PowerDNS DNSDist

Розгортаємо високопродуктивний безкоштовний DNS-сервер в складі авторитарного сервера з доменами користувача, окремим рекурсивним сервером і балансувальником dnsdist

All we do from sudo user!!!

https://repo.powerdns.com/

Install DNSdist

Встановлення та налаштування балансувальника 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.

В репозиторіях є зазвичай, застаріла версія, тому заглянемо на сайт https://repo.powerdns.com/

Там знаходимо стабільну версію програми і застосовуємо зміни в списку репозиторіїв

```
sh -c 'echo "deb [arch=amd64] http://repo.powerdns.com/debian bullseye-dnsdist-17 main" >
/etc/apt/sources.list.d/pdns.list'
```

```
tee /etc/apt/preferences.d/dnsdist<<EOF
Package: dnsdist*
Pin: origin repo.powerdns.com
Pin-Priority: 600
EOF
```

```
curl https://repo.powerdns.com/FD380FBB-pub.asc | sudo gpg --no-default-keyring --keyring gnupg-
ring:/etc/apt/trusted.gpg.d/powerdns.gpg --import
chmod 644 /etc/apt/trusted.gpg.d/*
```

Встановимо пакет dnsdist.

```
apt-get update -y
apt-get install -y dnsdist
```

Відкриваємо конфігураційний файл

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

і наводимо до такого вигляду:

```
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'))
```

```
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!

Підправимо конфіг рекурсора:

```
nano /etc/powerdns/recursor.d/recursor.local.conf
```

```
local-address=127.0.0.1
local-port=5301
```

Підправимо конфіг auth:

```
nano /etc/powerdns/pdns.d/pdns.local.conf
```

```
local-address=127.0.0.1
local-port=5300
```

```
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 start dnsdist
```

DNSDIST BLACK LIST

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

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

```
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")
```

```
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))
```

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

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

https://www.punycoder.com/

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

```
systemctl daemon-reload
systemctl enable dnsdist
systemctl restart dnsdist
journalctl -xe
```

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

```
#!/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;
```

```
cpanm -n JSON JSON::PP Net::DNS LWP::UserAgent LWP::Protocol::https
```

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

```
[
   "*.lohotron.shop",
   "porn.xxx",
   "xn--80ayhh.xn--c1avg"
]
```

SNMP

До основних репозиторіїв необхідно в кінець кожного додати “non-free”, інакше деякі пакунки не буде знайдено для встановлення.

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

```
# deb cdrom:[Debian GNU/Linux 11.4.0 _Bullseye_ - Official amd64 NETINST 20220709-10:31]/ bullseye main

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

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

# bullseye-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/ bullseye-updates main non-free
deb-src http://deb.debian.org/debian/ bullseye-updates main non-free
```

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

comment line “mibs :” for snmp-mibs-downloader

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

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

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

```
cd /usr/share/snmp/mibs
wget https://raw.githubusercontent.com/PowerDNS/pdns/master/pdns/dnsdistdist/DNSDIST-MIB.txt
wget https://www.circitor.fr/Mibs/Mib/F/FLOAT-TC-MIB.mib
```

```
chown -R _dnsdist:root  /etc/dnsdist/
chown -R pdns:root /etc/powerdns/
chmod 775 /var/agentx/
```

Для активації SNMP потрібно в кінець файлу

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

додати рядок

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

```
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

```
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

Установка DNSCrypt-сервера

https://dnslookup.online/ptr.html

https://repo.powerdns.com/

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

/etc/init.d/pdns-recursor restart

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