[debian, debian 12, dns]

PowerDNS AIO

В цій статті опишу встановлення комплексу для обслуговування DNS, в неї входить:

- Підготовка - базове налаштування системи та встановлення необхідних компонентів
- PowerDNS-Recursor - для обслуговування DNS-запитів клієнтів. Відповідає за рекурсивне вирішення DNS-запитів
- PowerDNS-Authoritative - для обслуговування DNS-запитів регуляторів та тримання власних DNS-зон. Відповідає за авторитарне вирішення DNS-запитів
- DNSDist - високопродуктивний балансувальник DNS-запитів (теж від команди PowerDNS). Забезпечує балансування навантаження та захист від DDoS-атак
- PowerDNS-Admin - веб-інтерфейс для керування PowerDNS-Authoritative
- Додаткові можливості - Додаткові можливості для покращення продуктивності та функціоналу
  - DNSDist SNMP - для моніторингу станів серверів
  - Чорний список DNSDist - для блокування шкідливого трафіку
- Нотатки - додаткові посилання та інформація

Підготовка

В репозиторіях Debian, зазвичай застарілі версії, тому заглянемо на сайт https://repo.powerdns.com/

Там знаходимо стабільні версії програм і додамо їх до списку репозиторіїв

```
cat <<EOT > /etc/apt/sources.list.d/powerdns.list
deb [signed-by=/etc/apt/keyrings/powerdns-pub.asc] http://repo.powerdns.com/debian bookworm-dnsdist-19 main
deb [signed-by=/etc/apt/keyrings/powerdns-pub.asc] http://repo.powerdns.com/debian bookworm-auth-49 main
deb [signed-by=/etc/apt/keyrings/powerdns-pub.asc] http://repo.powerdns.com/debian bookworm-rec-52 main
EOT
```

Встиновимо сертифікат підпису для репозиторіїв stable

```
install -d /etc/apt/keyrings; curl https://repo.powerdns.com/FD380FBB-pub.asc | sudo tee
/etc/apt/keyrings/powerdns-pub.asc
```

Налаштуємо пріоритет для репозиторіїв

```
cat <<EOT > /etc/apt/preferences.d/powerdns
Package: *
Pin: origin repo.powerdns.com
Pin-Priority: 600
EOT
```

PowerDNS Recursor

PowerDNS Recursor - це рекурсивний DNS-сервер, який відповідає за вирішення DNS-запитів клієнтів. Основні функції:

- Рекурсивне вирішення DNS-запитів
- Кешування відповідей
- Фільтрація запитів
- Підтримка DNSSEC

Встановимо пакет pdns-recursor і збережемо оригінальні конфігураційні файли

```
apt-get update
apt-get install -y pdns-recursor
cp -r /etc/powerdns/ /etc/powerdns.orig/
```

Налаштуємо адресу та порт для прослуховування для версій від 5.0

```
cat <<EOT > /etc/powerdns/recursor.d/00_incoming.yml
incoming:
  listen: !override
  - 127.0.0.1
  - 172.16.0.16
  port: 53
EOT
```

Налаштуємо адресу та порт для прослуховування для версій до 5.0

```
cat <<EOT > /etc/powerdns/recursor.d/recursor.local.conf
local-address=0.0.0.0
local-port=53
EOT
```

Налаштуємо дозволені мережі для прослуховування для версій від 5.0

```
cat <<EOT > /etc/powerdns/recursor.d/01_allow-from.yml
incoming:
  allow_from: !override
  - 0.0.0.0/0 # Allow from any
  - 127.0.0.1/8 # Allow from loopback
  - 193.0.0.1/8 # Allow from arpa (PTR)
  - 192.168.0.1/16 # Allow from local network
EOT
```

Налаштуємо дозволені мережі для прослуховування для версій до 5.0

```
cat <<EOT > /etc/powerdns/recursor.d/allow-from.conf
# !!! Першою лінією має бути запис "allow-from=[ip/mask]", будь-яка інша лінія як "allow-from+=[ip/mask]" (WITH "+" before
"=") !!!
allow-from=0.0.0.0/0 # Allow from any
```

```
allow-from+=127.0.0.1/8 # Allow from loopback
allow-from+=193.0.0.1/8 # Allow from arpa (PTR)
allow-from+=192.168.0.1/16 # Allow from arpa (PTR)
EOT
```

Далі необхідно налаштувати переадресацію запитів DNS на авторитарні сервери, які безпосередньо тримають зони доменів.

Тут можна піти двома шляхами на вибір:

- Перший і правильніший - налаштувати зв'язок з кореневими DNS серверами

```
cd /etc/powerdns/
wget ftp://ftp.rs.internic.net/domain/root.zone.gz && gunzip root.zone.gz
systemctl disable systemd-resolved
systemctl stop systemd-resolved
ls -lh /etc/resolv.conf
mv /etc/resolv.conf /etc/resolv.old.conf
echo "nameserver 127.0.0.1" > /etc/resolv.conf
echo "hint-file=/etc/powerdns/root.zone" >> /etc/powerdns/recursor.conf
systemctl restart pdns-recursor
```

- Другий простіший, але менш продуктивний - переадресувати запити на інші DNS рекурсори, наприклад Cloudflare та Google

для версій від 5.0

```
cat <<EOT > /etc/powerdns/recursor.d/recursor.local.conf
recursor:
  forward_zones_recurse: !override
  - zone: .
    recurse: true
    forwarders:
    - 1.1.1.1
    - 1.0.0.1
    - 8.8.8.8
    - 8.8.4.4
```

```
EOT
```

для версій до 5.0

```
echo "forward-zones-recurse=.=1.1.1.1;1.0.0.1;8.8.8.8;8.8.4.4;" > /etc/powerdns/recursor.d/forward-zones-recurse.conf
```

```
chown -R pdns:root /etc/powerdns/
systemctl restart pdns-recursor
netstat -tap | grep pdns
```

```
dig mydomain.local @127.0.0.1
dig @127.0.0.1
```

Install PowerDNS Server

PowerDNS Server (Authoritative) - це авторитарний DNS-сервер, який відповідає за обслуговування власних DNS-зон. Основні функції:

- Обслуговування авторитарних DNS-зон
- Підтримка різних бекендів (MySQL, PostgreSQL, SQLite)
- Підтримка DNSSEC
- API для управління зонами

Встановимо пакети pdns-server, pdns-backend-mysql та збережемо оригінальні конфігураційні файли

```
apt-get update
apt-get install -y pdns-server pdns-backend-mysql
cp -r /etc/powerdns/ /etc/powerdns.orig/
```

Налаштуємо конфігурацію pdns.conf

```
cat <<EOT > /etc/powerdns/pdns.conf
setgid=pdns
setuid=pdns
include-dir=/etc/powerdns/pdns.d
launch=
EOT
```

prepeare mysql db

Підключимося до MySQL сервера

```
mysql -u root -p
```

Створимо базу даних powerdns та користувача powerdns з паролем strongpassword (замініть на власний пароль)

```
CREATE DATABASE powerdns;
GRANT ALL ON powerdns.* TO 'powerdns'@'localhost' IDENTIFIED BY 'strongpassword';
FLUSH PRIVILEGES;
quit;
```

Якщо це нове встановлення, то створимо структуру бази даних

```
mysql powerdns < /usr/share/pdns-backend-mysql/schema/schema.mysql.sql
```

Інакше, відновимо структуру з вашої резервної копії

```
mysql powerdns < /tmp/backup_powerdns.sql
```

Після відновлення з резервної копії, перевіряємо наявність змін структури

```
ls /usr/share/pdns-backend-mysql/schema/ | grep -E -i '^[0-9]'
3.4.0_to_4.1.0_schema.mysql.sql
4.1.0_to_4.2.0_schema.mysql.sql
4.2.0_to_4.3.0_schema.mysql.sql
4.3.0_to_4.7.0_schema.mysql.sql
```

Якщо вони є, застосовуємо всі почерзі, починаючи з версії з резервної копії, наприклад з 4.1.0

```
mysql powerdns < /usr/share/pdns-backend-mysql/schema/4.1.0_to_4.2.0_schema.mysql.sql
mysql powerdns < /usr/share/pdns-backend-mysql/schema/4.2.0_to_4.3.0_schema.mysql.sql
mysql powerdns < /usr/share/pdns-backend-mysql/schema/4.3.0_to_4.7.0_schema.mysql.sql
```

Налаштуємо конфігурацію pdns.local.gmysql.conf для підключення до нашої базі даних

```
cat <<EOT > /etc/powerdns/pdns.d/pdns.local.gmysql.conf
# MySQL Configuration
# Launch gmysql backend
launch+=gmysql
# gmysql parameters
gmysql-host=127.0.0.1
gmysql-port=3306
gmysql-dbname=powerdns
gmysql-user=powerdns
gmysql-password=strongpassword
gmysql-dnssec=yes
# gmysql-socket=
EOT
```

Перезапустимо сервіси та перевіримо статус

```
systemctl restart pdns
netstat -tap | grep pdns
```

Перевіримо роботу сервера

```
dig mydomain.local @127.0.0.1
dig @127.0.0.1
```

DNSDist

Dnsdist - це високопродуктивний DNS-, DoS- та abuse балансувальник.

Основне його завдання полягає у маршрутизації трафіку на найкращий сервер, що забезпечує максимальну продуктивність для дозволених користувачів, у той час як відбувається шунтування або блокування шкідливого трафіку.

Має величезну кількість корисних функцій:

- Фільтрувати трафік (з ядра)
- Перевіряти прямий трафік з консолі
- Затримувати та обмежувати швидкість поганих запитів
- Інтелектуальне балансування навантаження
- Обмеження QPS та ін.

Встановимо пакети dnsdist, dnstop та збережемо оригінальні конфігураційні файли

```
apt-get update -y
apt-get install -y dnsdist dnstop
cp -r /etc/dnsdist/ /etc/dnsdist.orig/
```

Створімо конфігураційний файл

```
cat <<EOT > /etc/dnsdist/dnsdist.conf
```

```
setLocal('127.0.0.1')
addLocal('ANOTHER_IP')
addLocal('ANOTHER_IPV6_IP')
setACL({'0.0.0.0/0'}) -- Allow all IPs access
newServer({address='127.0.0.1:5300', pool='auth'})
newServer({address='127.0.0.1:5301', pool='recursor'})
recursive_ips = newNMG()
recursive_ips:addMask('127.0.0.1/32')
recursive_ips:addMask('192.168.2.0/23')
addAction(NetmaskGroupRule(recursive_ips), PoolAction('recursor'))
addAction(AllRule(), PoolAction('auth'))
EOT
```

Згенеруємо пароль для консолі

```
pip  install  dnsdist_console

python3 -c "from dnsdist_console import HashPassword as H;print(H().generate(\"mysupersecret\"))"
  $scrypt$ln=10,p=1,r=8$rY9YB+QnTOkxKOBlNUUYaw==$4C4Hm5IFiOTluLkjGtPMl4FtYQIwJvSA/eb7uqAlFg4=
```

Якщо хочемо відкрити рекурсію для всіх, то прибираємо всі правила і додаємо recursive_ips:addMask('0.0.0.0/0').

УВАГА! У такому режимі є можливість DDoS-атаки!

Змінимо конфігурацію прослуховування для рекурсора версії від 5.0

```
cat <<EOT > /etc/powerdns/recursor.d/00_incoming.yml
incoming:
  listen: !override
  - 127.0.0.1
  - 172.16.0.16
```

```
  port: 5301
EOT
```

Змінимо конфігурацію прослуховування для рекурсора версії до 5.0

```
cat <<EOT > /etc/powerdns/recursor.d/recursor.local.conf
local-address=127.0.0.1
local-port=5301
EOT
```

Змінимо конфігурацію прослуховування для авторитарного сервера:

```
cat <<EOT > /etc/powerdns/pdns.d/pdns.local.conf
local-address=127.0.0.1
local-port=5300
EOT
```

Перезапустимо сервіси та перевіримо статус

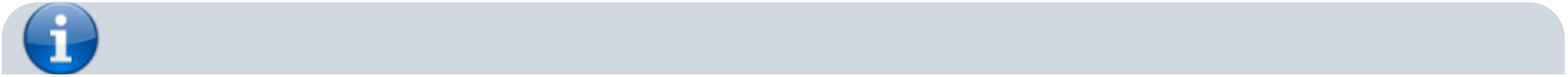
```
service pdns-recursor restart
service pdns restart
```

https://stat.ripe.net/widget/dns-check

Додаємо сервіс dnsdist в автозавантаження та перезапускаємо:

```
systemctl enable dnsdist
systemctl restart dnsdist
```

Панель адміністрування PowerDNS-Admin



PowerDNS-Admin - це веб-інтерфейс для управління PowerDNS:

- Управління DNS-зонами
- Управління записами
- Управління користувачами
- Моніторинг стану
- API для автоматизації

https://computingforgeeks.com/install-powerdns-and-powerdns-admin-on-debian/

https://github.com/PowerDNS-Admin/PowerDNS-Admin/wiki/Running-PowerDNS-Admin-on-Ubuntu-or-Debian

Встановимо необхідні пакети

```
apt install python3-dev python3-venv apt install libpq-devgit default-libmysqlclient-dev libsasl2-dev libldap2-dev libssl-dev libxml2-dev libxslt1-dev libxmlsec1-dev libffi-dev pkg-config apt-transport-https virtualenv build-essential curl jq
```

Встановимо Node.js

```
curl -sL https://deb.nodesource.com/setup_20.x | bash -
apt-get update && apt install -y nodejs
```

Встановимо Yarn

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | gpg --dearmor | sudo tee /usr/share/keyrings/yarnkey.gpg >/dev/null
echo "deb [signed-by=/usr/share/keyrings/yarnkey.gpg] https://dl.yarnpkg.com/debian stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
apt-get update && apt-get install yarn
```

Підключимося до MySQL сервера

```
mysql -u root -p
```

Створимо базу даних для панелі адміністрування

```
CREATE DATABASE poweradmin CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
GRANT ALL PRIVILEGES ON poweradmin.* TO 'pdnsadmin'@'%' IDENTIFIED BY 'p4ssw0rd';
FLUSH PRIVILEGES;
quit
```

Склонуємо репозиторій панелі адміністрування

```
git clone https://github.com/PowerDNS-Admin/PowerDNS-Admin.git /usr/local/powerdns-admin
```

УВАГА! Якщо не перейти на конкретну версію з доступних тегів, то буде використана остання версія розробки з гілки master, яка не є стабільною.

Перейдемо до директорії панелі адміністрування та створимо віртуальне середовище

```
cd /usr/local/powerdns-admin
```

Переглянемо список доступних тегів

```
git config --global --add safe.directory /usr/local/powerdns-admin
git tag
```

вивід

```
...
v0.4.0
v0.4.1
v0.4.2
```

Отримуємо код останньої версії

```
git checkout v0.4.2
```

Створюємо віртуальне середовище та активуємо його

```
python3 -m venv .venv --prompt VirtualEnv
source .venv/bin/activate
python -V
```

Встановимо необхідні залежності

```
pip install --upgrade pip
pip install -r requirements.txt
```

Створюємо конфігураційний файл

```
cat <<EOT > /usr/local/powerdns-admin/configs/production.py
import os
import urllib.parse
basedir = os.path.abspath(os.path.dirname(__file__))

### BASIC APP CONFIG
SALT = '$2b$12$yLUMTIfl21FKJQpTkRQXCu'
SECRET_KEY = 'e951e5a1f4b94151b360f47edf596dd2'
BIND_ADDRESS = '0.0.0.0'
PORT = 9191
OFFLINE_MODE = False

### DATABASE CONFIG
SQLA_DB_USER = 'pdnsadmin'
SQLA_DB_PASSWORD = 'p4ssw0rd'
SQLA_DB_HOST = '127.0.0.1'
SQLA_DB_NAME = 'poweradmin'
SQLALCHEMY_TRACK_MODIFICATIONS = True
```

```
### DATABASE - MySQL
SQLALCHEMY_DATABASE_URI = 'mysql://{}:{}@{}/{}'.format(
    urllib.parse.quote_plus(SQLA_DB_USER),
    urllib.parse.quote_plus(SQLA_DB_PASSWORD),
    SQLA_DB_HOST,
    SQLA_DB_NAME
)

# Зберігаємо сесію в файлі якщо версія  python  більше рівній 3.11 інакше  'sqlalchemy'
SESSION_TYPE = 'filesystem' if sys.version_info >= (3, 11) else 'sqlalchemy'

### DATABASE - SQLite
#SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(basedir, 'pdns.db')
EOT
```

Додаємо правило для порту 9191 в nftables. Не забудьте поправити правило в /etc/nftables.conf

```
nft add rule ip filter input ct state new tcp dport 9191 counter accept comment "PDNS_ADMIN"
```

Експортуємо змінні середовища

```
export FLASK_CONF=../configs/production.py
export FLASK_APP=powerdnsadmin/__init__.py
```

Виконуємо команду для оновлення бази даних

```
flask db upgrade
```

Виконуємо команду для побудови ресурсів

```
yarn install --pure-lockfile
flask assets build
```

Змінюємо власника директорії

```
chown -R www-data:root /usr/local/powerdns-admin
```

Створюємо файл сервісу

```
cat <<EOF > /etc/systemd/system/powerdns-admin.service
[Unit]
Description=PowerDNS-Admin
Wants=network.target
After=network.target

[Service]
PIDFile=/run/powerdns-admin/pid
User=www-data
Group=root
WorkingDirectory=/usr/local/powerdns-admin
ExecStart=/usr/local/powerdns-admin/.venv/bin/gunicorn --pid /run/powerdns-admin/pid --workers 2 --bind
'0.0.0.0:9191' 'powerdnsadmin:create_app()'
ExecReload=/bin/kill -s HUP $MAINPID
ExecStop=/bin/kill -s TERM $MAINPID
PrivateTmp=true
Restart=on-failure
RestartSec=10
StartLimitInterval=0

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

Додаємо змінні середовища

```
mkdir -p /etc/systemd/system/powerdns-admin.service.d/
tee /etc/systemd/system/powerdns-admin.service.d/override.conf<<EOF
```

```
[Service]
Environment="FLASK_CONF=../configs/production.py"
EOF
```

Створюємо директорію для зберігання PID та додаємо правило для неї

```
echo 'd /run/powerdns-admin 0755 www-data root -' > /etc/tmpfiles.d/powerdns-admin.conf
mkdir -p /run/powerdns-admin
chown -R www-data:root /run/powerdns-admin
```

Перезавантажуємо сервіси

```
systemctl daemon-reload
systemctl enable powerdns-admin
systemctl restart powerdns-admin
systemctl status powerdns-admin
```

Додаємо конфігурацію для API до авторитарного сервера

```
cat <<EOT > /etc/powerdns/pdns.d/api.conf
webserver-port=8081
api=yes
api-key=changemechangemechangeme
EOT

chown -R pdns:root /etc/powerdns/
```

Перезавантажуємо сервіси

```
systemctl restart pdns
```

Переходимо на http://your-host-ip:9191/ та створюємо користувача

Натисніть кнопку “Create an account” та зареєструйте користувача. Перший користувач буде в ролі Адміністратора.

Надайте інформацію про підключення до PowerDNS API перед тим, як PowerDNS-Admin зможе запитувати дані вашого PowerDNS. Це робиться в розділі Settings > PDNS

Додаткові можливості

Чорний список DNSDist

Чорний список DNSDist дозволяє блокувати небажані домени з різних причин:

- Блокування шкідливих доменів
- Блокування реклами
- Блокування фішингових сайтів
- Блокування піратського контенту

https://github.com/enilfodne/dnsdist-adblock/blob/master/dagg/dagg.lua

Часто буває необхідність з різних причин блокувати небажані домени, ось мій варіант

Створюємо директорію для додаткових конфігурацій

```
mkdir -p /etc/dnsdist/conf.d/
```

Для активації читання конфігурацій з /etc/dnsdist/conf.d/ підправимо

```
nano /etc/dnsdist/dnsdist.conf
```

додавши після рядків з “newServer()”

```
includeDirectory("/etc/dnsdist/conf.d")
```

Створюємо файл конфігурації

```
nano /etc/dnsdist/conf.d/dagg.conf
```

```
-- https://github.com/enilfodne/dnsdist-adblock/blob/master/dagg/dagg.lua
-- add "includeDirectory("/etc/dnsdist/conf.d")" to /etc/dnsdist/dnsdist.conf
-- put this file into "/etc/dnsdist/conf.d"
Dagg = {
        -- config
        config = {
        blacklistip = "127.0.0.127",
                actionlog = {
                        path = "/var/log/dnsdist_blocked.log",
                },
                blocklist = {
                        path = "/etc/dnsdist/black.list",
                },
                reload = {
                        target = "reload.blacklist.local.",
                },
                unload = {
                        target = "unload.blacklist.local.",
                },
        },
        -- table storing the domains that need to be blocked
        table = {
                -- only used for wildcard domains
                smn = newSuffixMatchNode(),
                -- default - fast string comparison
                str = {},
        },
}
```

```
-- read all the domains in a set
function DaggLoadDomainsFromFile(file)
        local f = io.open(file, "rb")

        -- verify that the file exists and it is accessible
        if f ~= nil then
                for domain in f:lines() do
            domain = domain:lower()
            domain = domain.."."
            if string.find(domain, "*.") then
                                local suffix = domain:gsub("*.", "")
                                Dagg.table.smn:add(suffix)
                        else
                                Dagg.table.str[domain] = true
                        end
                end

                f:close()
        end
end

-- verbose, but clear
function DaggLoadBlocklist()
        -- no reason, just for clarity
        local file = Dagg.config.blocklist.path
        -- not really necessary, but keep similarity to other versions

        local f = io.open(file, "rb")

        if f ~= nil then
                -- file exists, close and proceed with sed
                f:close()
```

```
                -- it appears that even when using:
                --
                -- 'local var = str2 .. str2'
                --
                -- the variable is not being garbage-collected
                -- and it ends up looking like a memory leak.
                --
                -- let me know if if there's a better way
                -- os.execute("sed '/\\.$/ ! s/$/\\./' -i " .. file)
        else
                errlog "[Dagg] the blocklist file is missing or inaccessible!"
        end

        DaggLoadDomainsFromFile(file)
end

-- clear the table from memory
function DaggClearTable()
        Dagg.table = {
                smn = newSuffixMatchNode(),
                str = {},
        }
end

-- write down a query to the action log
function DaggWriteToActionLog(dq)
        -- write-down the query
        local f = io.open(Dagg.config.actionlog.path, "a")

        if f ~= nil then
                local query_name = dq.qname:toString()
                local remote_addr = dq.remoteaddr:toString()

                local msg = string.format("[%s][%s] %s", os.date("!%Y-%m-%dT%TZ", t), remote_addr, query_name)
```

```
                f:write(msg, "\n")
                f:close()
        end
end

-- main query action
function DaggIsDomainBlocked(dq)
        local qname = dq.qname:toString():lower()

        if (Dagg.table.smn:check(dq.qname) or Dagg.table.str[qname] ) then
        errlog ("[Dagg] "..qname.." in blacklist and redirected to ".. Dagg.config.blacklistip)
                -- set QueryResponse, so the query never goes upstream
                dq.dh:setQR(true)

                -- set a CustomTag
                -- you can optionally set a tag and process
                -- this request with other actions/pools
                -- dq:setTag("Dagg", "true")

                -- WARNING: it (may?) affect(s) performance
                -- DaggWriteToActionLog(dq)

                -- return NXDOMAIN - its fast, but apparently
                -- some apps resort to hard-coded entries if NX
                -- try spoofing in this instance.

                -- return DNSAction.Nxdomain, ""

                -- return Spoof - you can spoof the response
                -- instead of NX, but this may lead to time-outs
        return DNSAction.Spoof, Dagg.config.blacklistip
        end
        return DNSAction.None, ""
```

```
end
addAction(AllRule(), LuaAction(DaggIsDomainBlocked))

-- reload action
function DaggReloadBlocklist(dq)
        infolog "[Dagg] re/loading blocklist..."

        -- prevent the query from going upstream
        dq.dh:setQR(true)

        -- clear
        DaggClearTable()

        -- clean-up
        collectgarbage "collect"

        -- load
        DaggLoadBlocklist()

        -- clean-up
        collectgarbage "collect"

        -- respond with a local address just in case
        return DNSAction.Spoof, "127.0.0.127"
end
addAction(Dagg.config.reload.target, LuaAction(DaggReloadBlocklist))

-- unload action
function DaggUnloadBlocklist(dq)
        infolog "[Dagg] unloading blocklist..."

        -- prevent the query from going upstream
        dq.dh:setQR(true)
```

```
        -- clear
        DaggClearTable()

        -- clean-up
        collectgarbage "collect"

        -- respond with a local address just in case
        return DNSAction.Spoof, "127.0.0.127"
end
addAction(Dagg.config.unload.target, LuaAction(DaggUnloadBlocklist))
```

Створюємо файл для чорного списку доменів

```
nano /etc/dnsdist/black.list
```

```
*.lohotron.shop
porn.xxx
xn--80ayhh.xn--c1avg
```

https://www.punycoder.com/

Додаємо зміни фалу сервісу для затримки завантаження чорного списку

```
mkdir -p /etc/systemd/system/dnsdist.service.d/
cat <<EOF > /etc/systemd/system/dnsdist.service.d/override.conf
[Service]
ExecStartPost=-/usr/bin/sleep 5
ExecStartPost=-/usr/bin/env dig @127.0.0.1 reload.blacklist.local
EOF
```

Перезавантажуємо сервіси

```
systemctl daemon-reload
systemctl enable dnsdist
```

```
systemctl restart dnsdist
journalctl -xe
```

Створюємо скрипт для завантаження чорного списку

```
nano /etc/dnsdist/get_black_list.pl
```

З таким вмістом

```
#!/usr/bin/perl
# ------------------------- Dagg -------------------------
# https://t.me/MrMethod
# --------------------------------------------------------
# ver. 2022.10.06 rev. 654
# --------------------------------------------------------
use strict;
use warnings;

use JSON;
use Net::DNS;
#use Data::Dumper;
use LWP::UserAgent;

my $blacklist_url  = 'http://example.com/blackdns.json';
my $blacklist_file = '/etc/dnsdist/black.list';
my $reload_host    = 'reload.blacklist.local';

my %old_dns = ();
if (open(my $fh, $blacklist_file)) {#'<:encoding(UTF-8)',
    while( my $line = <$fh>)  {
        chomp $line;
        $old_dns{$line} = 1;
    }
    close($fh);
```

```
} else {
    warn "Could not open file '$blacklist_file' $!";
}
#print Dumper \%old_dns;

my $ua = LWP::UserAgent->new(
    ssl_opts => { verify_hostname => 0 },
    protocols_allowed => ['http', 'https'],
    timeout => 60,
);

my $response = $ua->get($blacklist_url);

my %new_dns = ();
if ($response->is_success) {
    #print $response->decoded_content;
    my @new_list = @{decode_json($response->decoded_content)};
    %new_dns = map { $_ => 1 } @new_list;
#    print Dumper \%new_dns;
} elsif ($response->is_error){
    print "Error: $blacklist_url\n";
    print $response->error_as_HTML;
} else {
    die $response->status_line;
}

if ( scalar %new_dns && join( ' ', sort keys %old_dns) ne join( ' ', sort keys %new_dns) ) {
    my $new = join( "\n", sort keys %new_dns);
    #print $new."\n";
    if (open(my $fh, '>', $blacklist_file)) {
        print $fh $new;
        close($fh);

        # Set options in the constructor
```

```
        my $resolver = Net::DNS::Resolver->new(
            nameservers => [ '127.0.0.1' ],
            recurse     => 0,
            debug       => 0,
        );
        print "Have updates for black.list\n";
        my $packet = $resolver->send( $reload_host, 'A' ) if $reload_host;
    } else {
        warn "Could not open file '$blacklist_file' $!";
    }
} else {
    print "black.list already updated\n";
}

1;
```

Дамо дозвіл на виконання скрипту

```
chmod +x /etc/dnsdist/get_black_list.pl
```

Встановлюємо необхідні модулі для perl

```
apt install libperl-dev cpanminus
cpanm -n JSON JSON::PP Net::DNS LWP::UserAgent LWP::Protocol::https
```

Виконуємо скрипт для завантаження чорного списку

```
perl /etc/dnsdist/get_black_list.pl
```

Приклад файлу black.list

```
[
   "*.lohotron.shop",
   "porn.xxx",
```

```
   "xn--80ayhh.xn--c1avg"
]
```

Створимо юніт-файл сервісу завантаження чорного списку

```
cat <<EOT >> /etc/systemd/system/get-blacklist.service
[Unit]
Description=Run get_black_list.pl to update DNS blocklist

[Service]
Type=oneshot
ExecStart=/usr/bin/perl /etc/dnsdist/get_black_list.pl
EOT
```

Створимо юніт-файл таймеру для запуску нашого сервісу кожні 20 хвилин

```
cat <<EOT >> /etc/systemd/system/get-blacklist.timer
[Unit]
Description=Run get_black_list.pl every 20 minutes

[Timer]
OnBootSec=5min
OnUnitActiveSec=20min
Unit=get-blacklist.service

[Install]
WantedBy=timers.target
EOT
```

Додамо в автозавантаження та запустимо наш таймер

```
sudo systemctl daemon-reexec
sudo systemctl daemon-reload
```

```
sudo systemctl enable --now get-blacklist.timer
```

перевіримо лог спрацювання таймера

```
journalctl -u get-blacklist.service
```

DNSDist SNMP

SNMP (Simple Network Management Protocol) дозволяє моніторити стан DNS-серверів:

- Кількість запитів/відповідей
- Затримки обробки запитів
- Стан бекендів
- Використання ресурсів

Додаємо репозиторій non-free в apt

```
apt install software-properties-common
apt-add-repository non-free
```

Додаємо репозиторій non-free в apt (альтернативний спосіб)

```
nano /etc/apt/sources.list
```

Додаємо рядок “non-free” в кінець кожного додати “non-free”.

```
deb http://deb.debian.org/debian/ bookworm main non-free
deb-src http://deb.debian.org/debian/ bookworm main non-free
```

```
deb http://security.debian.org/debian-security bookworm-security main non-free
deb-src http://security.debian.org/debian-security bookworm-security main non-free

# bookworm-updates, to get updates before a point release is made;
# see https://www.debian.org/doc/manuals/debian-reference/ch02.en.html#_updates_and_backports
deb http://deb.debian.org/debian/ bookworm-updates main non-free
deb-src http://deb.debian.org/debian/ bookworm-updates main non-free
```

Встановлюємо необхідні пакунки

```
apt update && apt install -y snmp-mibs-downloader snmp snmpd
```

Коментуємо рядок “mibs :” для snmp-mibs-downloader

```
nano /etc/snmp/snmp.conf
```

Налаштовуємо snmpd agentx для dnsdist

```
nano /etc/snmp/snmpd.conf
```

Правимо конфігурацію snmpd

```
master  agentx
agentxperms 0700 0700 _dnsdist _dnsdist
rocommunity dnsdist42
```

Встановлюємо необхідні mibs

```
wget -P /usr/share/snmp/mibs https://raw.githubusercontent.com/PowerDNS/pdns/master/pdns/dnsdistdist/DNSDIST-
MIB.txt
wget -P /usr/share/snmp/mibs https://mibbrowser.online/mibs/FLOAT-TC-MIB.mib
```

Правимо права до каталогу

```
chmod 775 /var/agentx/
```

Для активації SNMP потрібно в кінець файлу

```
nano /etc/dnsdist/dnsdist.conf
```

додати рядок

```
snmpAgent(true,"/var/agentx/master")
```

Перезавантажуємо сервіси

```
systemctl restart snmpd
systemctl restart dnsdist
journalctl -xe
```

Перевірити працездатність SNMP можна наступними командами

```
snmpwalk -v2c -c dnsdist42 127.0.0.1 .1.3.6.1.4.1.43315
snmpwalk -v2c -m DNSDIST-MIB -c dnsdist42 127.0.0.1 1.3.6.1.4.1.43315
```

Ось вивід останньої команди

```
DNSDIST-MIB::queries.0 = Counter64: 18
DNSDIST-MIB::responses.0 = Counter64: 1
DNSDIST-MIB::servfailResponses.0 = Counter64: 0
DNSDIST-MIB::aclDrops.0 = Counter64: 0
DNSDIST-MIB::ruleDrop.0 = Counter64: 0
DNSDIST-MIB::ruleNXDomain.0 = Counter64: 0
DNSDIST-MIB::ruleRefused.0 = Counter64: 0
DNSDIST-MIB::selfAnswered.0 = Counter64: 17
DNSDIST-MIB::downstreamTimeouts.0 = Counter64: 0
DNSDIST-MIB::downstreamSendErrors.0 = Counter64: 0
```

```
DNSDIST-MIB::truncFailures.0 = Counter64: 0
DNSDIST-MIB::noPolicy.0 = Counter64: 0
DNSDIST-MIB::latency01.0 = Counter64: 17
DNSDIST-MIB::latency110.0 = Counter64: 0
DNSDIST-MIB::latency1050.0 = Counter64: 0
DNSDIST-MIB::latency50100.0 = Counter64: 0
DNSDIST-MIB::latency1001000.0 = Counter64: 1
DNSDIST-MIB::latencySlow.0 = Counter64: 0
DNSDIST-MIB::latencyAVG100.0 = STRING: "2404.139327"
DNSDIST-MIB::latencyAVG1000.0 = STRING: "263.185870"
DNSDIST-MIB::latencyAVG10000.0 = STRING: "26.556655"
DNSDIST-MIB::latencyAVG1000000.0 = STRING: "0.265830"
DNSDIST-MIB::uptime.0 = Counter64: 33051
DNSDIST-MIB::realMemoryUsage.0 = Counter64: 73154560
DNSDIST-MIB::nonCompliantQueries.0 = Counter64: 0
DNSDIST-MIB::nonCompliantResponses.0 = Counter64: 0
DNSDIST-MIB::rdQueries.0 = Counter64: 18
DNSDIST-MIB::emptyQueries.0 = Counter64: 0
DNSDIST-MIB::cacheHits.0 = Counter64: 0
DNSDIST-MIB::cacheMisses.0 = Counter64: 0
DNSDIST-MIB::cpuUserMSec.0 = Counter64: 18324
DNSDIST-MIB::cpuSysMSec.0 = Counter64: 60627
DNSDIST-MIB::fdUsage.0 = Counter64: 97
DNSDIST-MIB::dynBlocked.0 = Counter64: 0
DNSDIST-MIB::dynBlockNMGSize.0 = Counter64: 0
DNSDIST-MIB::ruleServFail.0 = Counter64: 0
DNSDIST-MIB::securityStatus.0 = Counter64: 1
DNSDIST-MIB::specialMemoryUsage.0 = Counter64: 55439360
DNSDIST-MIB::ruleTruncated.0 = Counter64: 0
DNSDIST-MIB::backendName.0 = STRING: 127.0.0.1:5300
DNSDIST-MIB::backendName.1 = STRING: 127.0.0.1:5301
DNSDIST-MIB::backendLatency.0 = Counter64: 0
DNSDIST-MIB::backendLatency.1 = Counter64: 2
DNSDIST-MIB::backendWeight.0 = Counter64: 1
```

```
DNSDIST-MIB::backendWeight.1 = Counter64: 1
DNSDIST-MIB::backendOutstanding.0 = Counter64: 0
DNSDIST-MIB::backendOutstanding.1 = Counter64: 0
DNSDIST-MIB::backendQPSLimit.0 = Counter64: 0
DNSDIST-MIB::backendQPSLimit.1 = Counter64: 0
DNSDIST-MIB::backendReused.0 = Counter64: 0
DNSDIST-MIB::backendReused.1 = Counter64: 0
DNSDIST-MIB::backendState.0 = STRING: up
DNSDIST-MIB::backendState.1 = STRING: up
DNSDIST-MIB::backendAddress.0 = STRING: "127.0.0.1:5300"
DNSDIST-MIB::backendAddress.1 = STRING: "127.0.0.1:5301"
DNSDIST-MIB::backendPools.0 = STRING: auth
DNSDIST-MIB::backendPools.1 = STRING: recursor
DNSDIST-MIB::backendQPS.0 = Counter64: 0
DNSDIST-MIB::backendQPS.1 = Counter64: 0
DNSDIST-MIB::backendQueries.0 = Counter64: 0
DNSDIST-MIB::backendQueries.1 = Counter64: 1
DNSDIST-MIB::backendOrder.0 = Counter64: 1
DNSDIST-MIB::backendOrder.1 = Counter64: 1
```

Для відкриття доступу до SNMPD з мережі потрібно відкрити порт в файрволі

```
nft add rule ip filter input ct state new udp dport 161 counter accept comment "SNMPD"
```

та змінити параметр agentaddress в /etc/snmp/snmpd.conf з 127.0.0.1 на іп адресу, за якою можна звернутись до хоста з мережі, або на 0.0.0.0 для глобального досупу

Нотатки

DnsDist Query cache

Установка DNSCrypt-сервера

https://dnslookup.online/ptr.html

https://repo.powerdns.com/

https://fossies.org/linux/pdns-dnsdist/pdns/dnsdistdist/docs/advanced/snmp.rst

big config

Зареєструвати свій номер (SNMP ENTERPRISE VENDOR NUMBER)

```
/etc/init.d/pdns-recursor restart
rec_control wipe-cache
/etc/init.d/pdns-recursor status
```