**Project2: Proxy Server and Network Address Translation (NAT)**

M2608.001200 Introduction to Data Communication Networks (2021 Fall)

Name: Yu Sangyoon Student ID: 2016-13343

1. Explain how packets are exchanged between a client to a server with the layering architecture shown below when using HTTP proxy and NAT (DNAT+SNAT configuration), respectively. (Draw arrows for request and response packets on the figures below to describe the path of packets and use these figures for your explanation). (30pts)

In HTTP proxy, get domain and path in application layer.

But in NAT, only by iptables on network layer, connect port and domain by iptables.

HTTP request.

Diagram, schematic

Description automatically generated

HTTP response.

Diagram, schematic

Description automatically generated

NAT request.

Diagram

Description automatically generated

NAT response.

Diagram

Description automatically generated

1. List the pros and cons of HTTP proxy and NAT proxy and compare them. (20pts)

NAT works at the network layer. It can be used to various applications. No additional modification should be done to application layer protocol. And since only using upto network layer by only iptables, cheap switch. So it has higher scalability. But because it is on network layer, no additional modification on file nor authentication of the use via id/password is possible.

Proxy server otherhand, works on the application layer. It requires modification in application level protocol, and thus slow and requires more memory space in proxy server application layer than NAT. And webcaching and computing on the fly(on network) is possible. But since application layer sees data, modification of file or authentication of user via id/password is possible. And switch is costly.

1. Describe how you implemented HTTP proxy server and how the addresses are changed by iptables rules with a screenshot of *result.pcap* opened by Wireshark. (30pts)

1) HTTP proxy server

Receive HTTP request from client (0)

🡪 get destination server hostname / port / path (1)

🡪 make new tcp socket connection with destination server hostname & port (2)

🡪 send HTTP request with modified path with removing server hostname and port (3)

🡪 receive HTTP response from the destination server (4)

🡪 modify content if needed (5)

🡪 send HTTP response to client (6)

(0) is similar to project 1

(1) GET [http://snu.nxclab.org:9000/index.html HTTP/1.1](http://snu.nxclab.org:9000/index.html%20HTTP/1.1)

🡪 http method: GET

🡪 hostname: snu.nxclab.org

🡪 port: 9000

🡪 path: /index.html

🡪 http version: HTTP/1.1

(2) using data from (1), make new connection as client

(3) send modified HTTP request, similar to project 1

(4) receive from destination server

(5) modify content, for modification here, we should disallow encoding when we will by modify the content by modifying the request header in (3)

(6) send response back to client

2) NAT rules

My Environment was WSL(Windows Subsystem for Linux) 2.

NAT rules will be implemented simply as below.

Local host computer 🡪 WSL VM 🡪 destination server 🡪 WSL VM 🡪 local host computer

Text

Description automatically generated

Use inet ip address to access WSL from local host computer.

Why not use wireless network interface ip?

My WSL can’t catch wifi interface of it’s own.

Only eth0 is found in ifconfig.

Next Followed the given instructions.

Graphical user interface

Description automatically generated with low confidence

Since using eth0 in WSL, our mobile phone can’t use the ip address.

So I used host computer browser to check proxy.

Proxy seems to work nicely.

Graphical user interface, text, application, email

Description automatically generated

Lets Check wireshark screenshot with result from tcpdump.

Graphical user interface, text, application

Description automatically generated

In wireshark, we can see that each requests are sent via proxy server 172.22.165.9.

4. One of the applications of the HTTP proxy is web page caching. The cache proxy saves the response file when the file is requested for the first time. Then, when the file is requested again, the proxy sends the stored file to the client without requesting it again from the server. By doing so, clients can receive the file faster. However, files in the server can be updated such that the files in the proxy are not up-to-date. To solve this problem, we can use HTTP conditional requests called ‘if-modified since’. Based on the concept above, provide your pseudocode for implementing HTTP cache proxy which can always send up-to-date files to the client. (20pts)

Skeleton for Pseudocode

Struct HTTPRequest {

string method; /\* e.g, GET \*/

string path; /\* e.g, /index.html \*/

string version; /\* e.g, HTTP/1.1 \*/

Dict<String, String> headers; /\* e.g, Content-Type: text/html \*/

/\* use : req->headers[“Contet-Type”]\*/

}

Struct HTTPResponse {

string version; /\* e.g, HTTP/1.1 \*/

int result\_code; /\* e.g, 200 \*/

string result\_message; /\* e.g, OK \*/

}

main() {

open\_socket\_and\_listen();

while True {

sock\_client = accept\_from\_client();

HTTPRequest req = parse\_http\_req(recv(sock\_client));

forward(req, sock\_client);

}

}

forward(struct HTTPRequest req, int sock\_client) {

sock\_server = open\_socket\_to\_server();

(file\_exist, cache\_file, last\_update\_time) = open\_file(req.path);

**/\* TODO : write your pseudo code for HTTP cache proxy \*/**

***If(file\_exist)***

***If (req.if-modified-since older than last\_update\_time)***

***(file, last-modified) = request(sock\_server, req.path)***

***savefile(file, last-modified);***

***send\_filet(sock\_client, file);***

***Else***

***send\_file(sock\_client, file);***

***Else***

***(file, last-modified) = request(sock\_server, req.path)***

***savefile(file, last-modified);***

***send\_filet(sock\_client, file);***

close(sock\_server)

}

Reference for conditional request: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Conditional_requests>

Diagram, timeline

Description automatically generated

<Example of using If-Modified-Since Request>