

Proxies

- SSL bumping ile Ubuntu 22.04 üzerinde Squid Server 5.7
- Nginx http load balance ve ldap authentication
- Tengine derleme
- HaProxy sni bazlı trafik dağıtımı
- Namespace kullanarak, glb nin fougue tünel fonksiyonunun operasyon testi
- Temel Nginx konfigürasyonu
- Http sunucu kontrol modülü ile NGINX kurulum ve konfigürasyonu
- Debug custom rules using http headers with haproxy

SSL bumping ile Ubuntu 22.04 üzerinde Squid Server 5.7

Squid 5.7 yi ubuntu 20.04 üzerine compile ederek kaynaktan kurmak için yapılması gerekenler

Derlemek için ön gereklilikler

```
apt-get install build-essential openssl libssl-dev pkg-config
```

```
root@log01:~# apt-get install build-essential openssl libssl-dev pkg-config
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssl is already the newest version (1.1.1f-1ubuntu2.16).
openssl set to manually installed.
The following package was automatically installed and is no longer required:
  libfwupdplugin1
Use 'apt autoremove' to remove it.
The following additional packages will be installed:
  dpkg-dev fakeroot g++ g++-9 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libdpkg-perl libfakeroot libfile-fcntllock-perl libstdc++-9-dev
Suggested packages:
  debian-keyring g++-multilib g++-9-multilib gcc-9-doc bzr libssl-doc libstdc++-9-doc
The following NEW packages will be installed:
  build-essential dpkg-dev fakeroot g++ g++-9 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libdpkg-perl libfakeroot libfile-fcntllock-perl libssl-dev
  libstdc++-9-dev pkg-config
0 upgraded, 14 newly installed, 0 to remove and 37 not upgraded.
Need to get 12.9 MB of archives.
After this operation, 60.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libstdc++-9-dev amd64 9.4.0-1ubuntu1-20.04.1 [1,722 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 g++-9 amd64 9.4.0-1ubuntu1-20.04.1 [8,420 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 g++ amd64 4:9.3.0-1ubuntu2 [1,604 B]
Get:4 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libdpkg-perl all 1.19.7ubuntu3.2 [231 kB]
Get:5 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 dpkg-dev all 1.19.7ubuntu3.2 [679 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 build-essential amd64 12.8ubuntu1.1 [4,664 B]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 libfakeroot amd64 1.24-1 [25.7 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 fakeroot amd64 1.24-1 [62.6 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/main amd64 libalgorithm-diff-perl all 1.19.03-2 [46.6 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal/main amd64 libalgorithm-diff-xs-perl amd64 0.04-6 [11.3 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal/main amd64 libalgorithm-merge-perl all 0.08-3 [12.0 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/main amd64 libfile-fcntllock-perl amd64 0.22-3build4 [33.1 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libssl-dev amd64 1.1.1f-1ubuntu2.16 [1,584 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal/main amd64 pkg-config amd64 0.29.1-0ubuntu4 [45.5 kB]
```

Bu işlem bitince download ediyoruz

```
wget -c http://www.squid-cache.org/Versions/v5/squid-5.7.tar.gz
```

```
root@log01:~# wget -c http://www.squid-cache.org/Versions/v5/squid-5.7.tar.gz
--2022-09-28 12:59:05-- http://www.squid-cache.org/Versions/v5/squid-5.7.tar.gz
Resolving www.squid-cache.org (www.squid-cache.org)... 212.199.163.170, 67.215.9.148, 196.200.160.70, ...
Connecting to www.squid-cache.org (www.squid-cache.org)|212.199.163.170|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5474041 (5.2M) [application/x-gzip]
Saving to: 'squid-5.7.tar.gz'

squid-5.7.tar.gz          100%[=====] 5.22M  2.38MB/s  in 2.2s

2022-09-28 12:59:08 (2.38 MB/s) - 'squid-5.7.tar.gz' saved [5474041/5474041]

root@log01:~#
```

Download edilen kaynak dosyasını açıyoruz.

```
tar zxvf squid-5.7.tar.gz
```

```
-rw-r--r-- 1 root root 5474041 Sep  5 16:10 squid-5.7.tar.gz
drwx----- 2 root root  4096 Nov  9 2021 .ssh/
-rw----- 1 root root  9683 Jun 30 08:44 .viminfo
-rw-r--r-- 1 root root  216 Feb  4 2022 .wget-hsts
-rw-r--r-- 1 root root 1278 Feb  4 2022 wget-log
root@log01:~# tar zxvf squid-5.7.tar.gz
squid-5.7/
squid-5.7/src/
squid-5.7/src/DiskIO/
squid-5.7/src/DiskIO/Mmapped/
squid-5.7/src/DiskIO/Mmapped/MmappedFile.cc
squid-5.7/src/DiskIO/Mmapped/Makefile.am
```

Çıkardığımız klasöre geçiyoruz

```
root@log01:~# cd squid-5.7/
root@log01:~/squid-5.7#
root@log01:~/squid-5.7#
root@log01:~/squid-5.7#
```

Derleme

Aşağıdaki parametreler ile compile başlatıyoruz

```
./configure --prefix=/usr --with-openssl --enable-ssl-crtld --localstatedir=/var --libexecdir=${prefix}/lib/squid --
datadir=${prefix}/share/squid --sysconfdir=/etc/squid --with-default-user=proxy --with-logdir=/var/log/squid --
with-pidfile=/var/run/squid.pid
```

```
root@log01:~/squid-5.7# ./configure --prefix=/usr --with-openssl --enable-ssl-crtld --localstatedir=/var --libexecdir=${prefix}/lib/squid --datadir=${prefix}/share/squid --sysconfdir=/etc
/squid --with-default-user=proxy --with-logdir=/var/log/squid --with-pidfile=/var/run/squid.pid
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a race-free mkdir -p... /usr/bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking whether make supports nested variables... yes
checking whether UID '0' is supported by ustar format... yes
checking whether GID '0' is supported by ustar format... yes
checking how to create a ustar tar archive... gnutar
checking whether to enable maintainer-specific portions of Makefiles... no
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether the compiler supports GNU C... yes
checking whether gcc accepts -g... yes
checking for gcc option to enable C11 features... none needed
checking whether gcc understands -c and -o together... yes
checking whether make supports the include directive... yes (GNU style)
checking dependency style of gcc... gcc3
checking for g++... g++
```

Hiçbir warning yada hata almadan bitmeli. Ardından derleme işlemini aşağıdaki komut ile başlatıyoruz. Bu 15 dakika kadar sürebiliyor.

```
make all
```

```
error -pipe -D REENTRANT -g -O2 -march=native -g -O cachemgr.cgi cachemgr_CGIEXT -CharacterSet.o cachemgr_CGIEXT -Here.o cachemgr_CGIEXT -MemBuf.o cachemgr_CGIEXT -cachemgr.o cachemgr_CGIEXT -test tools.o cachemgr_CGIEXT -time.o tests/cachemgr_CGIEXT -stub_cbdta.o tests/cachemgr_CGIEXT -stub_debug.o tests/cachemgr_CGIEXT -stub_libmem.o ../src/ip/libip.la ../lib/libmimiscoding.la ../lib/libmimiscutil.la ../compat/libcompatsquid.la -lm -lnsl -lresolv -lrt
libtool: link: g++ -DDEFAULT_CACHEMGR_CONFIG=\"etc/squid/cachemgr.conf\" -Wall -Wpointer-arith -Wwrite-strings -Wcomments -Wshadow -Woverloaded-virtual -Werror -pipe -D REENTRANT -g -O2 -march=native -g -O cachemgr.cgi cachemgr_CGIEXT -CharacterSet.o cachemgr_CGIEXT -Here.o cachemgr_CGIEXT -MemBuf.o cachemgr_CGIEXT -cachemgr.o cachemgr_CGIEXT -test tools.o cachemgr_CGIEXT -time.o tests/cachemgr_CGIEXT -stub_cbdta.o tests/cachemgr_CGIEXT -stub_debug.o tests/cachemgr_CGIEXT -stub_libmem.o ../src/ip/libip.la ../lib/libmimiscoding.la ../lib/libmimiscutil.la ../compat/libcompatsquid.la -lm -lnsl -lresolv -lrt
sed " s@DEFAULT_ERROR_DIR@%/share/squid/errors%g; s@DEFAULT_MIME_TABLE@%/etc/squid/mime.conf%g; s@DEFAULT_SSL_CRTD@%/lib/squid/'echo security file certgen | sed 's,x,x,s/$/'%g; s@DEFAULT_SSL_DB_DIR@%/var/cache/squid/ssl_db%g; s@\"PACKAGE_STRING\"@%Squid Web Proxy 5.7%g; s@SYSCONFDIR@%/etc/squid%g; " < ./cachemgr.cgi.8.in > cachemgr.cgi.8
make[2]: Leaving directory '/root/squid-5.7/tools'
make[1]: Leaving directory '/root/squid-5.7/tools'
Making all in test-suite
make[1]: Entering directory '/root/squid-5.7/test-suite'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/root/squid-5.7/test-suite'
make[1]: Entering directory '/root/squid-5.7'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/root/squid-5.7'
```

Make işlemi sorunsuz şekilde bittikten sonra

make install

yapıyoruz ve kurulum bitiyor.

Kontrol edelim.

squid -v

Aşağıdaki gibi görünmeli

```
root@log01:~/squid-5.7# cd
root@log01:~/squid-5.7# squid -v
Squid Cache: Version 5.7
Service Name: squid

This binary uses OpenSSL 1.1.1f  31 Mar 2020. For legal restrictions on distribution see https://www.openssl.org/source/license.html

configure options: '--prefix=/usr' '--with-openssl' '--enable-ssl-crtld' '--localstatedir=/var' '--libexecdir=/lib/squid' '--datadir=/share/squid' '--sysconfdir=/etc/squid' '--with-default-user=proxy' '--with-logdir=/var/log/squid' '--with-pidfile=/var/run/squid.pid' --enable-ltdl-convenience
root@log01:~/squid-5.7#
```

Systemd dosyası

Systemd servisi Ubuntu 22.04 versiyon linux kurulumlarının sistem servis yöneticisidir. Bu yöneticiye squid için gerekli detayların eklenmesi gerekir.

Derleyerek kurduğumuz squid 5.7 systemd dosyası olmadan gelir, oluşturmak için aşağıdaki şablonu kullanaabilirsiniz

```
## Copyright (C) 1996-2022 The Squid Software Foundation and contributors

##

## Squid software is distributed under GPLv2+ license and includes
## contributions from numerous individuals and organizations.
## Please see the COPYING and CONTRIBUTORS files for details.

##

[Unit]
Description=Squid Web Proxy Server
Documentation=man:squid(8)
```

```
After=network.target network-online.target nss-lookup.target
```

```
[Service]
```

```
Type=forking
```

```
PIDFile=/var/run/squid.pid
```

```
#ExecStartPre=/usr/sbin/squid -z
```

```
ExecStart=/usr/sbin/squid -f /etc/squid/squid.conf -d1
```

```
ExecStop=/usr/sbin/squid -k shutdown
```

```
ExecReload=/usr/sbin/squid -k reconfigure
```

```
[Install]
```

```
WantedBy=multi-user.target
```

Bu şablonu kopyalayıp aşağıdaki komutu kullanarak yeni systemd servis dosyasının içine kopyalayalım

```
sudo nano /lib/systemd/system/squid.service
```

```
root@log01:~#  
root@log01:~# sudo nano /lib/systemd/system/squid.service
```

kopyaladıktan sonra aşağıdaki gibi görünmelidir.

```
GNU nano 4.8 /lib/systemd/system/squid.service Modified
## Copyright (C) 1996-2022 The Squid Software Foundation and contributors
##
## Squid software is distributed under GPLv2+ license and includes
## contributions from numerous individuals and organizations.
## Please see the COPYING and CONTRIBUTORS files for details.
##

[Unit]
Description=Squid Web Proxy Server
Documentation=man:squid(8)
After=network.target network-online.target nss-lookup.target

[Service]
Type=forking
PIDFile=/var/run/squid.pid
#ExecStartPre=/usr/sbin/squid -z
ExecStart=/usr/sbin/squid -f /etc/squid/squid.conf -d1
ExecStop=/usr/sbin/squid -k shutdown
ExecReload=/usr/sbin/squid -k reconfigure

[Install]
WantedBy=multi-user.target

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos    M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line  M-E Redo
[0] 0:python3- 1:bash 2:htop 3:sudo* "log01" 13:37 28-Sep-22
```

ctrl+x, yes, yes yapıp çıkıyoruz

```
root@log01:~# systemctl daemon-reload
```

yapıp dameon dosyalarının yeniden yüklendiğine emin oluyoruz

```
systemctl status squid.service
```

yaptıktan sonra squid servisinin çalıştığını görebilirsiniz.

```
root@log01:~# systemctl daemon-reload
root@log01:~# systemctl status squid.service
● squid.service - Squid Web Proxy Server
   Loaded: loaded (/lib/systemd/system/squid.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: man:squid(8)
root@log01:~# systemctl start squid.service
root@log01:~# systemctl status squid.service
● squid.service - Squid Web Proxy Server
   Loaded: loaded (/lib/systemd/system/squid.service; disabled; vendor preset: enabled)
   Active: active (running) since Wed 2022-09-28 13:39:59 UTC; 1s ago
     Docs: man:squid(8)
  Process: 3247294 ExecStart=/usr/sbin/squid -f /etc/squid/squid.conf -d1 (code=exited, status=0/SUCCESS)
 Main PID: 3247307 (squid)
    Tasks: 2 (limit: 9418)
   Memory: 9.8M
    CGroup: /system.slice/squid.service
            └─3247307 /usr/sbin/squid -f /etc/squid/squid.conf -d1
              └─3247312 (squid-1) --kid squid-1 -f /etc/squid/squid.conf -d1

Sep 28 13:40:00 log01 squid[3247312]: 2022/09/28 13:40:00 kid1| Max Swap size: 0 KB
Sep 28 13:40:00 log01 squid[3247312]: 2022/09/28 13:40:00 kid1| Using Least Load store dir selection
Sep 28 13:40:00 log01 squid[3247312]: 2022/09/28 13:40:00 kid1| Set Current Directory to /var/cache/squid
Sep 28 13:40:00 log01 squid[3247307]: Squid Parent: (squid-1) process 3247312 started
Sep 28 13:40:00 log01 squid[3247312]: 2022/09/28 13:40:00 kid1| Finished loading MIME types and icons.
Sep 28 13:40:00 log01 squid[3247312]: 2022/09/28 13:40:00 kid1| HTCP Disabled.
Sep 28 13:40:00 log01 squid[3247312]: 2022/09/28 13:40:00 kid1| Squid plugin modules loaded: 0
Sep 28 13:40:00 log01 squid[3247312]: 2022/09/28 13:40:00 kid1| Adaptation support is off.
Sep 28 13:40:00 log01 squid[3247312]: 2022/09/28 13:40:00 kid1| Accepting HTTP Socket connections at conn2 local=0.0.0.0:3128 remote=[::] FD 8 flags=9
Sep 28 13:40:00 log01 squid[3247313]: fopen: Permission denied
root@log01:~#
```

Konfigürasyon

squid konfigürasyon dosyalarının olduğu yere gidiyoruz

```
cd /etc/squid/
```

mevcut konfigürasyonu yedekliyoruz.

```
root@log01:/etc/squid# mv squid.conf squid.conf.disabled
```

Yeni konfigürasyonu yapıştıracağımız dosyayı nano kullanarak açıyoruz

```
root@log01:/etc/squid# nano squid.conf
```

Aşağıdaki şablonu yeni açtığımız dosya içerisine kaydediyoruz.

```
# Recommended minimum configuration:
#
##
# NTLM
##
#auth_param ntlm program /usr/bin/ntlm_auth --diagnostics --helper-protocol=squid-2.5-ntlmssp --
domain=BANKA
#auth_param ntlm children 10
```

```
#auth_param ntlm keep_alive off
```

```
#icap_send_client_username on
```

```
#acl lan proxy_auth REQUIRED
```

```
# Example rule allowing access from your local networks.
```

```
# Adapt to list your (internal) IP networks from where browsing
```

```
# should be allowed
```

```
acl localnet src 0.0.0.1-0.255.255.255# RFC 1122 "this" network (LAN)
```

```
acl localnet src 10.0.0.0/8# RFC 1918 local private network (LAN)
```

```
acl localnet src 100.64.0.0/10# RFC 6598 shared address space (CGN)
```

```
acl localnet src 169.254.0.0/16# RFC 3927 link-local (directly plugged) machines
```

```
acl localnet src 172.16.0.0/12# RFC 1918 local private network (LAN)
```

```
acl localnet src 192.168.0.0/16# RFC 1918 local private network (LAN)
```

```
acl localnet src fc00::/7# RFC 4193 local private network range
```

```
acl localnet src fe80::/10# RFC 4291 link-local (directly plugged) machines
```

```
acl SSL_ports port 443
```

```
acl Safe_ports port 80# http
```

```
acl Safe_ports port 21# ftp
```

```
acl Safe_ports port 443# https
```

```
acl Safe_ports port 70# gopher
```

```
acl Safe_ports port 210# wais
```

```
acl Safe_ports port 1025-65535# unregistered ports
```

```
acl Safe_ports port 280# http-mgmt
```

```
acl Safe_ports port 488# gss-http
```

```
acl Safe_ports port 591# filemaker
```

```
acl Safe_ports port 777# multiling http
```

```
#
```

```
# Recommended minimum Access Permission configuration:
```

```
#
```

```
# Deny requests to certain unsafe ports
```

```
http_access deny !Safe_ports
```

```
# Deny CONNECT to other than secure SSL ports
```

```
http_access deny CONNECT !SSL_ports
```



```
# Only allow cachemgr access from localhost
http_access allow localhost manager
http_access deny manager

# We strongly recommend the following be uncommented to protect innocent
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
#http_access deny to_localhost

#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#

# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed

#http_access allow localnet
http_access allow localhost

# And finally deny all other access to this proxy
http_access allow all

# Squid normally listens to port 3128
#http_port 3128
http_port 3128 tcpkeepalive=60,30,3 ssl-bump generate-host-certificates=on
dynamic_cert_mem_cache_size=20MB cert=/etc/squid/bump.crt key=/etc/squid/bump.key
cipher=HIGH:MEDIUM:!LOW:!RC4:!SEED:!IDEA:!3DES:!MD5:!EXP:!PSK:!DSS
options=NO_TLSv1,NO_SSLv3,NO_SSLv2,SINGLE_DH_USE,SINGLE_ECDH_USE tls-
dh=prime256v1:/etc/squid/bump_dhparam.pem

# Uncomment and adjust the following to add a disk cache directory.
#cache_dir ufs /var/cache/squid 100 16 256

# Leave coredumps in the first cache dir
coredump_dir /var/cache/squid
```

```
#
# Add any of your own refresh_pattern entries above these.
#
refresh_pattern ^ftp:[]1440[]20%[]10080
refresh_pattern ^gopher:[]1440[]0%[]1440
refresh_pattern -i (/cgi-bin/|\?) 0[]0%[]0
refresh_pattern .[]0[]20%[]4320

sslrctd_program /usr/lib/squid/security_file_certgen -s /var/lib/squid/ssl_db -M 20MB

sslproxy_cert_error allow all

ssl_bump stare all
```

Sertifikalandırma

kullanıcılara gönderilecek sertifikayı generate ediyoruz

```
openssl req -new -newkey rsa:2048 -days 720 -nodes -x509 -keyout bump.key -out bump.crt
```

```

root@log01:/etc/squid/ssl_certs# openssl req -new -newkey rsa:2048 -days 720 -nodes -x509 -keyout bump.key -out bump.crt
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'bump.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:TR
State or Province Name (full name) [Some-State]:MARMARA
Locality Name (eg, city) []:Istanbul
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:*.com

```

Otomatik sertifika jenerasyonu için kullanacağımız parametrelere örnek dosyayı oluşturuyoruz.

```
openssl dhparam -outform PEM -out /etc/squid/ssl_certs/bump_dhparam.pem 2048
chmod 400 bump_dhparam.pem
```

[illegible]

Sertifikanını güvenlik ayarlarını yapıyoruz

```
chown proxy:proxy /etc/squid/bump*  
chmod 400 /etc/squid/bump*
```

Ubuntu için çalışacak olan ssl sertifikalarını içeren klasör ve veri tabanını oluşturuyoruz

```
mkdir -p /var/lib/squid/ssl_db  
/usr/lib/squid/ssl_crt -c -s /var/lib/squid/ssl_db
```

devreye alma

Yukarıdaki adımların tamamını yaptıktan sonra,

```
systemctl restart squid.services
```

ve kontrol için

```
systemctl status squid.service
```

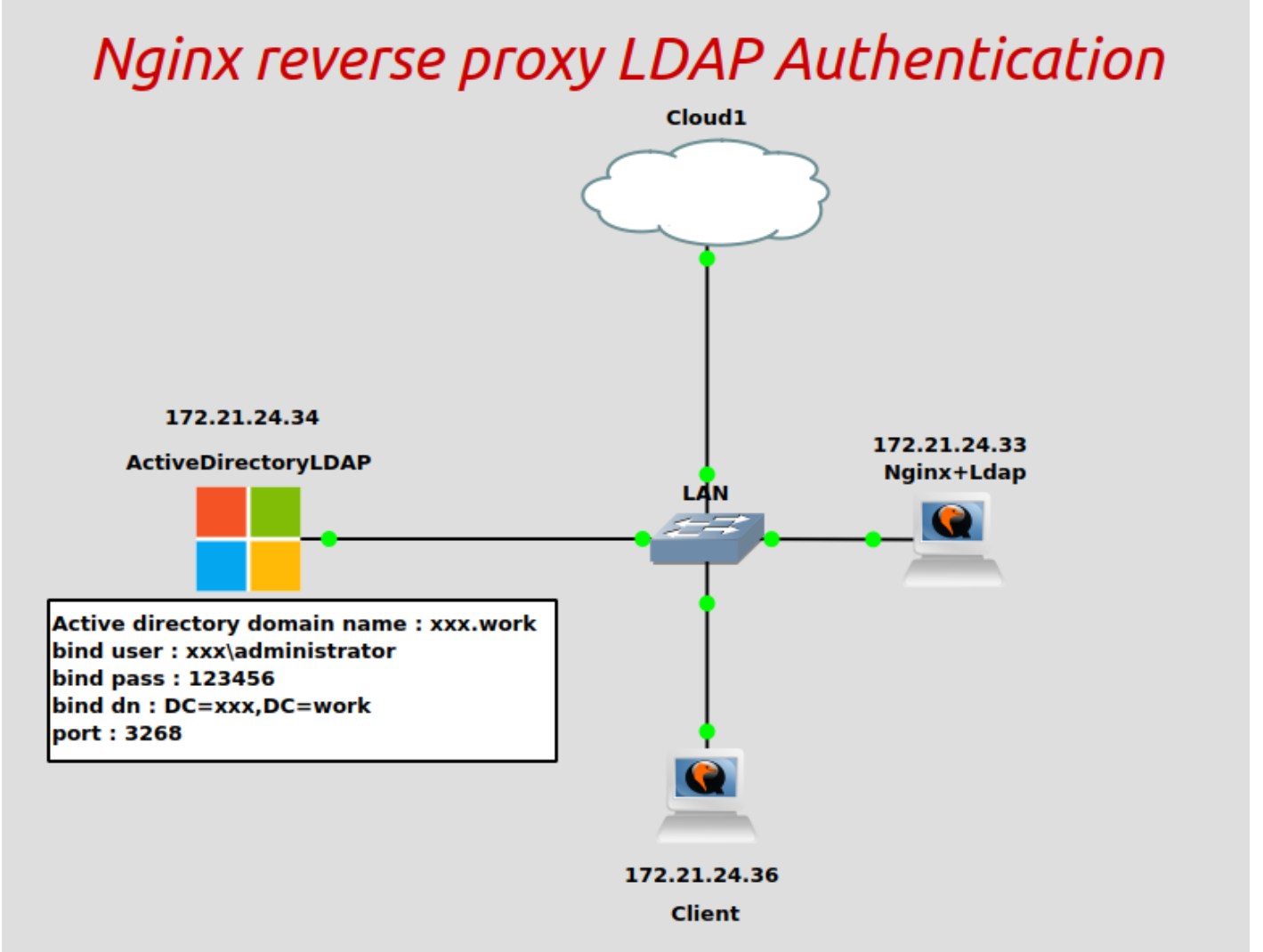
Bu noktadan sonra ;

- oluşturduğumuz ssl sertifikasını clientlara gönderip trusted root authorities klasörüne import ediyoruz
- client üzerinde proxy ayarlarını sunucu_ip _adresi:3128 gösterecek şekilde yapıyoruz.

test için

```
curl -kv -x http://172.17.21.7:3128 https://docs.mikronet.tech
```

Nginx http load balance ve Ldap authentication



Bu dökümanı takip edebilmek için admin seviyesinde linux tecrübesi gereklidir. Bu seviyede tecrübeniz eksik ise teknik destek ekibimizden yardım alabilirsiniz.

Bir müşterimizde kibana önüne koymak üzere active directory ldap entegrasyonu ile birlikte nginx reverse proxy çalıştırmamız gerekiyordu

Malum nginx plus içerisinde gerekli modül pre-install şeklinde bulunuyor fakat opensource olan versiyonda bu modülü bulamıyoruz.

Ancak,

<https://github.com/kvspb/nginx-auth-ldap>

Adresinden, opensource versiyon ile de kullanılabilen bir modülü indirebiliriz. modülü kullanabilmek için bir kaç ön şart var bunlar aşağıdaki gibi ;

- Ubuntu/Debian repolarındaki nginx bu modülü içermiyor bu sebeple source code kullanarak compile etmemiz gerekiyor
- Compile etmeden önce birkaç paket var bunları kurmak gerekiyor.
- kurduktan sonra da nginx.conf ve reverse proxy konfigürasyon içeriğinde bu modülü ve ayarlarını yapmak gerekiyor.

Kuruluma devam etmeden önce ;

- a. Bu install şekli internete açılacak şekilde kullanılmamalıdır.
- b. Brute force gibi tekniklere açıktır.
- c. Ldap bind için kullanılan hesabın şifresi cleartext saklanır dolayısı ile bu servisin çalıştığı makina harden edilmelidir
- d. örnekte protokol olarak ldap kullandım işiniz bitince ldaps ve port değişikliğini yapmayı kesinlikle unutmayınız.

Yukarıdaki uyarıları anladığınıza emin olduktan sonra sırasıyla ;

1. Nginx compile edebilmek için gerekli paketleri kuruyoruz
bu detayda, repositoryleri enable ettikten sonra compile edebilmek için gerekli kütüphaneleri yüklüyoruz

```
$ sudo -i #<-- sudo yapbilen bir kullanıcının şifresini girelim
$ sudo add-apt-repository universe
$ sudo add-apt-repository multiverse
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install ldap-utils zlib1g build-essential gcc make libldap2-dev libssl-dev libpcre3-dev
```

kurulumlar bittikten sonra reboot ediyoruz

2. Git kullanarak modülü indiriyoruz

```
$ cd ~/Downloads
$ git clone https://github.com/kvspb/nginx-auth-ldap
$ cd nginx-auth-ldap
$ git pull
```

3. Nginx source code indiriyor ve deflate ediyoruz
Ben bu denemem de nginx-1.21.6 versiyonunu kullandım

```
$ cd ~/
$ wget -c http://nginx.org/download/nginx-1.21.6.tar.gz
$ tar zxvf nginx-1.21.6.tar.gz
$ cd nginx-1.21.6
```

4. ./configure, make, sudo make instal Bu aşamada nginx i compile edeceğiz bunun için aşağıdaki komutu kullanacağız

```
./configure --user=nginx --group=nginx --prefix=/etc/nginx --sbin-path=/usr/sbin/nginx --conf-  
path=/etc/nginx/nginx.conf --pid-path=/var/run/nginx.pid --lock-path=/var/run/nginx.lock --error-log-  
path=/var/log/nginx/error.log --http-log-path=/var/log/nginx/access.log --with-http_gzip_static_module  
--with-http_stub_status_module --with-http_ssl_module --with-pcre --with-file-aio --with-  
http_realip_module --add-module=/root/nginx-auth-ldap/ --with-ipv6 --with-debug
```

Bütün compile işi sorunsuz tamamlandıktan sonra ;

```
$ make
$ sudo make install
```

5. systemd değişiklikleri

nginx compile edilerek kurulduğunda, systemd service unit dosyası olmadan kuruluyor bunu maalesef elimizle yapmak zorunda kalıyoruz. Bunun için yukarıdaki compile metoduna göre aşağıdaki unit içeriğini kopyalayabilirsiniz

```
[Unit]
Description=A high performance web server and a reverse proxy server
Documentation=man:nginx(8)
After=network.target nss-lookup.target

[Service]
Type=forking
PIDFile=/var/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t -q -g 'daemon on; master_process on;'
ExecStart=/usr/sbin/nginx -g 'daemon on; master_process on;'
ExecReload=/usr/sbin/nginx -g 'daemon on; master_process on;' -s reload
ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid
TimeoutStopSec=5
KillMode=mixed

[Install]
WantedBy=multi-user.target
```

yukarıdaki içeriği /lib/systemd/system/nginx dosyasına yapıştırıp kaydedip çıkın

```
$sudo nano /lib/systemd/system/nginx
```

6. Ldap-auth modülünün çalışabilmesi için gerekli nginx.conf değişiklikleri

Bu adımda ldap-auth modülünü nginx'e eklemek için

```
/etc/nginx/nginx.conf
```

Dosyasını nano ile açıp aşağıdaki içeriği http protokolüne ekliyoruz

```
http {  
    include /etc/nginx/conf.d/*.conf;  
    auth_ldap_cache_enabled on;  
    auth_ldap_cache_expiration_time 1000;  
    auth_ldap_cache_size 1000;  
    ldap_server adds {  
        url "ldap://172.21.24.34:3268/DC=xxx,DC=work?sAMAccountName?sub?(ObjectClass=user)";  
        binddn "XXX\administrator";  
        bind_passwd "12qwasZX";  
        require_valid_user;  
        ssl_check_cert off;  
    }  
}
```

Bu işlem tamamlandıktan sonra default konfig olan ve yine nginx.conf dosyası içindeki server { başlığını comment out ediyoruz

```
# server {
#     listen      80;
#     server_name localhost;

#     #charset koi8-r;

#     #access_log  logs/host.access.log  main;

#     location / {
#         root     html;
#         index    index.html index.htm;
#     }

#     #error_page  404              /404.html;

#     # redirect server error pages to the static page /50x.html
#     error_page   500 502 503 504  /50x.html;
#     location = /50x.html {
#         root     html;
#     }

#     # proxy the PHP scripts to Apache listening on 127.0.0.1:80
#     #location ~ /\.php$ {
#     #     proxy_pass http://127.0.0.1;
#     #}

#     # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
#     #location ~ /\.php$ {
#     #     root           html;
#     #     fastcgi_pass   127.0.0.1:9000;
#     #     fastcgi_index  index.php;
#     # }
```

6. Reverse proxy tanımı /etc/nginx/conf.d içerisine

```
$sudo nano /etc/nginx/conf.d/kibana.conf
```

aşağıdaki detayları kendinize göre değiştirerek yapıştırabilirsiniz

```
upstream kibana {
    server kibana_ip_adresi:port_numarası;
}

server {
    listen 80;

    server_name default_server;

    location / {
        auth_ldap "Enter AD credentials like 'mesut.bayrak@xxx.work'";
        auth_ldap_servers add;
    }
}
```



```
proxy_set_header Host $host;  
proxy_pass http://kibana;  
}  
}
```

7. Testler

Yukarıdaki tanımlar bittikten sonra

```
$sudo nginx -t
```

hata görmezseniz,

```
$sudo nginx restart
```

yazarak ldap authentication modüllü bu çözümü kullanabilirsiniz.

Bu konuda mesut[at]netdev.com.tr den, destek alabilirsiniz

Tengine derleme

Gerekli paketler

```
apt install git build-essential libjemalloc-dev libatomic-ops-dev libpcre3 libpcre3-dev zlib1g zlib1g-dev libssl-dev libgd-dev libgeoip-dev
```

extra modüller

```
https://github.com/vozlt/nginx-module-vts.git
https://github.com/FRiCKLE/nginx_cache_purge.git
https://github.com/yaoweibin/nginx_upstream_check_module.git
https://github.com/GetPageSpeed/nginx_security_headers.git
https://github.com/openresty/headers-more-nginx-module.git
```

Compile

```
#!/bin/bash
./configure \
--with-poll_module \
--with-threads \
--with-file-aio \
--with-http_ssl_module \
--with-http_v2_module \
--with-http_realip_module \
--with-http_image_filter_module \
--with-jemalloc \
--with-libatomic \
--with-http_sub_module \
--with-http_flv_module \
```

```
--with-http_mp4_module \  
--with-http_stub_status_module \  
--with-mail_ssl_module \  
--with-debug \  
--with-stream \  
--without-http_fastcgi_module \  
--without-http_uwsgi_module \  
--without-http_scgi_module \  
--add-module=/var/nginx-module-vts \  
--add-module=/var/nginx_cache_purge \  
--add-module=/var/nginx_upstream_check_module \  
--add-module=/var/nginx_security_headers \  
--add-module=/var/headers-more-nginx-module \  
--add-module=/var/tengine-2.3.2/modules/nginx_http_upstream_vnswrr_module \  
--add-module=/var/tengine-2.3.2/modules/nginx_http_upstream_session_sticky_module \  
--user=nginx \  
--prefix=/etc/nginx \  
--sbin-path=/usr/sbin/nginx \  
--conf-path=/etc/nginx/nginx.conf \  
--pid-path=/var/run/nginx.pid \  
--http-log-path=/var/log/nginx/access.log \  
--error-log-path=/var/log/nginx/error.log \  
--with-http_image_filter_module=dynamic \  
--with-http_geoip_module=dynamic \  
--with-mail=dynamic \  
--http-client-body-temp-path=/tmp/client-body-temp \  
--http-proxy-temp-path=/tmp/proxy-temp \  

```

make && make install

unit file

[Unit]

Description=A high performance web server and a reverse proxy server

After=network.target

[Service]

Type=forking

PIDFile=/var/run/nginx.pid

ExecStartPre=/usr/sbin/nginx -t -q -g 'daemon on; master_process on;'

ExecStart=/usr/sbin/nginx -g 'daemon on; master_process on;'

ExecReload=/usr/sbin/nginx -g 'daemon on; master_process on;' -s reload

ExecStop=-/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /var/run/nginx.pid

TimeoutStopSec=5

KillMode=mixed

LimitNOFILE=1048576

LimitNPROC=1048576

Restart=on-failure

RestartSec=10s

[Install]

WantedBy=multi-user.target[Unit]

Description=A high performance web server and a reverse proxy server

After=network.target

[Service]

Type=forking

PIDFile=/var/run/nginx.pid

ExecStartPre=/usr/sbin/nginx -t -q -g 'daemon on; master_process on;'

ExecStart=/usr/sbin/nginx -g 'daemon on; master_process on;'

ExecReload=/usr/sbin/nginx -g 'daemon on; master_process on;' -s reload

ExecStop=-/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /var/run/nginx.pid

TimeoutStopSec=5

KillMode=mixed

LimitNOFILE=1048576

LimitNPROC=1048576

Restart=on-failure

RestartSec=10s

[Install]

WantedBy=multi-user.target

Tengine Conf

```
worker_processes 15;
worker_cpu_affinity auto 111111111111110;
worker_rlimit_nofile 1048576;
user nginx;
error_log /var/log/nginx/error.log warn;
pid /var/run/nginx.pid;
events {
worker_connections 1048576;
use epoll;
multi_accept on;
accept_mutex on;
}
#don't need stream conf
worker_shutdown_timeout 300;
http {
#access_log on;
gzip_clear_etag off;
vhost_traffic_status_zone shared:vhost_traffic_status:128m;
map_hash_max_size 20480;
map_hash_bucket_size 20480;
map $status $loggable {
    ~^[23] 0;
    default 1;
}
sendfile on;
tcp_nopush on;
tcp_nodelay on;
client_header_timeout 1m;
client_body_timeout 1m;
client_header_buffer_size 2k;
client_body_buffer_size 512k;
client_max_body_size 500m;
large_client_header_buffers 8 16k;
send_timeout 30;
keepalive_timeout 60;
keepalive_requests 100000;
reset_timedout_connection on;
server_tokens off;
server_name_in_redirect off;
server_names_hash_max_size 1024;
```

```
server_names_hash_bucket_size 1024;
check_shm_size 2000m;
proxy_buffer_size 16k;
proxy_buffers 4 16k;
proxy_busy_buffers_size 16k;
proxy_next_upstream off;
more_clear_headers Server;
more_clear_headers server;
log_format secops 'default TENGINE Message 0 0 : "WEBLOG:~'
```

```
    '$time_local~'
    '$remote_addr~'
    '$http_x_client_ip~'
    '$http_True_Client_IP~'
    '$request_method~'
    '$upstream_http_X_Proxy_Cache~'
    '$uri~'
    '$query_string~'
    '$bytes_sent~'
    '$http_user_agent~'
    '$host~'
    '$server_port~'
    '$status~'
    '$http_referer'
    ''',
```

```
log_format main 'default TENGINE Message 0 0 : "WEBLOG:~'
```

```
    '$time_local~'
    '$remote_addr~'
    '$http_x_client_ip~'
    '$http_True_Client_IP~'
    '$request_method~'
    '$upstream_http_X_Proxy_Cache~'
    '$uri~'
    '$query_string~'
    '$bytes_sent~'
    '$http_user_agent~'
    '$host~'
    '$server_port~'
    '$status~'
    '$http_referer~'
    '$upstream_addr~'
```

```

        '$upstream_status~'
        '$upstream_response_time~'
        '$request~'
        '$request_time';

access_log syslog:server=10.84.82.30:5555 main if=$loggable;
error_log syslog:server=10.84.82.30:5556 warn;

# Mime settings
include mime.types;
default_type application/octet-stream;

# Compression settings - aggressively cache text file types
gzip off;
#gzip_comp_level 1;
#gzip_min_length 10240;
#gzip_vary on;
#gzip_buffers 8 64k;
#gzip_types text/plain text/css text/javascript text/js text/xml application/json application/javascript
application/x-javascript application/xml application/xml+rss application/x-font-ttf image/svg+xml font/opentype;
#gzip_proxied any;
#gzip_disable "MSIE [1-6]\.";

# SSL PCI Compliance
ssl_session_cache shared:SSL:10m;
ssl_protocols TLSv1.2;
ssl_ciphers
EECDH+CHACHA20:EECDH+AES128:RSA+AES128:EECDH+AES256:RSA+AES256:EECDH+3DES:RSA+3DES:!MD
5;
ssl_dhparam /etc/nginx/ssl/dhparam.pem;
ssl_prefer_server_ciphers on;

# Error pages
error_page 403 /403.html;
#error_page 404 /error/404.html;
#error_page 502 503 504 /error/50x.html;

# Cache bypass
map $http_cookie $no_cache {
default 0;
~SESS 1;
~wordpress_logged_in 1;
}

# File cache settings
open_file_cache max=200000 inactive=20s;
open_file_cache_valid 30s;

```

```
open_file_cache_min_uses      2;
open_file_cache_errors        on;
# Other settings
log_subrequest                on;
rewrite_log                    on;
include                        /etc/nginx/sites-enabled/*.conf;
include                        /etc/nginx/conf.d/*.conf;
}
```


HaProxy sni bazlı trafik dağıtımı

Routing based on SNI

Aşağıda haproxy için bir ip üzerinden sni özelliği kullanarak forwarding özelliğinin konfig çıktısını bulabilirsiniz.

bu çıktı da ;

- Domainlerimize ait bütün sertifikaların /etc/haproxy/certs/ klasörüne
- cert ve key alt alta aynı dosya içerisinde bulunacak şekilde yüklendikten sonra

http protokolünün sni özelliğini kullanarak farklı endpointlere route etmeyi gösteriyoruz.

```
#-----
# Global settings
#-----
global
    pidfile      /var/run/haproxy.pid
    maxconn      4000
    user         haproxy
    group        haproxy
    #daemon
    debug

#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    option http-server-close
    option forwardfor    except 127.0.0.0/8
    option               redispatch
```

```
retries          3
timeout http-request 10s
timeout queue     1m
timeout connect   10s
timeout client    1m
timeout server    1m
timeout http-keep-alive 10s
timeout check     10s
maxconn          3000
```

```
#-----
```

```
# main frontend which proxys to the backends
```

```
#-----
```

```
frontend main
```

```
bind *:80
acl docs hdr_dom(host) -i docs.mikronet.tech
use_backend bk_docs if docs
http-request redirect scheme https code 301 unless { ssl_fc }
default_backend bk_docs
```

```
frontend mains
```

```
bind *:443 ssl crt /etc/haproxy/certs/
use_backend bk_owncloud if { ssl_fc_sni -i storage.mikroservis.net }
use_backend bk_owncloud if { ssl_fc_sni -i storage.mikronet.tech }
use_backend bk_corelabs if { ssl_fc_sni -i users.corelabs.com.tr }
use_backend bk_hesk if { ssl_fc_sni -i destek.mikronet.tech }
use_backend bk_docs if { ssl_fc_sni -i docs.mikronet.tech }
```

```
backend bk_owncloud
```

```
balance roundrobin
mode http
server owncloud 172.21.23.222:80 check
```

```
backend bk_corelabs
```

```
balance roundrobin
mode http
server users_corelabs 192.168.17.90:80 check
```

```
backend bk_hesk
```

```
    balance      roundrobin
```

```
    mode http
```

```
    server hesk_server 172.21.23.220:80
```

```
backend bk_docs
```

```
    balance      roundrobin
```

```
    mode http
```

```
    server docs_server 172.21.23.239:80 check
```

```
#-----
```

```
# Monitoring stats
```

```
#-----
```

```
listen stats
```

```
bind *:81
```

```
stats enable
```

```
stats uri /
```

```
stats hide-version
```

```
/etc/haproxy #
```

Namespace kullanarak, glb nin fougue t nel fonksiyonunun operasyon testi

Ama

GLB ve apisix in aynı makina  zerinde alıřması

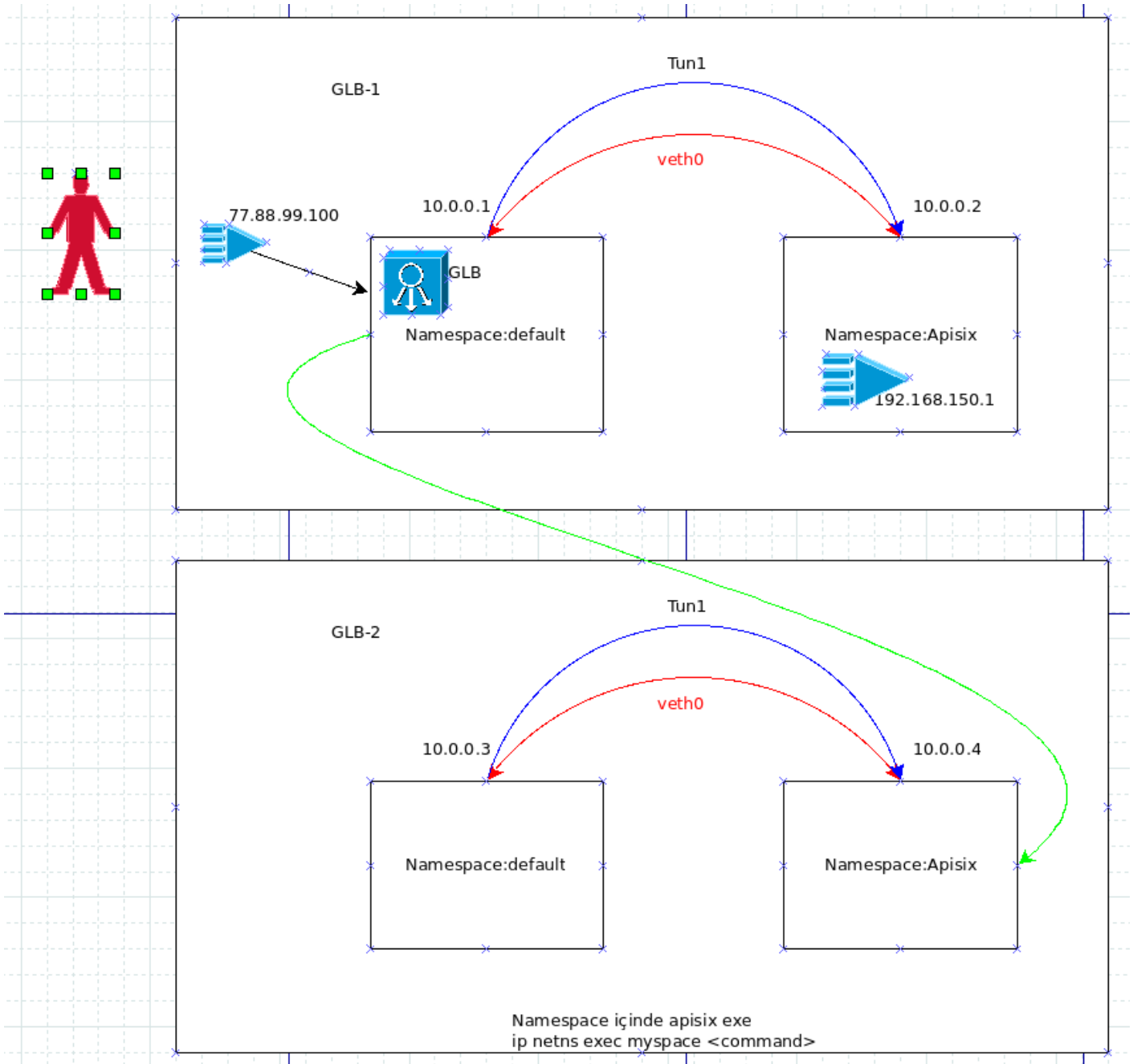
Dizayn

Glb, alıřmak ve second chance iřini yaptırabilmek adına gue t neli ve diğ r uta bu t nelden gelen gue headerlarını okuyabilmek iin bir kernel mod l  kullanır. Bu sayede gue headerdaki second chance verisini mod l okur, offline endpointten cevap alamazsa 2nd chance destinasyona yollar.

Sonuç

Olmadı, gue headerlarından destinasyon bilgisi  len kernel mod l  olan, glb_redirect namespace aware bir mod l olmadığından, alıřmadı. Bu tunel  zerinden healthcheck ve http requestlerin iřlenmemesine ve sonuta paketlerin drop edilmesine sebep oldu.

Ařağıda high level  rnek planı bulabiliriz.



namespace ler arası link için veth lerin oluşturulması ve default namespace için iplendirme.

```
ip link add veth0 type veth peer name veth1
ip addr add 10.10.10.1/30 dev veth0
ip link set up dev veth0
```

apisix'in ve kernel modülünün çalışacağı namespace in oluşturulması, veth pairin iplendirilmesi ve rotalandırma

```
ip netns add apisix
ip link set veth1 netns apisix
ip netns exec apisix ifconfig veth1 10.10.10.2 netmask 255.255.255.252 up
ip netns exec apisix ip r add default via 10.10.10.1
```

FouGue tünelinin process edilebilmesi için namespace te modüle redirect

```
ip netns exec apisix iptables -A INPUT -p udp -m udp --dport 19523 -j GLBREDIRECT
```

Default namespace ten apisix namespace e fougue tüneli kurulumu ve up edilmesi

```
modprobe fou  
ip fou add port 19523 gue  
ip link add name tun1 type ipip remote 10.10.10.2 local 10.10.10.1 ttl 225 encap gue encap-sport auto encap-  
dport 19523  
ip link set up dev tun1
```

apisix namespace te http listener

```
ip netns exec apisix nc -l 80 < index.html
```

[Hızlı bir demo için youtube'dan izleyebilirsiniz](#)

Temel Nginx konfigürasyonu

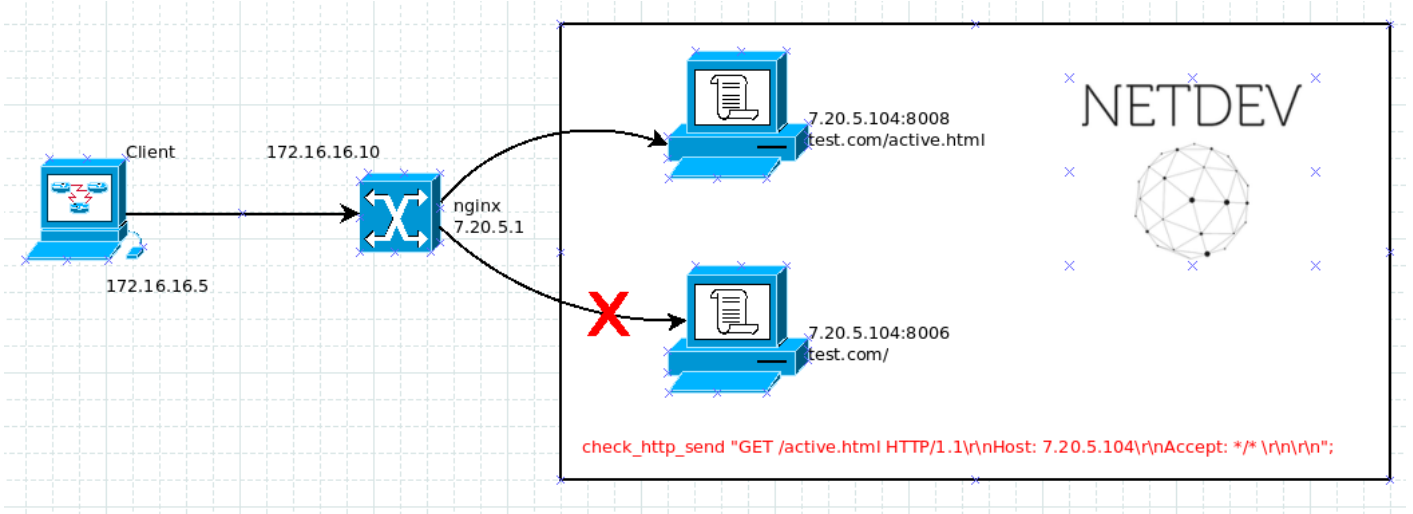
Aşağıda temel konfigürasyon dosyası içeriğini bulabilirsiniz.

```
server {
    listen 80 default_server;
        listen [::]:80 default_server ipv6only=on;
    root /var/www/default;
    index index.html index.htm;
    # Make site accessible from http://localhost/
    server_name localhost;
    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
        # Uncomment to enable naxsi on this location
        # include /etc/nginx/naxsi.rules
    }
    listen 443;
        server_name somecustomer.com.tr www.somecustomer.com.tr somecustomer.com
www.somecustomer.com;
    root /var/www/default;
    index index.html index.htm;
    access_log /var/log/nginx/somecustomer-access.log-ssl-access.log;
    error_log /var/log/nginx/somecustomer-access.log-ssl-error.log;
    ssl on;
    ssl_certificate /etc/keys/somecustomer/www.somecustomer.com.tr.chained.crt;
    ssl_certificate_key /etc/keys/somecustomer/www.somecustomer.com.tr.key;
    ssl_session_timeout 5m;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers "HIGH:!aNULL:!MD5 or HIGH:!aNULL:!MD5:!3DES";
    ssl_prefer_server_ciphers on;
    if ($http_user_agent ~ "Windows 95|Windows
98|biz360.com|xpymep|TurnitinBot|sindice|Purebot|libwww-perl") {
        return 403;
        break;
    }
}
```

```
location / {  
    proxy_redirect off;  
    proxy_set_header X-Real-IP $remote_addr;  
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;  
    proxy_pass http://iis;  
    proxy_set_header Host $http_host;  
    access_log    /var/log/nginx/somecustomer-access.log;  
    error_log     /var/log/nginx/somecustomer-error.log;  
    proxy_set_header  X-Forwarded-Port 443;  
}  
location /blog/ {  
    rewrite ^ http://$host$request_uri? permanent;  
}
```

```
}
```

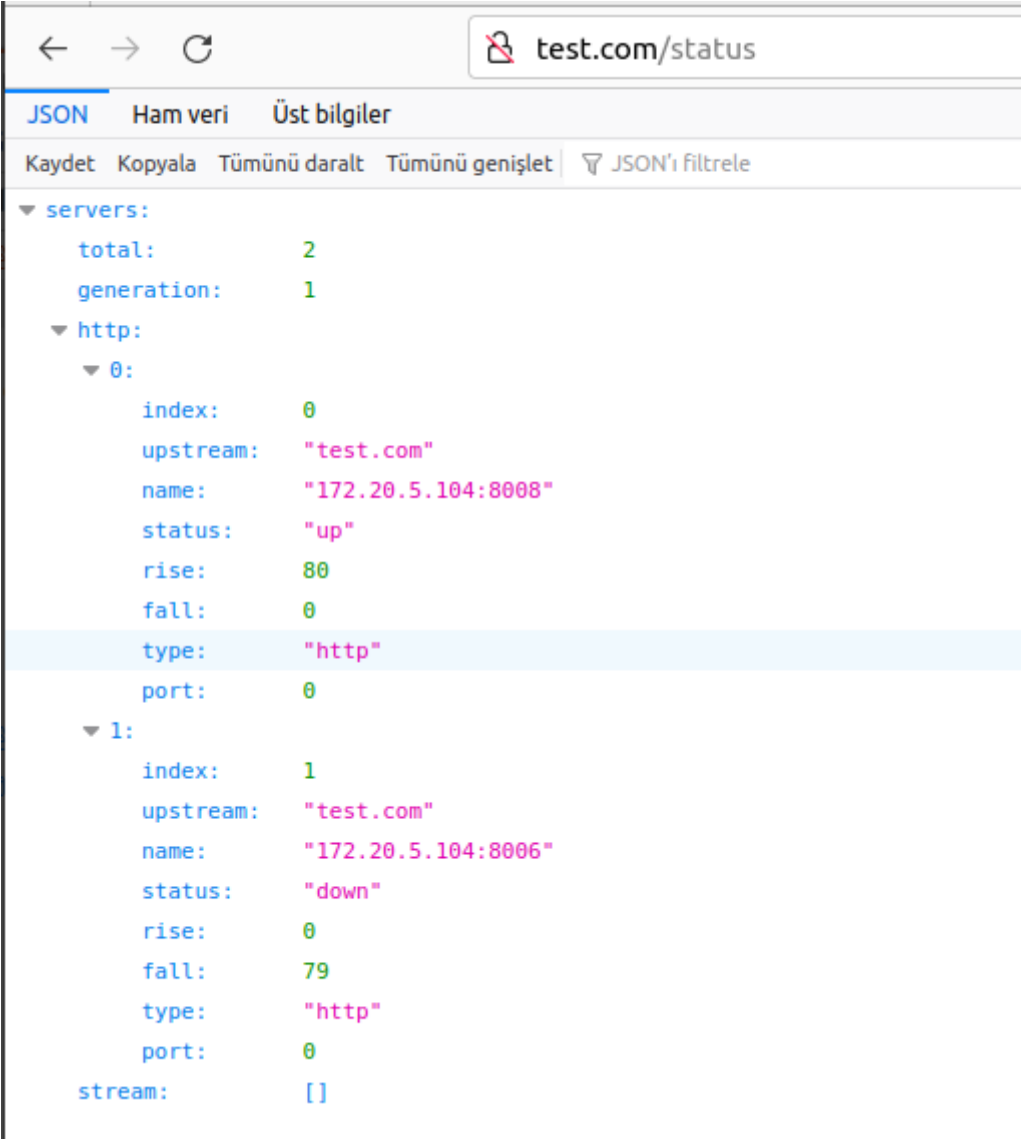

Http sunucu kontrol modülü ile NGINX kurulum ve konfigürasyonu



Bu yazıda, nginx i sunuculara aktif health check yollayacak şekilde derleyip kuruyoruz, sistem fedora36 rolling.

Config ve binary dosya /usr/local/nginx altında oluşacak dolayısı ile aşağıdaki bütün adımlar dosyaların bu klasörde biriktiğini kabul eder.

kurulum sonunda aşağıdaki gibi bir "status page" de elde ederiz



İşletim sisteminin eksiklerini yükle

eksik olan işletim sistemi paketlerini yüklüyoruz.

```
sudo yum install epel-release  
sudo yum install git  
sudo yum group install "C Development Tools and Libraries" "Development Tools"
```

derleme için eksikleri kütüphaneleri yükle

Aşağıdaki kütüphaneler fedora36 da yüklü gelmiyor nginx için yüklü olmaları gerekiyor.

```
sudo yum install pcre-devel  
sudo yum install zlib-devel  
sudo yum install openssl-devel
```

Uyumlu nginx ve http_health_chek modülünü klonla

Sadece aşağıdaki sürümü test ettim, daha yeni sürümler için çalışma yapmak gerekir.

```
git clone https://github.com/nginx/nginx.git  
git clone https://github.com/zhouchangxun/nginx_healthcheck_module.git  
git checkout branches/stable-1.12
```

Derleme

Derleme aşaması burada başlıyor

```
git apply ../ngx_healthcheck_module/nginx_healthcheck_for_nginx_1.12+.patch  
./auto/configure --with-stream --add-module=../ngx_healthcheck_module/ --with-http_ssl_module  
sudo make && sudo make install  
cd /usr/local/nginx/
```

systemd

lokasyon olarak kayıt edilecek yer

```
/usr/lib/systemd/system/nginx.service
```

Unit file içeriği ;

[Unit]

Description=The NGINX HTTP and reverse proxy server

After=syslog.target network-online.target remote-fs.target nss-lookup.target

Wants=network-online.target

[Service]

Type=forking

#PIDFile=/usr/local/nginx//nginx.pid

GuessMainPID=1

ExecStartPre=/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf -t

ExecStart=/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf

ExecReload=/usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf -s reload

ExecStop=/bin/kill -s QUIT \$MAINPID

PrivateTmp=true

User=root

[Install]

WantedBy=multi-user.target

Servisleri başlatalım

```
sudo systemctl daemon-reload
```

```
sudo systemctl status nginx
```

Standart nginx conf dosyası

lokasyonu :

```
/usr/local/nginx/conf/nginx.conf
```

içeriği ;

```
user root root;
```

```
worker_processes 1;
```

```
error_log logs/error.log info;
```

```
#pid logs/nginx.pid;
```

```
events {  
    worker_connections 1024;  
}  
  
http {  
    include /usr/local/nginx/conf/conf.d/*.conf;  
}  
  
stream {  
  
}
```

Load balance ile ilgili config

Lokasyonu :

```
/usr/local/nginx/conf/conf.d
```

Dosyanın içeriği:

```
Server {  
    location /status {  
        healthcheck_status json;  
    }  
    location / {  
        proxy_pass http://test.com;  
    }  
}  
  
upstream test.com {  
    # simple round-robin  
    server 7.20.5.104:8008;  
    server 7.20.5.104:8006;  
  
    check interval=3000 rise=2 fall=5 timeout=500 type=http;  
    check_http_send "GET /active.html HTTP/1.1\r\nHost: 172.20.5.104\r\nAccept: */* \r\n\r\n";  
    check_http_expect_alive http_2xx http_3xx;
```

```
}
```

Kaynak

https://github.com/zhouchangxun/ngx_healthcheck_module#installation

Debug custom rules using http headers with haproxy

Step 1: Define a Custom Header

You can use HAProxy's `http-request set-header` directive to add a custom header to requests passing through HAProxy. This header can contain information like a rule ID or a Lua snippet identifier.

in the end you should have something like this

1	Client		HAProxy Frontend		Content Switching Rules		Backend Servers		Client Response
2	+-----+		+-----+		+-----+		+-----+		+-----+
3		--->	Frontend	--->	- if path_beg /api/v1		Backend A Backend B		
4	Client				use_backend backend_a		(api/v1) (api/v2)		Client
5					set-header X-Rule-ID		+-----+		Response
6	+-----+				rule_v1		+-----+		+-----+
7					- if path_beg /api/v2		+-----+		
8					use_backend backend_b		Backend C More		
9					set-header X-Rule-ID		(default) Backends		
10					rule_v2		+-----+		
11					- default				
12					use_backend backend_c				
13					set-header X-Rule-ID				
14					rule_default				
15					+-----+				
16									

Step 2: Log the Custom Header

Configure the logging format to include the custom header, so it gets logged.

Example Configuration

Here's an example configuration that demonstrates how to set a custom header and log it.

1. Define the custom header in the frontend or backend section

```
frontend http_in
    bind *:80
    mode http
```

```
# Insert custom header with rule ID
http-request set-header X-Rule-ID %[unique-id]

default_backend servers

backend servers
    mode http

server server1 192.168.1.100:80 check
```

In this example, the header `X-Rule-ID` is added to each request with a unique ID (you can customize the value as needed).

2. Customize the log format to include the custom header

```
global
    log 127.0.0.1 local0

defaults
    log global
    option httplog
    log-format "%ci:%cp [%t] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Tt %ST %B %CC %CS %tsc %ac/%fc/%bc/%sc/%rc %sq/%bq %hr %hs %{+Q}r %ST %B %H %{+Q}r %hr %hs %{+Q}r %ht(X-Rule-ID)"

frontend http_in
    bind *:80
    mode http

    # Insert custom header with rule ID
    http-request set-header X-Rule-ID %[unique-id]

    default_backend servers

backend servers
    mode http

server server1 192.168.1.100:80 check
```

In the log format, `%ht(X-Rule-ID)` is used to log the value of the `X-Rule-ID` header.

Using Lua to Set Headers

If you need more complex logic for setting the header, you can use a Lua script:

1. Enable Lua support in HAProxy

Ensure your HAProxy is built with Lua support. You can check this by running `haproxy -vv` and looking for `Built with Lua`.

2. Create a Lua script (`/etc/haproxy/lua/add_header.lua`)

```
core.register_action("set_rule_id", { "http-req" }, function(txn)
    txn.http.req_set_header("X-Rule-ID", "your_rule_id_or_identifier")
end)
```

3. Update the HAProxy configuration to use the Lua script

```
global
    lua-load /etc/haproxy/lua/add_header.lua
    log 127.0.0.1 local0

defaults
    log global
    option httplog
    log-format "%ci:%cp [%t] %ft %b/%s %TR/%Tw/%Tc/%Tr/%Tt %ST %B %CC %CS %tsc %ac/%fc/%bc/%sc/%rc %sq/%bq %hr %hs %{+Q}r %ST %B %H %{+Q}r %hr %hs %{+Q}r %ht(X-Rule-ID)"

frontend http_in
    bind *:80
    mode http

    # Call Lua script to set the header
    http-request lua.set_rule_id

    default_backend servers

backend servers
    mode http

    server server1 192.168.1.100:80 check
```

In this configuration:

- The Lua script sets the `X-Rule-ID` header.
- The log format includes the custom header.

Sample access.log output

There are rule identifiers at the end of every line like `rule_v2` or `rule_v1`

```
192.168.0.1:54321 [11/Jun/2024:15:23:45.123] http_in backend_a/server1 0/0/2/3/5 200 1234 - - ---- 2/1/0/0/0
0/0 "GET /api/v1/resource HTTP/1.1" 200 1234 - - ---- %Qr %hr %hs %Qr rule_v1
192.168.0.2:54322 [11/Jun/2024:15:23:46.456] http_in backend_b/server2 0/0/3/4/7 200 2345 - - ---- 2/1/0/0/0
0/0 "GET /api/v2/resource HTTP/1.1" 200 2345 - - ---- %Qr %hr %hs %Qr rule_v2
192.168.0.3:54323 [11/Jun/2024:15:23:47.789] http_in backend_c/server3 0/0/1/2/3 404 345 - - ---- 2/1/0/0/0 0/0
"GET /other/resource HTTP/1.1" 404 345 - - ---- %Qr %hr %hs %Qr rule_default
```

Summary

By configuring HAProxy to set custom headers and customizing the log format to include these headers, you can effectively log custom identifiers, rule IDs, or other information that may be critical for your debugging and monitoring purposes. This setup provides a powerful way to gain insights into how requests are being processed and routed through your HAProxy instance.