Національний Технічний Університет України

«Київський Політехнічний Інститут Імені Ігоря Сікорського»

Факультет Інформатики Та Обчислювальної Техніки

Кафедра Обчислювальної Техніки

Лабораторна робота №6

Web сервіс

Виконала

Шпак Діана

Група: ІО-з21мп

Київ 2023

2. Завдання на роботу.

2.1. Створити закритий ключ та запит на сертифікат для доменного імені, яке відповідає варіанту завдання. Підписати запит на сертифікат, використовуючи сертифікат і закритий ключ центру сертифікації (CA), створеного в лабораторній роботі № 2.

2.2. Встановити та налаштувати Web-сервер, який реалізує протоколи HTTP та HTTPS, виконує функції балансування навантаження для двох Web-серверів та відповідає наступним вимогам:

- доменне ім’я сервера балансування навантаження відповідає варіанту завдання; - сервер балансування навантаження слухає порти для протоколів HTTP та HTTPS, номери портів відповідають варіанту завдання;

- доступ до сервера балансування навантаження мають тільки користувачі, що пройшли автентифікацію, імена яких відповідають варіанту завдання;

- метод балансування навантаження відповідає варіанту завдання;

2.3. Встановити та налаштувати 2 Web-сервери, які реалізують протокол HTTP та виконують роль серверів обробки запитів клієнтів.

2.4. Виконати аналіз протокольного обміну між клієнтом та сервером балансування навантаження під час виконання запитів по протоколам HTTP та HTTPS.

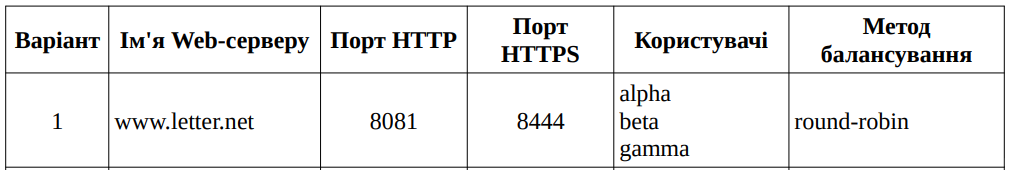
2.5. Рекомендується використовувати наступне програмне забезпечення:

- Web-сервер: nginx або apache;

- Web-клієнт: curl.

2.6. Для перевірки роботи Web-серверу та аналізу протокольного обміну рекомендується використовувати утиліти: telnet, netcat, openssl, curl.

2.7. Додати запис типу A для доменного імені Web-серверу в DNS. Додати сертифікат власного СА у список довірених на вузлі, де запускається Web-клієнт.



Перед початком конфігурації і і нсталяції балансувальника трафіку і аплікейшин серверів нам потрібно підготувати віртуальні машини, для цього напишимо вагрант скрипт, який буде мати наступний вигляд

Vagrant.configure("2") do |config|  
 config.vm.define "app1" do |app1|  
 app1.vm.box\_download\_insecure = true  
 app1.vm.box = "ubuntu/xenial64"  
  
 app1.vm.network :private\_network, ip: "172.20.1.151"  
  
 app1.vm.hostname = "app1"  
 app1.vm.post\_up\_message = "Run vm for app1"  
  
 app1.vm.provider :virtualbox do |vb|  
 vb.memory = "500"  
 end  
  
 app1.vm.provision "shell" do |s|  
 s.path = 'script1.sh'  
 end  
 end  
  
 config.vm.define "app2" do |app2|  
 app2.vm.box\_download\_insecure = true  
 app2.vm.box = "ubuntu/xenial64"  
  
 app2.vm.network :private\_network, ip: "172.20.1.152"  
  
 app2.vm.hostname = "app2"  
 app2.vm.post\_up\_message = "Run vm for app2"  
  
 app2.vm.provider :virtualbox do |vb|  
 vb.memory = "500"  
 end  
  
 app2.vm.provision "shell" do |s|  
 s.path = 'script2.sh'  
 end  
 end  
  
 config.vm.define "balancer" do |balancer|  
 balancer.vm.box\_download\_insecure = true  
 balancer.vm.box = "ubuntu/xenial64"  
  
 balancer.vm.network :private\_network, ip: "172.20.1.153"  
 balancer.vm.network "forwarded\_port", guest: 8081, host: 8081  
 balancer.vm.network "forwarded\_port", guest: 8444, host: 8444  
  
 balancer.vm.hostname = "balancer"  
 balancer.vm.post\_up\_message = "Run vm for balancer"  
  
 balancer.vm.provider :virtualbox do |vb|  
 vb.memory = "500"  
 end  
  
 balancer.vm.provision "shell" do |s|  
 s.path = 'script3.sh'  
 end  
 end  
end

2.2. Встановити та налаштувати Web-сервер, який реалізує протоколи HTTP та HTTPS, виконує функції балансування навантаження для двох Web-серверів та відповідає наступним вимогам:

- доменне ім’я сервера балансування навантаження відповідає варіанту завдання; - сервер балансування навантаження слухає порти для протоколів HTTP та HTTPS, номери портів відповідають варіанту завдання;

- доступ до сервера балансування навантаження мають тільки користувачі, що пройшли автентифікацію, імена яких відповідають варіанту завдання;

- метод балансування навантаження відповідає варіанту завдання;

Переходимо до налаштування нашого серверу балансування трафіку, що виконається автоматично скриптом який запуститься після старту віртуальної машини, скрипт буде мати наступний вигляд:

#!/bin/bash  
sudo apt update  
export LC\_CTYPE=en\_US.UTF-8  
export LC\_ALL=en\_US.UTF-8  
  
# Стіорюємо самопідписаний сертифікат за допомогою openssl.  
openssl genrsa -out /tmp/app.key 2048  
openssl req -new -key /tmp/app.key -out /tmp/app.csr -subj "/C=UA/ST=Ukraine/L=Kyiv/O=KPI/OU=OT/CN=letter CA"  
openssl x509 -req -days 365 -in /tmp/app.csr -signkey /tmp/app.key -out /tmp/app.crt  
sudo chmod 644 /tmp/app.crt /tmp/app.key  
echo "self signed cert done" >> /tmp/debug.log  
  
# Встановлюємо веб-сервер nginx і бібліотеку для генерації паролів користувачів, які ми будемо викобистовувати для автентифікації по паролю в nginx.  
sudo apt install -y nginx  
sudo apt install -y apache2-utils

# Копіююємо наші ключі в каталоги конфігурації веб-сервера nginx, встановлюємо на них потрібні нам права, створюємо файл для конфігурації нашого сервера, встановлюємо потрібні нам права для файлів конфігурації і лог файлів щоб у нас був доступ до цих файлів у процесу nginx  
mkdir -p /etc/nginx/ssl  
sudo cp -f /tmp/app.key /etc/nginx/ssl/app.key  
sudo cp -f /tmp/app.crt /etc/nginx/ssl/app.crt  
sudo chmod 755 /etc/nginx/ssl && sudo chmod -R 644 /etc/nginx/ssl/\*  
sudo mv -f /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak  
sudo echo "nginx installed" >> /tmp/debug.log  
sudo cp /etc/nginx/nginx.conf.bak /etc/nginx/nginx.conf

sudo chmod 777 /etc/nginx/nginx.conf  
sudo chmod 777 /var/log/nginx/error.log  
sudo chmod 777 /var/log/nginx/access.log

sudo cat /dev/null > /etc/nginx/nginx.conf

# Створюємо файл паролів, який Nginx може використовувати для автентифікації користувачів. Для цього ми створимо прихований файл, що називається .htpasswdв нашому /etc/nginxкаталозі конфігурації, після того як ми створили файл і назначили йому потрібні права доступу записуємо в його потрібний список користувачів з зашифрованими паролями.

sudo touch /etc/nginx/.htpasswd  
sudo chmod 777 /etc/nginx/.htpasswd  
cat > /etc/nginx/.htpasswd <<'EOF'   
alpha:$apr1$riJKaHtA$HauUETCMFlbp4PQNj.Pbk/  
beta:$apr1$wWqOPZV8$seEXZdQruZudJhcr8noBu1  
gamma:$apr1$Jzqa4xq4$qeWt99f/A5kH2QWXpPiNw1  
EOF  
  
# Переходимо до налаштування нашого серверу  
cat > /etc/nginx/nginx.conf <<'EOF'

# Користувач, від імені якого працює nginx, це www-data;  
user www-data;

# кількість процесів сервера, значення виставляється рівним кількості ядер процесора, auto – сервер визначить автоматично;  
worker\_processes auto;

# шлях до журналу реєстрації помилок;  
error\_log /var/log/nginx/error.log;

# файл, у якому зберігається ідентифікатор запущеного головного процесу (PID);  
pid /run/nginx.pid;

# файл або файли конфігурації, що підключається;  
include /usr/share/nginx/modules/\*.conf;

# блок директив, що визначають роботу з мережевими з'єднаннями;  
events {  
 worker\_connections 1024;  
}

# блок директив http сервера;  
http {

# Під час налаштування NGINX вам потрібно визначити сервери для балансування навантаження або проксі. Сервери, визначені у upstream, посилаються на сервер за допомогою директив proxy\_pass, оскільки у нас два протоколи http і https то визначаємо два блоки, для кожного порта свій, оскільки в завданні балансування раунд робін і воно використовується в балансері по замовчуванню то нам не потрібно вказувати тип балансування. можна вказатияк доменне ім’я чи IP-адресу з додатковим портом або як шлях сокета домену UNIX, указаний після префікса «unix:».Якщо порт не вказано, використовується 80 порт.

upstream backend1 {  
 server 172.20.1.151:80;  
 server 172.20.1.152:80;  
 }  
  
 upstream backend2 {  
 server 172.20.1.151:443;  
 server 172.20.1.152:443;  
 }  
# Файл конфігурації NGINX має містити одну дві дерективи серверів для визначення віртуальних серверів які будуть слухати відповідні порти для http і https протоколів.

server {  
 listen 8081;  
 server\_name \_;  
# NGINX Plus може надсилати трафік до різних проксі-серверів або обслуговувати різні файли на основі URI запиту, для прикладу зробимо дві статичні сторінки з різними URI. Щоб налаштувати автентифікацію, нам потрібно визначити контекст, який потрібно обмежити. Серед інших варіантів Nginx дозволяє встановлювати обмеження на рівні сервера або всередині певного місця.У нашому прикладі ми обмежимо кожен location.

У кожному блоці розташування встановлюємо auth\_basic директиву, щоб увімкнути автентифікацію та вибрати ім’я області, яке відображатиметься користувачеві під час запиту облікових даних. Ми будемо використовувати auth\_basic\_user\_file директиву, щоб вказати Nginx на файл паролів, який ми створили:

location / {  
 proxy\_pass http://backend1;  
 auth\_basic "Restricted Content";  
 auth\_basic\_user\_file /etc/nginx/.htpasswd;  
 }  
  
 location /bar {  
 proxy\_pass http://backend1/bar;  
 auth\_basic "Restricted Content";  
 auth\_basic\_user\_file /etc/nginx/.htpasswd;  
 }  
 }

# Налаштовуємо сервер який слухає порт для HTTPS і обробляє данц запити, він відрізняється від блока для HTTP додатковими параметрами для роботи з SSL  
  
 server {  
 listen 8444 ssl http2 default\_server;  
 listen [::]:8444 ssl http2 default\_server;  
 server\_name \_;  
 ssl\_certificate "/etc/nginx/ssl/app.crt";  
 ssl\_certificate\_key "/etc/nginx/ssl/app.key";  
 ssl\_session\_cache shared:SSL:1m;  
 ssl\_session\_timeout 10m;  
 ssl\_ciphers HIGH:!aNULL:!MD5;  
 ssl\_prefer\_server\_ciphers on;  
  
 location / {  
 proxy\_pass https://backend2;  
 auth\_basic "Restricted Content";  
 auth\_basic\_user\_file /etc/nginx/.htpasswd;  
 }  
  
 location /bar {  
 proxy\_pass https://backend2/bar;  
 auth\_basic "Restricted Content";  
 auth\_basic\_user\_file /etc/nginx/.htpasswd;  
 }  
 }  
}  
EOF  
  
## Встановлюємо firewalld, в відкриваємо тільки потрібні нам порти  
sudo apt install -y firewalld  
sudo systemctl unmask firewalld  
sudo systemctl restart firewalld  
sudo firewall-cmd --zone=public --permanent --add-service=http  
sudo firewall-cmd --zone=public --permanent --add-service=https  
sudo firewall-cmd --zone=public --permanent --add-port=8081/tcp  
sudo firewall-cmd --zone=public --permanent --add-port=8444/tcp  
sudo firewall-cmd –reload

## Робимо рестарт фаєрвола і nginx для застосування всіх змін  
sudo systemctl enable firewalld  
sudo systemctl restart firewalld  
sudo systemctl restart nginx

2.3. Встановити та налаштувати 2 Web-сервери, які реалізують протокол HTTP та виконують роль серверів обробки запитів клієнтів.

Конфігурація даних серверів також буде проводитись автоматично, і скрипт який все налаштує має наступний вигляд

#!/bin/bash  
sudo apt update  
export LC\_CTYPE=en\_US.UTF-8  
export LC\_ALL=en\_US.UTF-8

# Стіорюємо самопідписаний сертифікат за допомогою openssl.  
# # Create self signed cert for HTTPS reverse proxy as Nginx  
openssl genrsa -out /tmp/app.key 2048  
openssl req -new -key /tmp/app.key -out /tmp/app.csr -subj "/C=UA/ST=Ukraine/L=Kyiv/O=KPI/OU=OT/CN=letter CA"  
openssl x509 -req -days 365 -in /tmp/app.csr -signkey /tmp/app.key -out /tmp/app.crt  
sudo chmod 644 /tmp/app.crt /tmp/app.key  
echo "self signed cert done" >> /tmp/debug.log

# Встановлюємо веб-сервер nginx  
sudo apt install -y nginx

# Копіююємо наші ключі в каталоги конфігурації веб-сервера nginx, встановлюємо на них потрібні нам права, створюємо файл для конфігурації нашого сервера, встановлюємо потрібні нам права для файлів конфігурації і лог файлів щоб у нас був доступ до цих файлів у процесу nginx  
mkdir -p /etc/nginx/ssl  
sudo cp -f /tmp/app.key /etc/nginx/ssl/app.key  
sudo cp -f /tmp/app.crt /etc/nginx/ssl/app.crt  
sudo chmod 755 /etc/nginx/ssl && sudo chmod -R 644 /etc/nginx/ssl/\*  
sudo mv -f /etc/nginx/nginx.conf /etc/nginx/nginx.conf.bak  
sudo echo "nginx installed" >> /tmp/debug.log  
sudo cp /etc/nginx/nginx.conf.bak /etc/nginx/nginx.conf  
sudo chmod 777 /etc/nginx/nginx.conf  
sudo chmod 777 /var/log/nginx/error.log  
sudo chmod 777 /var/log/nginx/access.log  
sudo cat /dev/null > /etc/nginx/nginx.conf  
  
  
# Налаштовуємо наш nginx сервер, для того щоб він прослуховував 2 порти 80 і 443, для http sі реезі протоколыв выдповыдно, ы за двома локацыями щоб повертам нас статичну сторынку  
cat > /etc/nginx/nginx.conf <<'EOF'   
user www-data;  
worker\_processes auto;  
error\_log /var/log/nginx/error.log;  
pid /run/nginx.pid;  
# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.  
include /usr/share/nginx/modules/\*.conf;  
events {  
 worker\_connections 1024;  
}

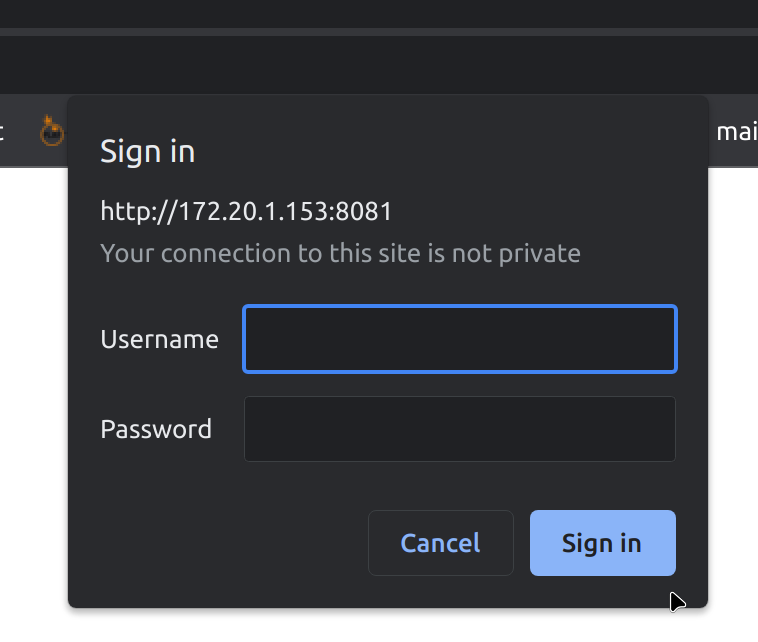
http {  
 log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '  
 '$status $body\_bytes\_sent "$http\_referer" '  
 '"$http\_user\_agent" "$http\_x\_forwarded\_for"';  
 access\_log /var/log/nginx/access.log main;  
 sendfile on;  
 tcp\_nopush on;  
 tcp\_nodelay on;  
 keepalive\_timeout 65;  
 types\_hash\_max\_size 2048;  
 include /etc/nginx/mime.types;  
 default\_type application/octet-stream;  
 include /etc/nginx/conf.d/\*.conf;  
 server   
  
 {  
 listen 80;  
 listen 443 ssl http2 default\_server;  
 listen [::]:443 ssl http2 default\_server;  
 server\_name \_;  
 root /etc/nginx/www;  
 index index.html index.htm;  
 ssl\_certificate "/etc/nginx/ssl/app.crt";  
 ssl\_certificate\_key "/etc/nginx/ssl/app.key";  
 ssl\_session\_cache shared:SSL:1m;  
 ssl\_session\_timeout 10m;  
 ssl\_ciphers HIGH:!aNULL:!MD5;  
 ssl\_prefer\_server\_ciphers on;  
 # Load configuration files for the default server block.  
 include /etc/nginx/default.d/\*.conf;  
  
 location / {  
 # it picks up default root and checks for default index.html file at the path  
 }  
  
 location /bar {  
 # it picks up default root, adds /bar to the root and looks for the default index.html file at the path  
 }  
  
  
 error\_page 404 /404.html;  
 location = /40x.html {  
 }  
  
 error\_page 500 502 503 504 /50x.html;  
 location = /50x.html {  
 }  
 }  
}  
EOF  
  
## Створюємо статичні веб сторінки  
sudo mkdir -p /etc/nginx/www  
cat > /etc/nginx/www/index.html <<'EOF'   
<h1> Hello There Server APP2</h1>  
 <p>  
 This webpage is serverd through nginx at default root path  
 </p>  
EOF  
sudo chmod 0755 /etc/nginx/www  
sudo chmod 644 /etc/nginx/www/index.html  
sudo echo "index webpage created " >> /tmp/debug.log  
  
sudo mkdir -p /etc/nginx/www/bar  
cat > /etc/nginx/www/bar/index.html <<'EOF'   
<h1> Hello There Server APP2</h1>  
 <p>  
 This webpage is serverd through nginx at path /$root/bar  
 </p>  
EOF  
sudo chmod 0755 /etc/nginx/www/bar  
sudo chmod 644 /etc/nginx/www/bar/index.html  
sudo echo "index webpage created for /bar" >> /tmp/debug.log  
  
## Встановлюємо firewalld, в відкриваємо тільки потрібні нам порти  
sudo apt install -y firewalld  
sudo systemctl unmask firewalld  
sudo systemctl restart firewalld  
sudo firewall-cmd --zone=public --permanent --add-service=http  
sudo firewall-cmd --zone=public --permanent --add-service=https

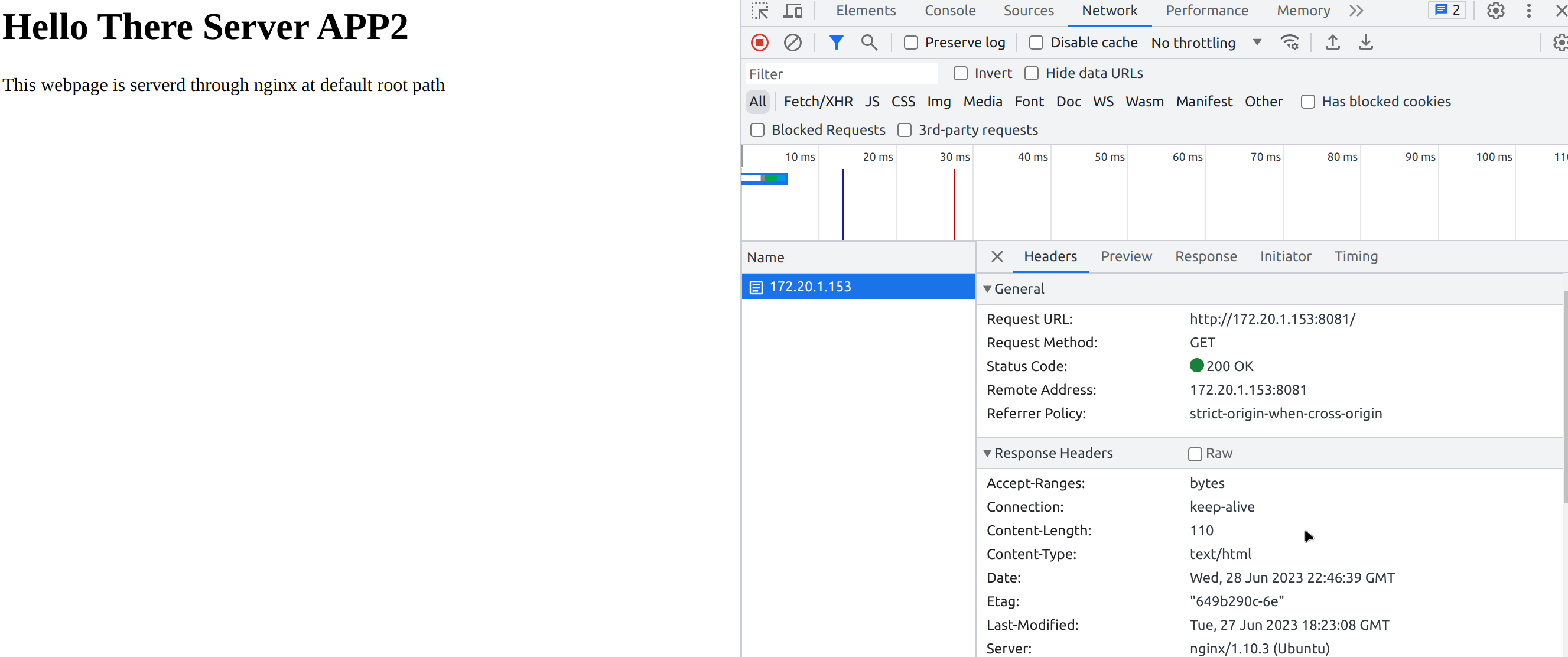
## Робимо рестарт фаєрвола і nginx для застосування всіх змін

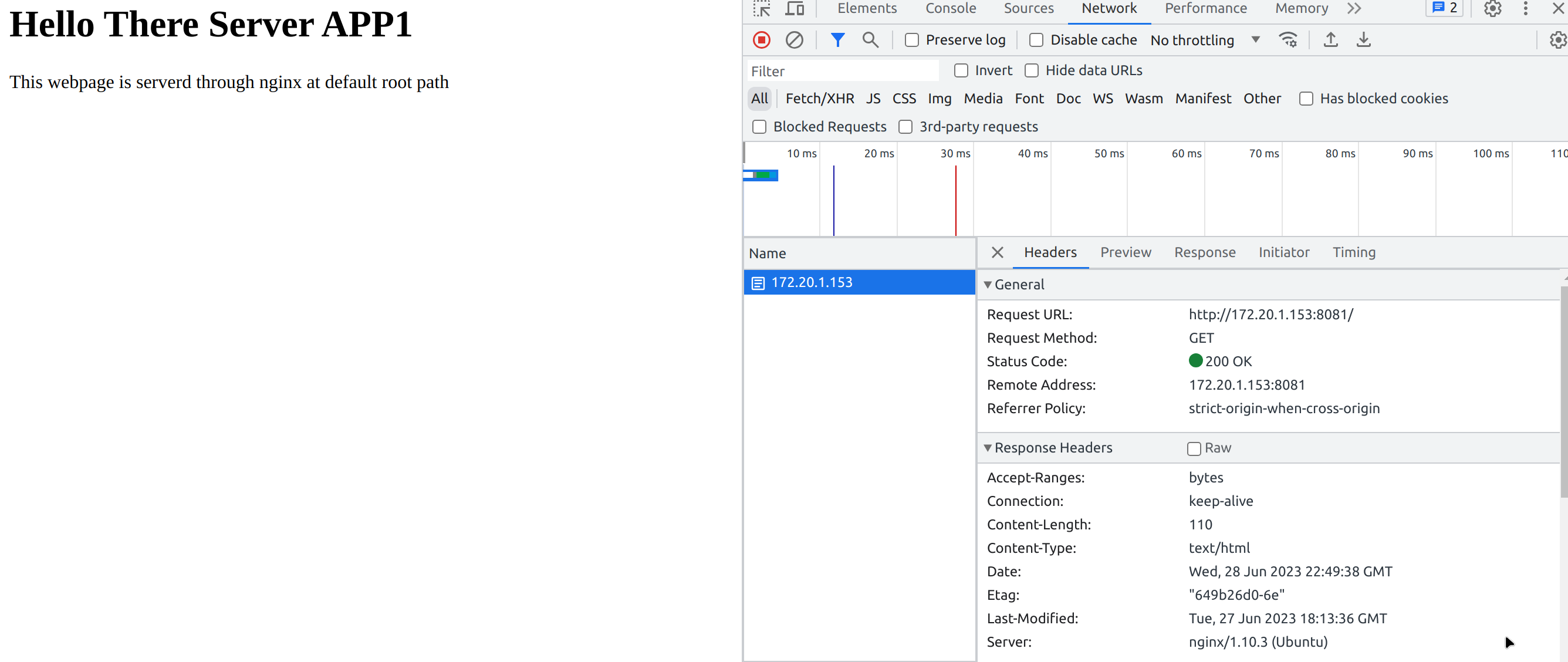
sudo firewall-cmd --reload  
sudo systemctl enable firewalld  
sudo systemctl restart firewalld  
sudo systemctl restart nginx

2.4. Виконати аналіз протокольного обміну між клієнтом та сервером балансування навантаження під час виконання запитів по протоколам HTTP та HTTPS.

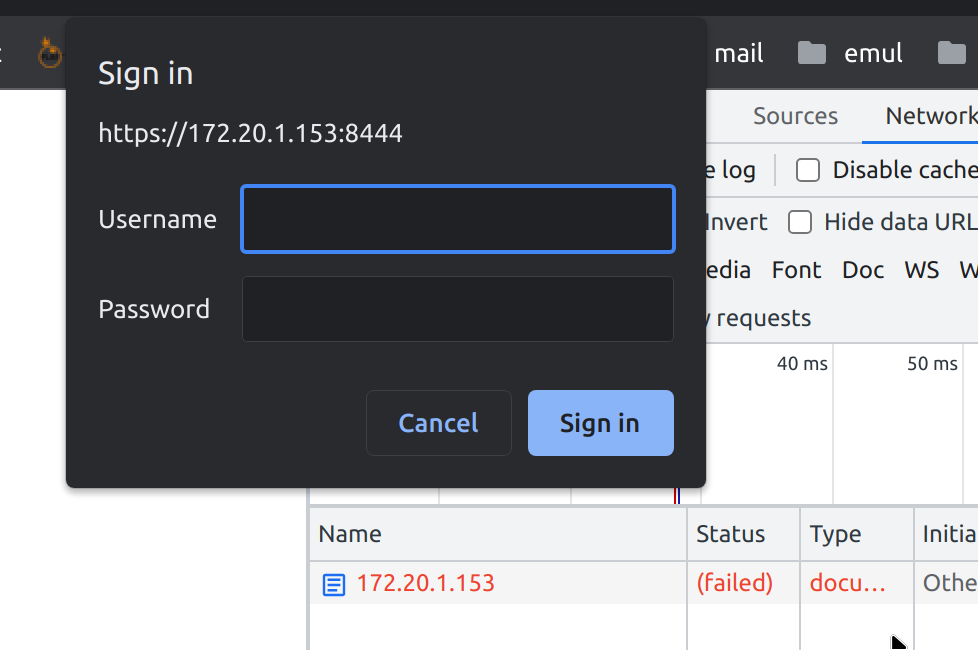
Перевіряємо роботу сервера спочатку HTTP, який у нас має бути на порті 8081 і як ми бачимо для того щоб нам зробити запити потрібно спочатку пройти аутентифікацію, вводимо будь якого юзера з варіанту з паролем 123



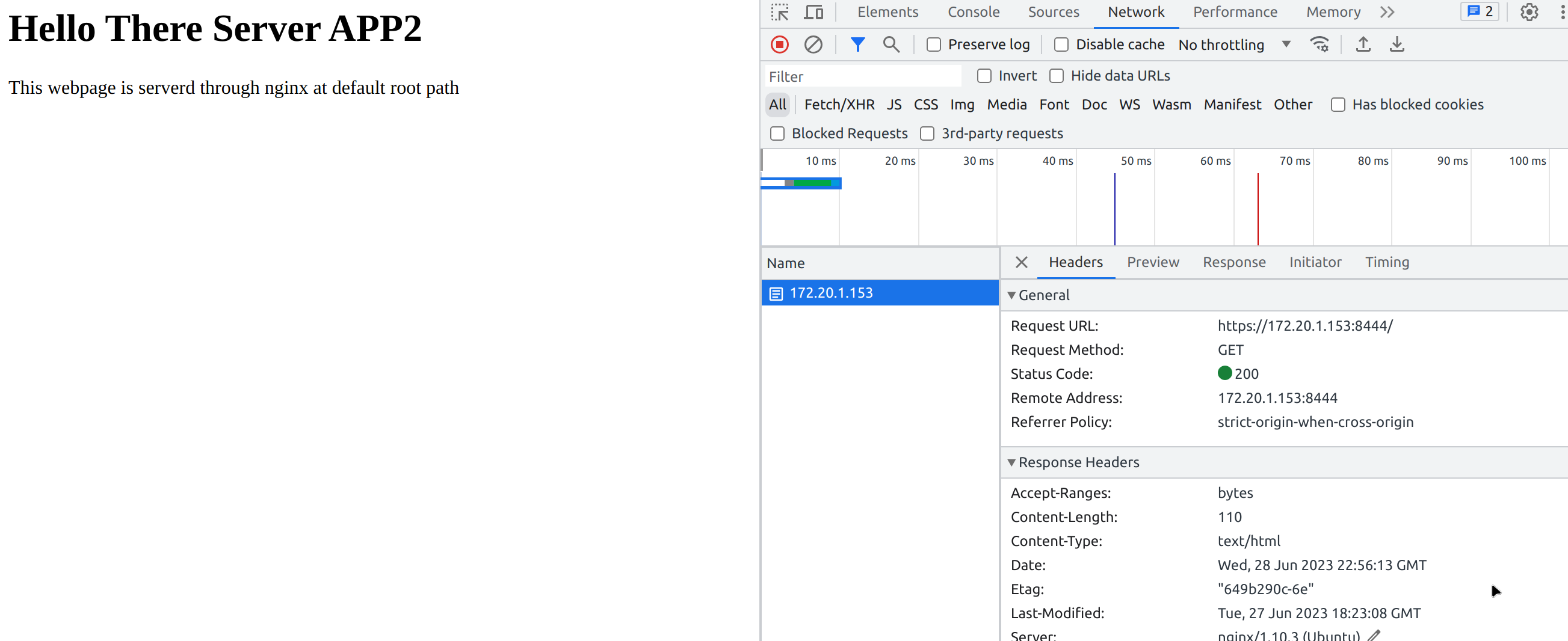
після того як ми залогінились ми попадаємо на статичну сторінгу одного з аплікейшин серверів

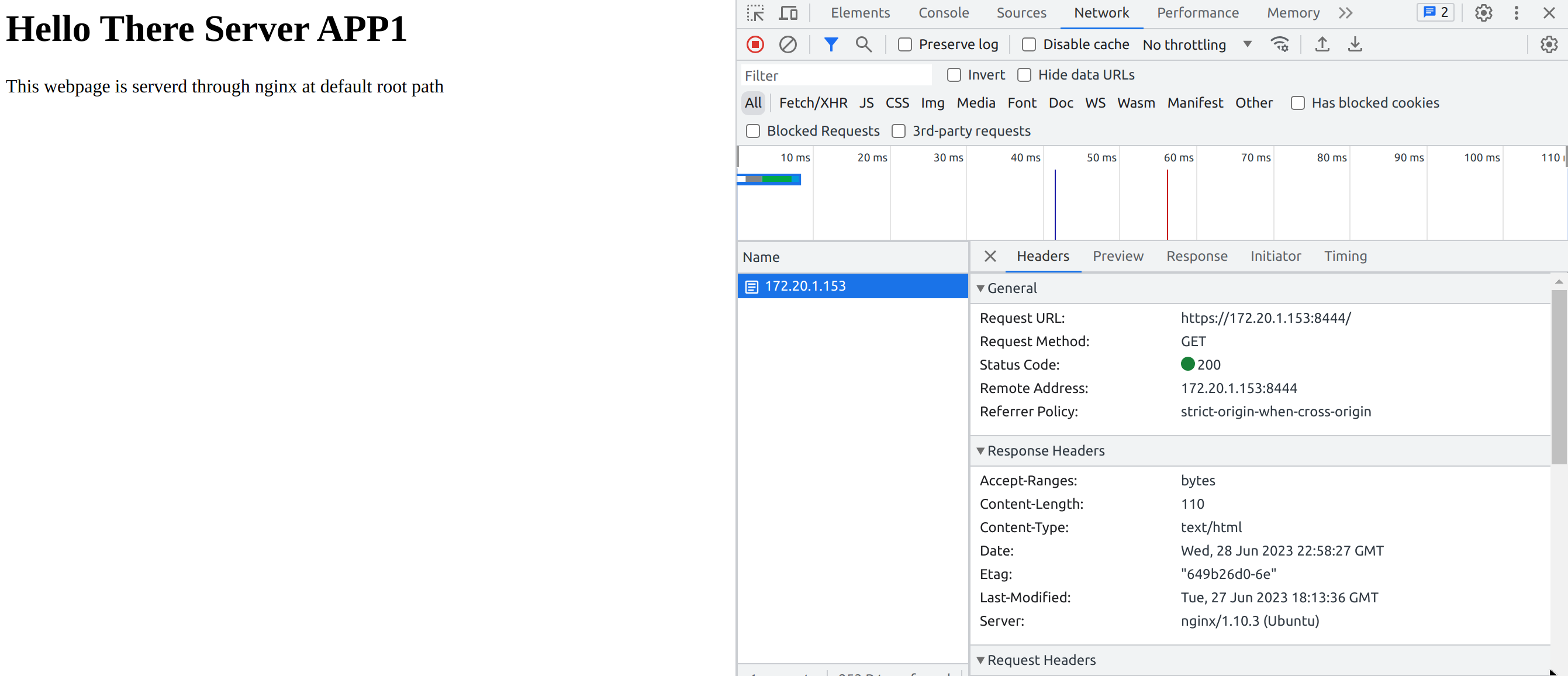


повторивши запит декілька разів ми бачимо що попадаємо приблизно 50/50 запитів то на один то на інший сервіс а значить балансування трафіку працює

Далі перевіримо протокол HTTPS який у нас на порті 8444?як бачимо у нас йде аналогічний запит на аутентифікацію

після проходження аутентифікації ми аналогічно потрапляємо на один з аплікейшин серверів, і повторивши декілька запитів бачимо що балансування працює





2.6. Для перевірки роботи Web-серверу та аналізу протокольного обміну рекомендується використовувати утиліти: telnet, netcat, openssl, curl.

nc -z -v 172.20.1.153 8080-8082

nc: connect to 172.20.1.153 port 8080 (tcp) failed: No route to host

Connection to 172.20.1.153 8081 port [tcp/tproxy] succeeded!

nc: connect to 172.20.1.153 port 8082 (tcp) failed: No route to host

nc -z -v 172.20.1.153 8443-8445

nc: connect to 172.20.1.153 port 8443 (tcp) failed: No route to host

Connection to 172.20.1.153 8444 port [tcp/\*] succeeded!

nc: connect to 172.20.1.153 port 8445 (tcp) failed: No route to host

openssl s\_client -showcerts -connect 172.20.1.153:8444

CONNECTED(00000003)

Can't use SSL\_get\_servername

depth=0 C = UA, ST = Ukraine, L = Kyiv, O = KPI, OU = OT, CN = letter CA

verify error:num=18:self-signed certificate

verify return:1

depth=0 C = UA, ST = Ukraine, L = Kyiv, O = KPI, OU = OT, CN = letter CA

verify return:1

---

Certificate chain

0 s:C = UA, ST = Ukraine, L = Kyiv, O = KPI, OU = OT, CN = letter CA

i:C = UA, ST = Ukraine, L = Kyiv, O = KPI, OU = OT, CN = letter CA

a:PKEY: rsaEncryption, 2048 (bit); sigalg: RSA-SHA256

v:NotBefore: Jun 28 23:18:10 2023 GMT; NotAfter: Jun 27 23:18:10 2024 GMT

-----BEGIN CERTIFICATE-----

MIIDNjCCAh4CCQDC8AgSCUmoOjANBgkqhkiG9w0BAQsFADBdMQswCQYDVQQGEwJV

QTEQMA4GA1UECAwHVWtyYWluZTENMAsGA1UEBwwES3lpdjEMMAoGA1UECgwDS1BJ

MQswCQYDVQQLDAJPVDESMBAGA1UEAwwJbGV0dGVyIENBMB4XDTIzMDYyODIzMTgx

MFoXDTI0MDYyNzIzMTgxMFowXTELMAkGA1UEBhMCVUExEDAOBgNVBAgMB1VrcmFp

bmUxDTALBgNVBAcMBEt5aXYxDDAKBgNVBAoMA0tQSTELMAkGA1UECwwCT1QxEjAQ

BgNVBAMMCWxldHRlciBDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB

ANBeRIY7+oFUj5qvdj6uwwwFkAd+GcclIvNfhv/DiqOX4YbVigoEDGhNiNMQ6Etf

GHpbmxdiDMjbC6Ynq4PsW0ALmEzMGfUPRN2xkr6YG9ZlaRAt9z3RxlarHUf4M7t+

l+nmUBzKLAzqtUSdLPRZa4tXnIuxN2w1QX0lFudKVpbNNQ8+0TPEQwyJeHYMMDMf

YXEpz8SMBTt0nMNIuK+gw+NjmNPFc0BeWdMcYFX0AQK0FCEJqNA5QmDFuPF63mXi

kaa0Bv8iIIn2mtlDS72ChMpmNJOEc8h9NvOfuM+ooVqzKqpUT3OzdvinvBiVGVGt

ewJm0dSbxiLYR5eAlAGCx6cCAwEAATANBgkqhkiG9w0BAQsFAAOCAQEABvBTkR9J

NATAxbPebNwK+JkOrxNf/qkP9r2u/Ny+REUCrTFolYK+j12B3S8BppvBlQfslgVj

wxFtG3ZcWZTcHlKMF1s71KYYo8ADXEFvcMyKd7Z8cIExeq6zfHYkwzQfkZ/2v6qT

pa5Ka6aYeICECw+RAymgQQ9WN5nCHEpZQg0bg4xhgsC9NPv17GSqcg45qhho2Phx

4IOq5Qpd0suTrJwHMlCQoN+ggTjyIqb1GqosQjVLGqATQQggIlmAHN7LfRF9zRwL

sJRHxvIpy6GiR5rnNUL6lM/sNhdQiPGKlYBaUtY72QFbJjHbekr5d/3YYSBFoZJ5

UeDhfxjUsaH4/Q==

-----END CERTIFICATE-----

---

Server certificate

subject=C = UA, ST = Ukraine, L = Kyiv, O = KPI, OU = OT, CN = letter CA

issuer=C = UA, ST = Ukraine, L = Kyiv, O = KPI, OU = OT, CN = letter CA

---

No client certificate CA names sent

Peer signing digest: SHA512

Peer signature type: RSA

Server Temp Key: ECDH, prime256v1, 256 bits

---

SSL handshake has read 1496 bytes and written 419 bytes

Verification error: self-signed certificate

---

New, TLSv1.2, Cipher is ECDHE-RSA-AES256-GCM-SHA384

Server public key is 2048 bit

Secure Renegotiation IS supported

Compression: NONE

Expansion: NONE

No ALPN negotiated

SSL-Session:

Protocol : TLSv1.2

Cipher : ECDHE-RSA-AES256-GCM-SHA384

Session-ID: EEB410ADD39E74F7BC62CB2D4AD252ACDAD8D985E74E0E22DBE0247B00892CF2

Session-ID-ctx:

Master-Key: D2ECDF67749293ACEF8C0643632F59230112C340ACD4F1E15B41A9E9F6DDEA603FF0B70B81B645A4838F7657751DC012

PSK identity: None

PSK identity hint: None

SRP username: None

TLS session ticket lifetime hint: 600 (seconds)

TLS session ticket:

0000 - b0 99 76 cc ef a1 9e 09-58 a0 49 44 e9 5f f8 7a ..v.....X.ID.\_.z

0010 - ac a9 1f 3f c0 12 74 41-0f f1 a9 27 15 4f c6 fe ...?..tA...'.O..

0020 - 50 a6 3a 25 c2 6d eb 2b-e3 d7 1c e4 e4 42 61 1a P.:%.m.+.....Ba.

0030 - 9a 5a 53 d9 26 10 21 b7-b1 0f 2a da e2 db 7b f8 .ZS.&.!...\*...{.

0040 - 57 9d d5 e0 34 b1 c2 db-ef 58 f5 86 3e ed 93 19 W...4....X..>...

0050 - 29 d2 a6 d6 1d f2 9d dd-d8 b4 de 91 6e 55 fe 52 )...........nU.R

0060 - 94 de 42 aa 89 cf 64 53-ac d7 a2 81 32 b1 77 e2 ..B...dS....2.w.

0070 - 34 a7 ba c5 6d 5b 04 d3-01 20 f0 2c 22 e1 5f 96 4...m[... .,".\_.

0080 - ab 7f 8a de c2 2f 74 ab-8f e8 d1 b4 ef 3a 13 00 ...../t......:..

0090 - c6 e9 a0 0e 0c b2 49 f7-c5 67 28 83 eb ea 63 e2 ......I..g(...c.

00a0 - 08 ae b6 6c 0d 4e 94 03-23 f3 c9 58 57 d5 b6 4a ...l.N..#..XW..J

Start Time: 1687994365

Timeout : 7200 (sec)

Verify return code: 18 (self-signed certificate)

Extended master secret: no

---