**Building Your Own DNS Server with DNS Firewall**

This guide presents **two concrete approaches** to implement a DNS server with firewall and filtering:

1. **PowerDNS Recursor + dnsdist** (open‑source, battle‑tested)
2. **Custom Go‑based DNS Server** (built from scratch, lightweight)

Each section includes:

* **Conceptual overview**
* **Key features covered**
* **Step‑by‑step setup** with code/config snippets
* **Latest Git repository link**

## 1. PowerDNS Recursor + dnsdist

Leverages the **PowerDNS Recursor** for recursive resolution and **dnsdist** as a programmable DNS firewall and load balancer.

### 1.1. Key Features

* **Firewall**: exact, regex, suffix blocking via Lua scripts in dnsdist
* **Filtering**: blocklists loaded from files or external feeds
* **DNSSEC**: validation in recursor
* **Encrypted DNS**: DoT/DoH via dnsdist
* **Metrics**: Prometheus endpoints, built‑in web UI

### 1.2. Git Repositories

* Recursor: <https://github.com/PowerDNS/pdns-recursor>
* dnsdist: <https://github.com/PowerDNS/dnsdist>

### 1.3. Architecture

Client → dnsdist (53) → [Lua firewall] → pdns\_recursor (127.0.0.1:5300) → Internet

* **dnsdist**: front‑end, applies firewall rules, supports DoT/DoH
* **pdns\_recursor**: back‑end resolver, DNSSEC, caching

### 1.4. Step‑by‑Step Setup

#### 1.4.1. Install Packages

sudo apt update  
sudo apt install -y pdns-recursor dnsdist

#### 1.4.2. Configure Recursor (/etc/powerdns/recursor.conf)

local-address=127.0.0.1  
local-port=5300  
allow-from=127.0.0.1/32  
dnssec=process  
prometheus-address=127.0.0.1:9171

sudo systemctl enable --now pdns-recursor

#### 1.4.3. Configure dnsdist (/etc/dnsdist/dnsdist.conf)

-- listen on 53  
setLocal("0.0.0.0:53")  
setACL({"0.0.0.0/0"})  
  
-- forward to recursor  
newServer({address="127.0.0.1:5300", name="recursor"})  
  
-- refuse action  
dropAction = DropAction  
  
-- exact blocks  
addAction("malicious.com", dropAction)  
addAction("ads.example.net", dropAction)  
  
-- regex blocks  
addAction(RE2(".\*tracker.\*"), dropAction)  
  
-- suffix blocklist  
local f = io.open("/etc/dnsdist/blocklist.txt","r")  
if f then for line in f:lines() do local d=line:match('%S+'); if d then local n=newSuffixMatchNode(); n:add(d); addAction(n, dropAction); end end f:close() end  
  
-- optional web UI  
setWebserver("0.0.0.0:8083")  
setWebserverConfig({password="changeme"})

sudo mkdir -p /etc/dnsdist  
echo "phishing.com" | sudo tee /etc/dnsdist/blocklist.txt  
sudo dnsdist --check-config  
sudo systemctl enable --now dnsdist

#### 1.4.4. Testing

dig @<EC2-IP> google.com # allowed  
dig @<EC2-IP> malicious.com # blocked (no response)

## 2. Custom Go‑based DNS Server + Firewall

A lightweight DNS resolver built in Go using the miekg/dns library, with simple in‑process filtering.

### 2.1. Key Features

* **Minimal dependencies**: pure Go, single binary
* **Firewall**: inline exact/regex/suffix matching
* **Caching**: in‑memory map with TTL
* **Extensible**: add blocklist from file

### 2.2. Git Repository

* Boilerplate: <https://github.com/yourusername/go-dns-firewall> (template)

### 2.3. Code Overview

package main  
import (  
 "github.com/miekg/dns"  
 "regexp"; "strings"; "time"  
)  
var (  
 exactBlock = map[string]bool{"bad.com":true}  
 regexBlock = regexp.MustCompile(`.\*ads.\*`)  
 suffixBlock = []string{"tracker.net"}  
 cache = make(map[string]dns.RR)  
)  
  
func handle(w dns.ResponseWriter, r \*dns.Msg) {  
 q := r.Question[0].Name  
 name := strings.TrimSuffix(q, ".")  
 // firewall rules  
 if exactBlock[name] || regexBlock.MatchString(name) {  
 m:= new(dns.Msg); m.SetRcode(r, dns.RcodeRefused)  
 w.WriteMsg(m); return  
 }  
 for \_, s := range suffixBlock { if strings.HasSuffix(name, s) { m:=new(dns.Msg); m.SetRcode(r, dns.RcodeRefused); w.WriteMsg(m); return }}  
 // simple cache hit  
 if rr, ok := cache[name]; ok { m:=new(dns.Msg); m.SetReply(r); m.Answer= []dns.RR{rr}; w.WriteMsg(m); return }  
 // fallback to upstream  
 c:= new(dns.Client)  
 resp,\_:= c.Exchange(r, "8.8.8.8:53")  
 if len(resp.Answer)>0 { cache[name]=resp.Answer[0]; time.AfterFunc(5\*time.Minute, func(){ delete(cache,name) }) }  
 w.WriteMsg(resp)  
}  
func main(){  
 dns.HandleFunc(".", handle)  
 server := &dns.Server{Addr:":53", Net:"udp"}  
 server.ListenAndServe()  
}

### 2.4. Build & Run

git clone https://github.com/yourusername/go-dns-firewall.git  
cd go-dns-firewall  
go build -o go-dns-fw  
sudo ./go-dns-fw

### 2.5. Deployment

Run as a service on EC2, open port 53, and test as above.

*Document generated on 2025‑07‑11*