають властивість backward secrecy, а які – ні. Який набір шифрів використовується для встановлення TLS з'єднання.
- Розшифруйте HTTPS трафік використовуючи скомпрометований ключ, що знаходиться в репозиторії (`./assets/domain.key`). Проаналізуйте розшифрований трафік. Знайдіть пакети, що відповідають за аутентифікацію та витягніть логін і пароль.

4.4 Додаткові питання

1. Які ще типи (окрім 0x0800) Ethernet фреймів бувають і за що вони відповідають.
2. Який найменший розмір Ethernet II кадру і чому.
3. Чому для визначення розміру кадру не можна використовувати значення довжини у відповідному полі заголовку протокола вищого рівня (наприклад, IPv4).
4. Що таке псевдозаголовок в протоколі TCP
5. Яка різниця між методами сканування портів SYN та CONNECT.
6. Яким чином браузер розпізнає межі полів HTTP заголовку, а також межу між заголовком та тілом повідомлення.
7. Як співвідносяться між собою протоколи SSL та TLS.
8. Яка різниця між базовою HTTP аутентифікацією (basic authentication) та аутентифікацією на основі хешу (digest authentication). Які переваги і недоліки однієї та іншої.