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

**Overview**  
This guide covers two approaches for implementing a DNS server with firewall capabilities:

1. **Using Open Source DNS Software** (leveraging battle‑tested projects)
2. **Building from Scratch** (a custom DNS engine with filtering logic)

At the end of each section, you’ll find the latest Git repository links for actively maintained DNS servers.

## 1. Using Open Source DNS Software

Leveraging existing DNS implementations saves time and benefits from community support, security patches, and extensible firewall features.

### 1.1. Unbound + RPZ Firewall

* **Description**: High‑performance recursive resolver with built‑in Response Policy Zones (RPZ).
* **Firewall**: Native RPZ support for domain blocking.
* **Key Features**: DNSSEC, DoT/DoH, qname‑minimisation.
* **GitHub**: <https://github.com/NLnetLabs/unbound>
* **Latest Release**: v1.19.3 (Jul 2025)

### 1.2. PowerDNS Recursor + dnsdist

**Description**: Modular recursive resolver (pdns\_recursor) paired with dnsdist, a high‑performance DNS load balancer and firewall written in C++.  
**Firewall**: dnsdist uses Lua scripts to apply exact, suffix, or regex‑based blocking rules and can load external blocklists.  
**Key Features**: High QPS handling, DNSSEC, DNS‑over‑HTTPS/TLS support, Lua scripting, real‑time metrics, Prometheus integration.  
**Repositories**:

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

#### 1.2.1. Architecture Overview

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

* **dnsdist** listens on UDP/TCP 53, applies filters, and forwards allowed queries to the recursor.
* **pdns\_recursor** performs iterative resolution, caching, DNSSEC validation, and returns answers back through dnsdist.

#### 1.2.2. Prerequisites

* Ubuntu 22.04 LTS (or Debian 11+) on EC2
* Administrative (sudo) privileges
* Security Group: open UDP/TCP port 53

#### 1.2.3. Installation

# Add official PowerDNS repository  
sudo apt update  
sudo apt install -y curl ca-certificates apt-transport-https gnupg software-properties-common  
sudo install -d /etc/apt/keyrings  
curl https://repo.powerdns.com/FD380FBB-pub.asc | sudo tee /etc/apt/keyrings/pdns.asc  
echo "deb [signed-by=/etc/apt/keyrings/pdns.asc] http://repo.powerdns.com/ubuntu jammy-rec-48 main" | sudo tee /etc/apt/sources.list.d/pdns-recursor.list  
echo "deb [signed-by=/etc/apt/keyrings/pdns.asc] http://repo.powerdns.com/ubuntu jammy-dnsdist-18 main" | sudo tee /etc/apt/sources.list.d/pdns-dnsdist.list  
sudo apt update  
sudo apt install -y pdns-recursor dnsdist

#### 1.2.4. Configure PowerDNS Recursor

Edit /etc/powerdns/recursor.conf:

# Bind only to localhost  
local-address=127.0.0.1  
local-port=5300  
allow-from=127.0.0.1/32  
# Enable DNSSEC validation  
dnssec=process  
# Optional: metrics on Prometheus port  
prometheus-address=127.0.0.1:9171

Restart:

sudo systemctl enable --now pdns-recursor

#### 1.2.5. Configure dnsdist Firewall

Create /etc/dnsdist/dnsdist.conf:

-- Listen for DNS queries  
setLocal("0.0.0.0:53")  
setACL({"0.0.0.0/0"})  
  
-- Forward to recursor  
newServer({address="127.0.0.1:5300", name="recursor"})  
  
-- Define block-action for REFUSED responses  
local refuse = DNSResponseAction({rcode = RCode.REFUSED})  
  
-- Exact domain blocks  
addAction("malicious.com", refuse)  
addAction("ads.example.net", refuse)  
  
-- Regex blocks  
addAction(RE2(".\*tracker.\*"), refuse)  
  
-- Suffix blocklist from file  
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 addAction(newSuffixMatchNode():add(d), refuse) end  
 end  
 f:close()  
end  
  
-- Web admin (optional)  
setWebserver("0.0.0.0:8083")  
setWebserverConfig({password = "changeme"})

Create a sample blocklist:

sudo mkdir -p /etc/dnsdist  
echo "phishing.com" | sudo tee /etc/dnsdist/blocklist.txt

Validate & start:

sudo dnsdist --check-config  
sudo systemctl enable --now dnsdist

#### 1.2.6. Testing

dig @<EC2-IP> google.com # allowed  
 dig @<EC2-IP> malicious.com # REFUSED  
 dig @<EC2-IP> ads.example.net # REFUSED

Monitor metrics: http://:8083 or Prometheus endpoint.

### 1.3. CoreDNS CoreDNS

* **Description**: Go‑based DNS server designed for cloud‑native environments.
* **Firewall**: Plugins such as rpz, filter, block allow domain filtering.
* **Key Features**: Kubernetes integration, plugin architecture.
* **GitHub**: <https://github.com/coredns/coredns>

### 1.4. BIND 9 + RPZ

* **Description**: The most widely deployed authoritative/recursive DNS server.
* **Firewall**: RPZ support via response-policy.
* **Key Features**: Views, zones, ACLs, DNSSEC.
* **Repo**: <https://gitlab.isc.org/isc-projects/bind9>

## 2. Building a DNS Server from Scratch

When you need maximum control, you can implement your own DNS protocol stack and filtering logic.

### 2.1. Components & Protocol

1. **DNS Message Parser/Serializer**
   * Parse UDP/TCP payloads according to RFC 1035.
   * Build query/response structures.
2. **Recursive Resolver Logic**
   * Root hints, iterative queries, caching layer.
3. **Firewall Module**
   * Domain blacklist/whitelist checks.
   * Regex and suffix matching.
4. **Networking Layer**
   * UDP socket listener on port 53.
   * Threading or event loop (e.g., libevent, asyncio).

### 2.2. Technology Choices

* **Language**: Go, Rust, or C++ for performance; Python for ease of prototyping.
* **Caching**: In‑memory LRU cache (e.g., hashicorp/golang-lru).
* **Filtering**: Trie or suffix tree for fast domain lookups; regex engine for patterns.

### 2.3. Sample Projects & Code Samples

* **Python DNS Server Skeleton**: <https://github.com/irungentoo/pydns>
* **Go DNS Library**: <https://github.com/miekg/dns>
* **Rust DNS Prototype**: <https://github.com/sile/dns-rust>

## 3. Deployment Considerations

* **High Availability**: Anycast IPs, multiple instances behind load balancers.
* **Logging & Monitoring**: Export metrics via Prometheus, logs to ELK.
* **Security**: DNSSEC validation, rate limiting, IP whitelists.

## 4. Additional Resources

* **DNS RFCs**: <https://tools.ietf.org/html/rfc1035>
* **DNS Security**: <https://tools.ietf.org/html/rfc7858> (DoT), RFC8484 (DoH)

*Document generated on 2025‑07‑11*