User Guide

LLM-Optimized Reference

OPNsense 24.x
February 2026

Table of Contents

1 Introduction to OPNsense

> ★ THIS GUIDE IS OPTIMIZED FOR LLM/AI AGENTS. It provides structured, concise reference material for querying OPNsense configuration via MCP tools or API. Human-readable but

OPNsense is an open-source firewall and routing platform based on FreeBSD with HardenedBSD security enhancements. Core capabilities: stateful packet filtering (pf), NAT, VPN (IPsec/OpenVPN/WireGuard), IDS/IPS (Suricata), traffic shaping, and high availability (CARP).

Quick Reference: OPNsense Fundamentals

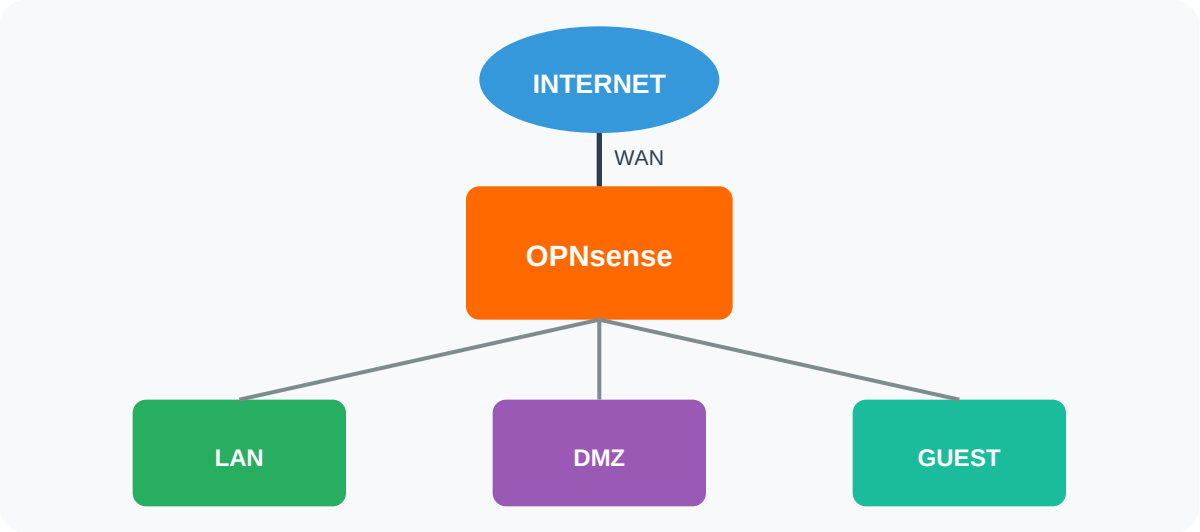
Concept	Key Point
Firewall Engine	pf (packet filter) from OpenBSD - rules in /tmp/rules.debug
Rule Direction	Rules apply where traffic ENTERS the firewall (inbound)
Rule Order	First match wins (with 'quick' flag, default enabled)
State Table	Connections tracked automatically, return traffic auto-allowed
NAT Processing	NAT rules processed BEFORE firewall rules
Default Policy	Implicit deny - traffic blocked unless explicitly permitted
Config File	XML at /conf/config.xml - never edit directly, use API/GUI
API	REST API on port 443, key+secret auth, /api/ endpoints

Key Features

Feature	Description	Category
Stateful Firewall	Full packet inspection with state tracking	Security

Multi-WAN	Load balancing and failover support	Networking
VPN Support	IPsec, OpenVPN, WireGuard	Connectivity
Traffic Shaping	QoS with pipes and queues	Performance
Intrusion Detection	Suricata IDS/IPS integration	Security
High Availability	CARP with configuration sync	Reliability
Web Proxy	Squid with SSL inspection	Services
Captive Portal	Guest network authentication	Access Control

Network Architecture Overview



OPNsense sits between your internal networks and the internet, providing security, routing, and network services. It can manage multiple network segments (VLANs) and apply different security policies to each zone.

2 Initial Setup & Installation

System Requirements

Component	Minimum	Recommended
CPU	1 GHz dual-core	Multi-core 64-bit (AES-NI)
RAM	2 GB	8 GB or more
Storage	8 GB SSD	120 GB SSD
Network	2 NICs	Intel NICs recommended

✓ AES-NI support is highly recommended for VPN performance. Check CPU compatibility before deployment.

Installation Steps

- 1. Download the latest OPNsense ISO from opnsense.org
- 2. Create bootable USB using Rufus, Etcher, or dd command
- 3. Boot from USB and select 'Install (UFS)' for most setups
- 4. Configure network interfaces during installation
- 5. Set root password and complete installation
- 6. Access web interface at https://192.168.1.1 (default)
- 7. Complete initial setup wizard

Post-Installation Checklist

- ■ Update to latest version (System → Firmware → Status)
- ■ Configure WAN and LAN interfaces
- ■ Set hostname and domain
- ■ Configure DNS servers
- ■ Enable SSH access (if needed)
- ■ Set timezone and NTP servers
- ■ Create admin user (avoid using root)

- Configure backup schedule

3 Dashboard & Navigation

The OPNsense web interface provides a comprehensive dashboard and intuitive navigation system. The main menu is organized into logical categories for easy access to all features.

Main Menu Structure

Menu	Primary Functions
Lobby	Dashboard, widgets, password management
System	Settings, firmware, users, HA, certificates
Interfaces	Network interface configuration, VLANs
Firewall	Rules, NAT, aliases, traffic shaper
VPN	IPsec, OpenVPN, WireGuard
Services	DHCP, DNS, proxy, IDS/IPS
Reporting	Traffic graphs, logs, NetFlow
Diagnostics	System tools, packet capture, ping

Dashboard Widgets

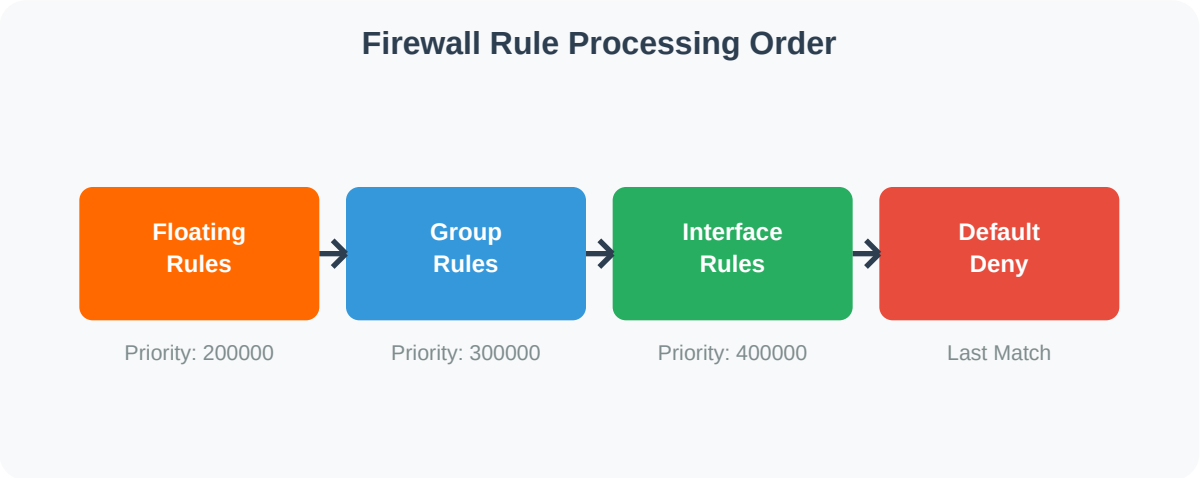
The dashboard is customizable with drag-and-drop widgets. Common widgets include:

- **System Information**: CPU, memory, uptime, version
- **Interfaces**: Interface status and traffic
- **Gateways**: WAN gateway status and latency
- **Traffic Graphs**: Real-time bandwidth visualization
- **Services**: Status of enabled services
- **Firewall Logs**: Recent blocked/passed traffic

4 Firewall Configuration

OPNsense contains a stateful packet filter based on pf (packet filter) from OpenBSD. The firewall inspects packets, maintains connection states, and applies rules to permit or deny traffic. Understanding rule processing, state management, and interface binding is essential for effective security.

Rule Processing Order



Rules are evaluated in a specific order based on their category and position. Understanding this order is critical for troubleshooting:

Priority	Rule Type	Description
1	Floating (quick)	Processed first, can match any interface
2	Group Rules	Applied to interface groups (e.g., all LANs)
3	Interface Rules	Per-interface rules, most common type
4	Default Deny	Implicit block if no rule matches

> ! By default, all rules have 'quick' enabled, meaning the first matching rule wins. Without 'quick', the last matching rule would apply (rarely desired).

Rule Actions Explained

Action	Network Response	Best Use Case
Pass	Traffic allowed	Explicitly permit desired traffic
Block	Silently dropped	WAN/untrusted - no response to attackers
Reject	TCP RST or ICMP unreachable	LAN/internal - faster client feedback

Block vs Reject: Use Block on WAN interfaces to avoid revealing firewall presence. Use Reject on internal interfaces so clients receive immediate feedback rather than waiting for connection timeout.

Stateful Packet Inspection

OPNsense maintains a state table tracking all active connections. This provides:

- **Performance**: Return traffic matched by state, not rule evaluation
- **Security**: TCP sequence number validation prevents injection
- **Simplicity**: Only need rules for initiating direction
- **Tracking**: Connection limits, timeouts, and diagnostics

State Table Options

Option	Effect	Use Case
Keep State	Normal stateful (default)	Most traffic
Sloppy State	Less strict validation	Asymmetric routing
Synproxy State	Proxy TCP handshake	Public servers (DDoS protection)
No State	Stateless rule	High-volume UDP, broadcasts

The state table size defaults to 1,000,000 entries. Monitor with 'pfctl -si' and increase under Firewall > Settings > Advanced if needed for high-traffic environments.

Aliases

Aliases are named groups of hosts, networks, ports, or URLs that simplify rule management. When an alias is updated, all rules using it automatically reflect changes.

Type	Content	Example Values
Hosts	IPs or FQDNs	10.0.0.5, server.example.com
Networks	CIDR subnets	192.168.1.0/24, 10.0.0.0/8
Ports	Port numbers/ranges	80, 443, 8080-8090
URL Table (IPs)	Remote IP list URL	Spamhaus DROP, Firehol
URL Table (Ports)	Remote port list	Dynamic port lists
GeoIP	Country codes	US, CN, RU (MaxMind DB)
BGP ASN	AS numbers	AS15169 (Google)
Dynamic IPv6 Host	Interface IPv6	Track WAN IPv6 address
MAC Address	Hardware addresses	00:11:22:33:44:55
External (advanced)	pfTables via script	Custom integrations

URL Table Aliases

URL Tables fetch IP lists from remote sources, enabling integration with threat intelligence feeds. Configure update frequency under the alias settings.

- **Spamhaus DROP**: https://www.spamhaus.org/drop/drop.txt
- **Firehol Level1**: https://iplists.firehol.org/files/firehol_level1.netset
- **Emerging Threats**: https://rules.emergingthreats.net/fwrules/emerging-Block-IPs.txt

Nest aliases within other aliases for complex rule sets. For example, create 'INTERNAL_NETS' containing 'LAN_NET', 'DMZ_NET', and 'GUEST_NET' aliases.

Rule Configuration Fields

Each firewall rule contains multiple fields controlling match criteria and behavior:

Field	Options	Notes
Action	Pass / Block / Reject	What to do with matched traffic
Interface	WAN, LAN, VLANs, Groups	Where rule applies
Direction	In / Out	'In' = entering interface (most common)
TCP/IP Version	IPv4, IPv6, IPv4+IPv6	Address family
Protocol	TCP, UDP, ICMP, Any, etc.	Layer 4 protocol
Source	Any, IP, Alias, Network	Traffic origin
Destination	Any, IP, Alias, Network	Traffic target
Destination Port	Any, Port, Alias, Range	Service port(s)
Log	Enabled/Disabled	Write to firewall log
Category	User-defined label	Organize rules visually

Advanced Rule Options

- **Source Port**: Match originating port (rarely needed)
- **Gateway**: Policy-based routing to specific gateway
- **Schedule**: Time-based rule activation
- **State Type**: Keep/Sloppy/Synproxy/None
- **State Timeout**: Override default connection timeout
- **Max States**: Limit connections from this rule
- **Max Src Nodes**: Limit unique source IPs
- **Max Connections / Src**: Per-source connection limit
- **Tag / Tagged**: Mark packets for later matching
- **Allow Options**: Permit IP options (rarely needed)
- **TCP Flags**: Match specific TCP flags

Interface Rules Best Practices

Rules are applied on the interface where traffic **enters** the firewall. This is the most common source of confusion for new users.

> ! Traffic FROM LAN TO WAN is filtered by LAN interface rules (inbound to firewall). Traffic FROM WAN TO LAN is filtered by WAN interface rules.

LAN Rules Example

A typical LAN ruleset allowing internet access while blocking inter-VLAN traffic:

#	Action	Source	Destination	Port	Description
1	Pass	LAN net	LAN address	443,22	Access firewall GUI/SSH
2	Block	LAN net	RFC1918	Any	Block other internal nets
3	Pass	LAN net	Any	Any	Allow internet access

WAN Rules Example

WAN interface typically has minimal rules - most traffic blocked by default. Only add rules for explicitly exposed services:

#	Action	Source	Destination	Port	Description
1	Pass	Any	WAN address	443	HTTPS to web server
2	Pass	Any	WAN address	51820/UDP	WireGuard VPN
-	Block	Any	Any	Any	(implicit default deny)

Floating Rules

Floating rules apply across multiple interfaces and are processed before interface rules. Use them for policies that span the entire firewall:

- Block known-bad IPs on ALL interfaces
- Apply QoS/traffic shaping globally
- Enforce policy across interface groups

- Quick deny rules that must match first
- Outbound direction filtering (rare)

> **i** Set 'Quick' on floating rules to stop processing immediately on match. Without 'Quick', floating rules mark the packet but processing continues to interface rules.

Viewing the Active Ruleset

The actual pf ruleset loaded in memory may differ from the GUI representation. Always verify with CLI tools when troubleshooting:

- `pfctl -sr` - Show all loaded rules
- `pfctl -ss` - Show state table (active connections)
- `pfctl -si` - Show filter statistics and counters
- `pfctl -vvsr` - Verbose rules with hit counters
- `cat /tmp/rules.debug` - View rules file before loading

5 Network Address Translation

NAT allows internal networks with private IPs to communicate with external networks. OPNsense supports multiple NAT types for different scenarios.

NAT Types

Type	Direction	Common Use
Port Forward	Inbound	Expose internal services
Outbound NAT	Outbound	Internet access for LAN
One-to-One	Bidirectional	Static IP mapping
NPTv6	IPv6	IPv6 prefix translation

Port Forwarding

Port forwarding (Destination NAT) redirects incoming traffic to internal hosts. Essential fields include:

- **Interface**: Usually WAN
- **Protocol**: TCP, UDP, or both
- **Destination Port**: External port to listen on
- **Redirect Target IP**: Internal host IP
- **Redirect Target Port**: Internal port

NAT rules are processed BEFORE firewall rules. A port forward with 'Pass' association will bypass other firewall rules for that traffic.

Outbound NAT Modes

Mode	Description

Automatic	Default, auto-generates rules for all interfaces
Hybrid	Auto rules plus custom manual rules
Manual	Full manual control, no auto rules
Disabled	No outbound NAT (transparent bridge)

Practical NAT Examples

Common NAT configurations for real-world scenarios. Each example includes the complete configuration steps.

Example 1: Web Server Port Forward (HTTP/HTTPS)

Expose an internal web server (192.168.1.100) to the internet on standard ports.

Setting	HTTP Rule	HTTPS Rule
Interface	WAN	WAN
Protocol	TCP	TCP
Destination	WAN address	WAN address
Destination Port	80	443
Redirect Target IP	192.168.1.100	192.168.1.100
Redirect Target Port	80	443
NAT Reflection	Enable (for hairpin)	Enable (for hairpin)
Filter Rule Assoc.	Pass	Pass

✗ SECURITY: Never expose management interfaces (SSH, WebUI) directly. Use VPN for remote administration. Consider adding source IP restrictions if public access is required.

Example 2: Game Server Port Forward

Common game server ports. Host is 192.168.1.50 on LAN.

Game/Service	Protocol	Port(s)	Notes
Minecraft Java	TCP	25565	Default server port
Minecraft Bedrock	UDP	19132	Bedrock edition
Valheim	UDP	2456-2458	Range required
Terraria	TCP	7777	Default server port
Plex Media Server	TCP	32400	Remote streaming
ARK: Survival	UDP	7777, 27015	Game + query ports
Counter-Strike 2	TCP/UDP	27015-27020	Server + RCON

- 1. Firewall → NAT → Port Forward → Add
- 2. Interface: WAN, Protocol: as shown above
- 3. Destination: WAN address, Port: game port(s)
- 4. Redirect target: Game server LAN IP
- 5. Filter rule association: Pass (creates firewall rule)
- 6. Save and Apply Changes

Example 3: Hairpin NAT (NAT Reflection)

Allows LAN clients to access port-forwarded services using the public IP/domain instead of the internal IP. Required when internal DNS returns public IP.

Setting	Location	Value
NAT Reflection mode	Firewall → Settings → Advanced	Pure NAT (recommended)
Reflection for port forwards	Firewall → Settings → Advanced	Enabled
Automatic outbound NAT for Reflection	Firewall → Settings → Advanced	Enabled
Per-rule reflection	Each port forward rule → NAT reflection	Enable

SPLIT DNS ALTERNATIVE: Instead of hairpin NAT, configure internal DNS (Unbound) to resolve your domain to the internal IP. This is more efficient and avoids NAT complexity. Services →

Example 4: Policy-Based Outbound NAT

Route specific traffic through different WAN IPs or interfaces. Useful for multi-WAN setups or VPN policy routing.

Scenario	Source	Translation Address
VPN clients use VPN gateway	10.0.8.0/24 (VPN subnet)	VPN interface address
Servers use static IP	192.168.1.100/32	WAN static IP alias
IoT uses secondary WAN	192.168.40.0/24 (IoT VLAN)	WAN2 interface address
No NAT for site-to-site VPN	192.168.1.0/24 to 10.0.0.0/24	None (NO NAT rule)

- 1. Set NAT mode to Hybrid or Manual (Firewall → NAT → Outbound)
- 2. Add manual rule ABOVE automatic rules
- 3. Interface: Outbound interface (WAN, WAN2, VPN)
- 4. Source: Network/host to match
- 5. Translation: Interface address or specific IP
- 6. For NO NAT: Check 'Do not NAT' or set Translation to 'None'

Example 5: One-to-One (1:1) NAT

Maps an entire public IP to an internal host. All ports forwarded bidirectionally. Useful when ISP provides multiple static IPs.

Setting	Value	Notes
Interface	WAN	Interface with public IP
External IP	203.0.113.10	Additional public IP from ISP
Internal IP	192.168.1.100	Server to expose
Destination	Any or specific subnet	Usually 'any'

NAT Reflection	Enable	If internal access needed

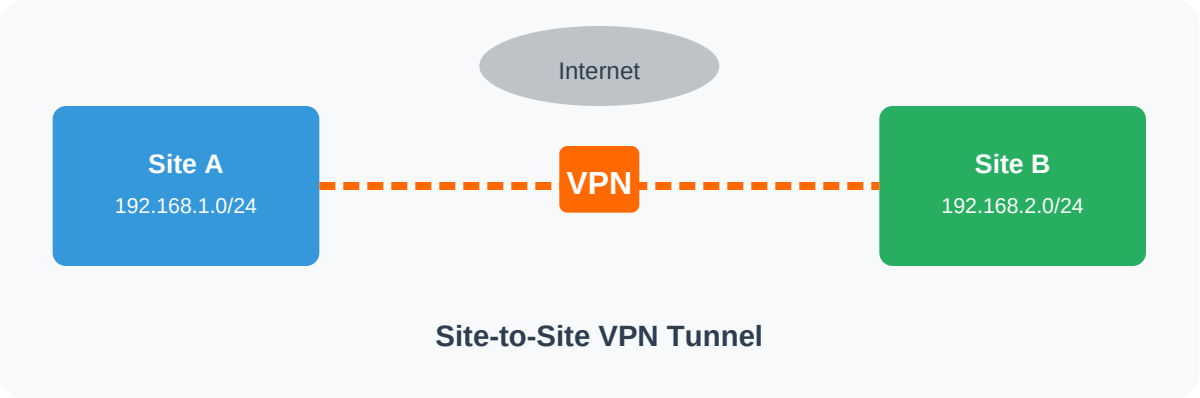
! 1:1 NAT FIREWALL RULES: Unlike port forwards, 1:1 NAT does NOT automatically create firewall rules. You must manually create WAN rules to allow desired traffic to the external IP.

MCP NAT Tools Quick Reference

Task	MCP Tool
List port forwards	opnsense_nat_list_port_forwards
Create port forward	opnsense_nat_create_port_forward
Delete port forward	opnsense_nat_delete_port_forward
List outbound NAT	opnsense_nat_list_outbound
Create outbound rule	opnsense_nat_create_outbound_rule
Get/Set NAT mode	opnsense_nat_get_mode / opnsense_nat_set_mode
Analyze NAT config	opnsense_nat_analyze_config
Apply NAT changes	opnsense_nat_apply_changes

6 Virtual Private Networks

OPNsense supports multiple VPN technologies for secure remote access and site-to-site connectivity. Each technology has different strengths for specific deployment scenarios. This chapter covers IPsec, OpenVPN, and WireGuard in detail.



Site-to-Site VPN Tunnel

VPN Technology Comparison

Feature	IPsec	OpenVPN	WireGuard
Best Use Case	Site-to-Site	Road Warriors	Modern/Mobile
Performance	Excellent	Good	Excellent
Configuration	Complex	Moderate	Simple
Protocol	UDP 500/4500	UDP 1194 or TCP	UDP 51820
NAT Traversal	NAT-T (UDP 4500)	Native	Native
Mobile Support	Limited	Good	Excellent
Interoperability	Universal	OpenVPN only	WireGuard only

IPsec VPN

IPsec is the industry standard for site-to-site VPNs, supported by virtually all enterprise firewalls. OPNsense 23.1+ uses the modern swanctl/Connections interface based on strongSwan's vici protocol.

IPsec Phases Explained

Phase	Purpose	Key Settings
Phase 1 (IKE)	Authenticate peers, establish secure channel	Encryption, Hash, DH Group, Lifetime
Phase 2 (ESP)	Negotiate data encryption parameters	Encryption, Hash, PFS Group, Lifetime

Recommended Phase 1 Settings

Setting	Recommended Value	Notes
Key Exchange	IKEv2	More secure, faster than IKEv1
Encryption	AES-256-GCM	Authenticated encryption
Hash	(implicit in GCM)	GCM includes authentication
DH Group	ECP384 or Curve25519	Modern elliptic curve
Lifetime	28800 seconds (8 hours)	Balance security/overhead
DPD	Enabled, 10s interval	Detect dead peers

Recommended Phase 2 Settings

Setting	Recommended Value	Notes
Encryption	AES-256-GCM	Match Phase 1
PFS Group	ECP384 or Curve25519	Perfect Forward Secrecy
Lifetime	3600 seconds (1 hour)	Shorter than Phase 1
Local Network	LAN subnet	What to encrypt
Remote Network	Remote LAN subnet	Peer's protected network

> **!** Both sides must have matching Phase 1 and Phase 2 settings. Mismatched encryption, hash, or DH groups are the most common cause of tunnel failures.

IPsec Site-to-Site Setup Checklist

- 1. VPN → IPsec → Connections → Add new connection
- 2. Configure local and remote endpoints (IPs or FQDNs)
- 3. Set authentication (PSK or certificates)
- 4. Define local and remote networks (Children/Phase 2)
- 5. Create firewall rule: WAN, UDP 500 and 4500, to WAN address
- 6. Create firewall rule: IPsec interface, permit traffic to/from remote nets
- 7. Enable connection and check Status → IPsec → Overview

OpenVPN

OpenVPN is an SSL/TLS-based VPN ideal for remote access (road warrior) scenarios. It works well through restrictive firewalls and NAT, making it reliable for mobile users. OPNsense uses the Instances model for OpenVPN configuration.

OpenVPN Modes

Mode	Authentication	Use Case
SSL/TLS	Certificates + optional user auth	Standard, most secure
SSL/TLS + User Auth	Cert + username/password	Road warriors with MFA
Shared Key	Static pre-shared key	Simple site-to-site (not recommended)

Topology Options

Topology	IP Assignment	Best For
subnet	Unique IP per client from pool	Most common, recommended
net30	/30 subnet per client	Legacy compatibility
p2p	Point-to-point (no pool)	Site-to-site only

OpenVPN Server Setup

- 1. System → Trust → Authorities: Create or import CA certificate
- 2. System → Trust → Certificates: Create server certificate signed by CA
- 3. VPN → OpenVPN → Instances: Add new server instance
- 4. Configure: Role=Server, Protocol=UDP, Port=1194, Topology=subnet
- 5. Set tunnel network (e.g., 10.10.0.0/24) and local networks to push
- 6. Select server certificate and CA
- 7. Firewall: Allow UDP 1194 on WAN
- 8. Firewall: Allow traffic on OpenVPN interface
- 9. VPN → OpenVPN → Client Export: Generate client configs

Use the Client Export plugin (os-openvpn-client-export) to generate ready-to-use configuration files and installers for Windows, macOS, Linux, iOS, and Android.

WireGuard VPN

WireGuard is a modern VPN protocol designed for simplicity and performance. It uses state-of-the-art cryptography (Curve25519, ChaCha20, Poly1305) and has a minimal attack surface with ~4,000 lines of code vs 100,000+ for OpenVPN/IPsec.

WireGuard Concepts

Term	Description
Instance	Local WireGuard interface with keypair and listen port
Peer	Remote endpoint with public key and allowed IPs
Allowed IPs	What traffic to send through this peer (routing)
Endpoint	Public IP:port of peer (optional for server)
Keepalive	Periodic packets to maintain NAT mappings

WireGuard Server Setup

- 1. VPN → WireGuard → Instances: Create new instance
- 2. Generate keypair (or paste existing public/private keys)
- 3. Set listen port (default 51820) and tunnel address (e.g., 10.10.10.1/24)
- 4. Optionally add DNS servers to push to clients
- 5. Save and note the public key for peer configuration
- 6. VPN → WireGuard → Peers: Add each client as a peer
- 7. Enter client's public key and Allowed IPs (client's tunnel IP/32)
- 8. Firewall → Rules → WAN: Allow UDP 51820 to WAN address
- 9. Firewall → Rules → WireGuard: Allow traffic from tunnel network
- 10. Interfaces → Assignments: Optionally assign WG interface for advanced rules

> **i** WireGuard is 'silent' - it won't respond to unauthenticated packets. If a peer can't connect, it appears as if nothing is listening. This is a security feature.

VPN Firewall Rules

All VPN types require firewall rules to permit traffic. Remember:

- **WAN rules**: Allow VPN protocol inbound (UDP 500/4500, 1194, 51820)
- **VPN interface rules**: Control what VPN clients can access
- **IPsec uses its own tab**: Firewall → Rules → IPsec (not an interface)
- **NAT bypass**: VPN traffic typically bypasses outbound NAT

7 DNS Services (Unbound)

Unbound is OPNsense's default DNS resolver, providing recursive DNS resolution with DNSSEC validation, caching, and advanced features like DNS-over-TLS (DoT). It replaces the older Dnsmasq and can function as both a resolver and forwarder.

DNS Resolution Modes

Mode	Behavior	Privacy	Speed
Recursive	Query root servers directly	Best (no third party)	Slower initial
Forwarding	Forward to upstream DNS	Depends on upstream	Faster
DoT Forwarding	Encrypted forwarding	Good (encrypted)	Moderate

> **i** Recursive mode queries authoritative servers directly, eliminating reliance on third-party DNS providers. Enable DNSSEC to validate responses cryptographically.

General Settings

Setting	Recommended	Notes
Enable	Checked	Activate Unbound service
Listen Port	53	Standard DNS port
Network Interfaces	LAN, VLANs	Where to accept queries
DNSSEC	Enabled	Validate signed responses
DNS64	Disabled	Only for IPv6-only networks
Register DHCP Leases	Enabled	Resolve local hostnames

Register Static Mappings	Enabled	Include DHCP static entries
Local Zone Type	Transparent	Allow upstream for unknown local

DNS-over-TLS (DoT) Configuration

DoT encrypts DNS queries to upstream servers, preventing eavesdropping and manipulation. Configure under Services → Unbound DNS → DNS over TLS.

Provider	Server	Port	Hostname Verification
Cloudflare	1.1.1.1	853	cloudflare-dns.com
Google	8.8.8.8	853	dns.google
Quad9	9.9.9.9	853	dns.quad9.net
Mullvad	194.242.2.2	853	dns.mullvad.net

✓ Enable 'Use System Nameservers' under Query Forwarding and add DoT servers above. The hostname verification ensures you're connecting to the legitimate server.

DNS Blocklists

Unbound can block domains using DNSBL (DNS Blocklist) feature, providing network-wide ad blocking and malware protection without client-side software.

- **Ads/Tracking**: Block advertising and analytics domains
- **Malware**: Block known malicious domains
- **Adult Content**: Family-safe filtering
- **Custom**: Your own domain blacklist

Popular Blocklist Sources

- **Steven Black Unified**: Ads, malware, fakenews, gambling, social
- **OISD**: Optimized comprehensive blocklist
- **AdGuard DNS Filter**: Curated ad blocking
- **Energized Protection**: Multiple protection levels

Host and Domain Overrides

Override DNS responses for specific hosts or entire domains. Useful for:

- Split-horizon DNS (internal vs external resolution)
- Redirect internal services to local IPs
- Block specific domains by pointing to 0.0.0.0
- Override public DNS for local resources

8 Intrusion Detection (Suricata)

Suricata is a high-performance Network IDS/IPS (Intrusion Detection/Prevention System) integrated into OPNsense. It inspects traffic for malicious patterns, exploits, and policy violations using signature-based detection.

IDS vs IPS Mode

Mode	Action on Match	Impact	Use Case
IDS	Alert only	None	Monitoring, tuning rules
IPS (Inline)	Block traffic	May block legitimate	Active protection

! Start with IDS mode to tune rules and eliminate false positives before enabling IPS blocking. Aggressive IPS rules can break legitimate applications.

Ruleset Sources

Ruleset	Type	Focus
ET Open	Free	Emerging threats, malware, exploits
ET Pro	Commercial	Enhanced ET with more coverage
Snort Community	Free (registration)	Traditional IDS rules
Snort Subscriber	Commercial	Latest Snort rules
Abuse.ch	Free	SSL certs, malware URLs, botnets
OPNsense App Detection	Free	Application identification

Configuration Steps

- 1. Services → Intrusion Detection → Administration
- 2. Enable IDS, select interface(s) to monitor
- 3. Set Pattern Matcher (Hyperscan if available, otherwise AC)
- 4. Download tab: Enable desired rulesets
- 5. Click 'Download & Update Rules'
- 6. Rules tab: Review and enable rule categories
- 7. Policy tab: Create policies for different interfaces
- 8. User Defined tab: Add custom rules if needed
- 9. Start with IDS mode, monitor alerts
- 10. After tuning, optionally enable IPS mode

Rule Categories

- **emerging-malware**: Known malware signatures
- **emerging-exploit**: Exploit attempts
- **emerging-scan**: Port scans and recon
- **emerging-dos**: Denial of service patterns
- **emerging-web_server**: Web application attacks
- **emerging-policy**: Policy violations (P2P, etc.)

Tuning and False Positives

False positives are inevitable. Use these strategies to tune your deployment:

- Disable overly broad rules generating excessive alerts
- Use 'Disable SID' to suppress specific signatures
- Create suppression rules for known-good traffic
- Whitelist internal scanners and security tools
- Review alerts regularly and adjust policies

> **i** Suricata is CPU-intensive. On lower-powered hardware, enable only essential rule categories or consider monitoring only WAN interface.

9 Traffic Shaping & QoS

Traffic shaping (QoS - Quality of Service) manages bandwidth allocation and prioritizes critical traffic. OPNsense provides two shaping systems: the legacy ALTQ scheduler and the modern pipes/queues system based on dummynet.

Key Concepts

Component	Function	Location
Pipes	Define maximum bandwidth	Firewall → Shaper → Pipes
Queues	Traffic classes within pipes	Firewall → Shaper → Queues
Rules	Match traffic to queues	Firewall → Shaper → Rules

Shaping Workflow

- 1. Create Pipes: Define bandwidth limits (download/upload)
- 2. Create Queues: Assign queues to pipes with weights/priorities
- 3. Create Rules: Match traffic to queues (by IP, port, DSCP, etc.)
- 4. Test: Verify traffic is being shaped correctly

Common QoS Scenarios

- **VoIP Priority**: Low latency queue for SIP/RTP traffic
- **Bandwidth Caps**: Per-user or per-network limits
- **Gaming**: Prioritize game traffic over bulk downloads
- **Work Applications**: Prioritize VPN and business apps
- **Guest Network**: Limit guest VLAN bandwidth

✓ Shape to 95% of actual bandwidth. This keeps queuing under YOUR control rather than letting your ISP's equipment decide what gets dropped.

FQ-CoDel for Bufferbloat

FQ-CoDel (Fair Queuing with Controlled Delay) reduces bufferbloat - excessive latency caused by large buffers. Enable CoDel on queues for responsive connections even under heavy load.

10 High Availability & CARP

OPNsense supports high availability through CARP (Common Address Redundancy Protocol), enabling automatic failover between two firewalls. Combined with pfsync for state synchronization and XMLRPC for configuration sync, this provides seamless failover.

HA Components

Component	Purpose	Protocol
CARP	Virtual IP failover	IP Protocol 112
pfsync	State table sync	IP Protocol 240
XMLRPC	Config synchronization	TCP 443 (HTTPS)

Prerequisites

- Two identical OPNsense installations (same version)
- Dedicated sync interface directly connected (crossover or switch)
- Same interface assignments on both nodes
- Unique real IPs + shared CARP VIPs on each interface
- Licenses (if using commercial plugins) on both nodes

Setup Procedure

- 1. Configure dedicated SYNC interface on both nodes
- 2. Assign static IPs to SYNC interface (e.g., 172.16.0.1 and 172.16.0.2)
- 3. On PRIMARY: System → High Availability → Settings
- 4. Enable state synchronization, set SYNC interface
- 5. Add secondary node IP as synchronization peer
- 6. Configure XMLRPC sync settings (admin credentials for secondary)
- 7. Select items to synchronize (rules, aliases, NAT, etc.)
- 8. Create CARP VIPs on WAN and LAN interfaces
- 9. Update DHCP/services to use CARP VIPs, not real IPs
- 10. Test failover by stopping CARP on primary

CARP VIP Configuration

Field	Description
Mode	CARP
Interface	WAN, LAN, or VLAN
Address	Shared virtual IP
VHID	Virtual Host ID (unique per VIP per broadcast domain)
Password	Shared secret (same on both nodes)
Advertising Frequency	Base=1, Skew=0 for primary, Skew=100 for secondary

The node with lower skew value becomes MASTER. Set primary to skew 0 and secondary to skew 100. During maintenance, increase primary's skew to force failover.

Troubleshooting HA

- **Both nodes MASTER**: VHID conflict or sync interface down
- **State not syncing**: Check pfsync interface, firewall rules
- **Config not syncing**: Verify XMLRPC credentials and connectivity
- **VPN failover slow**: Enable DPD on VPN connections

11 Additional Services

Beyond the core firewall and VPN functionality, OPNsense includes many built-in services and supports extensive plugins for additional functionality.

Core Services Overview

Service	Purpose	Location
DHCPv4/v6	IP address assignment	Services → DHCPv4/v6
Router Adv.	IPv6 SLAAC	Services → Router Advertisements
NTP	Time synchronization	Services → Network Time
Syslog	Remote logging	Services → Syslog
SNMP	Monitoring protocol	Services → SNMP
Dynamic DNS	Update DNS records	Services → Dynamic DNS
Wake on LAN	Remote power-on	Services → Wake on LAN

DHCP Configuration

DHCP Server provides automatic IP configuration to clients. Key settings:

- **Range**: Pool of addresses to assign
- **Gateway**: Default route (usually OPNsense LAN IP)
- **DNS Servers**: Use OPNsense IP for local resolution
- **Lease Time**: How long assignments are valid
- **Static Mappings**: Fixed IPs for specific MAC addresses

Popular Plugins

Plugin	Function

os-acme-client	Let's Encrypt certificate automation
os-haproxy	Load balancer and reverse proxy
os-nginx	Web server/reverse proxy
os-crowdsec	Collaborative threat intelligence
os-wireguard	WireGuard VPN
os-zerotier	ZeroTier SD-WAN
os-tailscale	Tailscale mesh VPN
os-zabbix-agent	Zabbix monitoring
os-telegraf	Metrics collection (InfluxDB)
os-theme-*	UI themes

Install plugins via System → Firmware → Plugins. Search for 'os-' prefix.

12 Troubleshooting & Diagnostics

Effective troubleshooting requires familiarity with OPNsense's diagnostic tools, log files, and command-line utilities. This chapter covers essential techniques for diagnosing network and firewall issues.

GUI Diagnostic Tools

Tool	Location	Use Case
Live Log	Firewall → Log Files → Live View	Real-time rule matching
Packet Capture	Interfaces → Diagnostics → Packet Capture	Deep packet analysis
States	Firewall → Diagnostics → States	View active connections
pfTop	Diagnostics → pfTop	Real-time state viewer
Activity	Diagnostics → Activity	CPU/process monitoring
DNS Lookup	Interfaces → Diagnostics → DNS Lookup	Test DNS resolution
Ping	Interfaces → Diagnostics → Ping	ICMP connectivity test
Traceroute	Interfaces → Diagnostics → Traceroute	Path analysis
Port Probe	Interfaces → Diagnostics → Port Probe	TCP port test

Essential CLI Commands

Command	Purpose
pfctl -sr	Show loaded firewall rules
pfctl -ss	Show state table (active connections)
pfctl -si	Show filter statistics

pfctl -vvsr	Verbose rules with counters
pfctl -k host	Kill states for specific host
netstat -rn	Show routing table
netstat -an	Show all connections
sockstat -l	Show listening ports
tcpdump -i em0	Capture traffic on interface
configctl filter reload	Reload firewall rules
configctl interface reconfigure	Reconfigure interfaces
tail -f /var/log/filter.log	Live firewall log

Common Issues & Solutions

Traffic blocked

→ Enable logging on rules, check Live Log, verify rule order

Cannot reach internet

→ Check WAN status, gateway, outbound NAT

DNS not resolving

→ Verify Unbound running, check forwarding settings

VPN won't connect

→ Verify ports open, matching phase settings, check logs

Slow performance

→ Check interface errors, CPU usage, disable SYN proxy

CARP not failing over

→ Check VHID uniqueness, sync interface, skew values

NAT not working

→ Verify mode, check associated filter rule

Can't access GUI

→ SSH in, check webconfigurator service, anti-lockout rule

Important Log Files

`/var/log/filter.log` - Firewall blocks/passes
`/var/log/system.log` - General system messages
`/var/log/latest.log` - Most recent log entries
`/var/log/dhcpd.log` - DHCP server activity
`/var/log/suricata.log` - IDS/IPS messages
`/var/log/openvpn*.log` - OpenVPN connection logs
`/tmp/rules.debug` - Active pf ruleset

For persistent issues, enable debug logging under System → Settings → Logging. Remember to disable after troubleshooting to avoid filling disk space.

Always check /tmp/rules.debug to see the actual pf ruleset loaded. This file shows the complete firewall configuration as interpreted by the system.

13 OPNsense MCP Server

> ★ THIS CHAPTER IS FOR AI/LLM AGENTS. The OPNsense MCP Server provides programmatic access to firewall management via Model Context Protocol. Use these tools instead of SSH/CLI when

The opnsense-mcp-server (by vespo92) provides MCP tools for managing OPNsense firewalls from AI agents like Claude. It supports both API and SSH access methods, enabling firewall rule management, VLAN configuration, NAT rules, and diagnostics.

Available MCP Tool Categories

Category	Prefix	Purpose
Connection	opnsense_configure, _test_connection	Setup and verify API access
VLANs	opnsense_list_vlans, _get_vlan, _create_vlan	VLAN management
Firewall Rules	opnsense_list_firewall_rules, _create_*, _delete_*	Rule CRUD operations
NAT	opnsense_nat_list_*, _create_*, _fix_dmz	NAT rule management
Interfaces	opnsense_get_interfaces, _interface_*	Interface config
DHCP/ARP	opnsense_list_dhcp_leases, _list_arp_*	Network discovery
DNS Blocking	opnsense_block_domain, _unblock_*	DNS blocklist management
HAProxy	opnsense_haproxy_*	Load balancer/reverse proxy
SSH Commands	opnsense_ssh_execute, _ssh_*	Direct CLI access
Routing	opnsense_routing_diagnostics, _fix_*	Inter-VLAN routing
Backups	opnsense_create_backup, _list_backups	Config backup/restore
Macros/IaC	opnsense_macro_*, _iac_*	Automation and infrastructure-as-code

Core Tools Reference

Connection & Discovery

Tool	Parameters	Returns
opnsense_test_connection	(none)	API connectivity status
opnsense_get_interfaces	(none)	All interfaces with status
opnsense_list_vlans	(none)	VLAN tag, interface, description
opnsense_list_dhcp_leases	interface (optional)	IP, MAC, hostname, lease time
opnsense_list_arp_entries	(none)	Full ARP table

Firewall Rules

Tool	Key Parameters	Notes
opnsense_list_firewall_rules	(none)	All rules with UUIDs
opnsense_get_firewall_rule	uuid	Single rule details
opnsense_create_firewall_rule	action, interface, direction, protocol, source, destination	Create new rule
opnsense_create_firewall_preset	preset, interface	Presets: allow-web, allow-ssh, block-all
opnsense_update_firewall_rule	uuid, + fields to update	Modify existing rule
opnsense_delete_firewall_rule	uuid	Remove rule
opnsense_toggle_firewall_rule	uuid	Enable/disable rule
opnsense_find_firewall_rules	description	Search by description

NAT Management

Tool	Purpose

opnsense_nat_list_outbound	List outbound NAT rules
opnsense_nat_list_port_forwards	List port forward rules
opnsense_nat_get_mode	Get NAT mode (auto/hybrid/manual)
opnsense_nat_set_mode	Set NAT mode
opnsense_nat_create_outbound_rule	Create outbound NAT rule
opnsense_nat_create_port_forward	Create port forward
opnsense_nat_fix_dmz	Add no-NAT rules for inter-VLAN
opnsense_nat_quick_fix_dmz	Quick DMZ NAT fix
opnsense_nat_analyze_config	Analyze NAT for issues

SSH/CLI Tools

When API tools are insufficient, use SSH tools for direct CLI access:

Tool	Purpose	Example Use
opnsense_ssh_execute	Run any command	pfctl -sr, netstat -rn
opnsense_ssh_show_pf_rules	Show packet filter rules	Verify loaded ruleset
opnsense_ssh_show_routing	Display routing table	Check route paths
opnsense_ssh_reload_firewall	Reload pf rules	Apply config changes
opnsense_ssh_fix_dmz_routing	Fix DMZ routing issues	Inter-VLAN problems
opnsense_ssh_check_nfs_connectivity	Test NFS from OPNsense	Verify NFS access
opnsense_ssh_system_status	Full system status	Health check
opnsense_ssh_batch_execute	Run multiple commands	Complex diagnostics

Common MCP Workflows

OPSEC Audit Workflow

- 1. opnsense_get_interfaces → Verify VPN tunnel (wg0) is UP
- 2. opnsense_nat_list_outbound → Confirm no WAN NAT for secure VLAN
- 3. opnsense_list_firewall_rules → Check VLAN isolation rules exist
- 4. opnsense_ssh_execute('pfctl -sr | grep vlan50') → Verify PF rules loaded
- 5. opnsense_ssh_execute('netstat -rn') → Check routing table

Troubleshooting Connectivity

- 1. opnsense_list_arp_entries → Check if device is seen on network
- 2. opnsense_find_device_by_name('hostname') → Find device IP/MAC
- 3. opnsense_list_firewall_rules → Check for blocking rules
- 4. opnsense_ssh_execute('pfctl -ss | grep ') → Check state table
- 5. opnsense_routing_diagnostics → Run inter-VLAN diagnostics

Create Firewall Rule

1. opnsense_list_vlans → Get interface names

2. opnsense_create_firewall_rule(
action='pass',
interface='opt2', # DMZ interface
direction='in',
protocol='tcp',
source='172.16.10.0/24',
destination='192.168.1.50',
destinationPort='2049',
description='DMZ to NFS'
)
3. opnsense_ssh_reload_firewall → Apply changes

HAProxy Tools (Reverse Proxy)

Tool	Purpose
opnsense_haproxy_service_control	start/stop/restart/reload/status
opnsense_haproxy_backend_list	List all backends
opnsense_haproxy_backend_create	Create backend with servers
opnsense_haproxy_frontend_list	List all frontends
opnsense_haproxy_frontend_create	Create frontend (listener)
opnsense_haproxy_acl_create	Create ACL for routing
opnsense_haproxy_action_create	Create routing action
opnsense_haproxy_certificate_list	List SSL certificates
opnsense_haproxy_stats	Get HAProxy statistics

The MCP server is actively being extended. Check for new tools periodically. Current repo: github.com/vespo92/opnsense-mcp-server (being forked/enhanced).

Complete MCP Tool Reference

The following tables document ALL available MCP tools organized by category. Use this as the authoritative reference for tool selection.

Connection & Configuration

Tool	Parameters	Description
opnsense_configure	host, apiKey, apiSecret, verifySsl	Configure OPNsense connection
opnsense_test_connection	(none)	Test API connectivity
opnsense_get_interfaces	(none)	List all network interfaces
opnsense_interface_list_overview	(none)	Interface overview with status
opnsense_interface_get_config	interfaceName	Get interface configuration

VLAN Management

Tool	Parameters	Description
opnsense_list_vlans	(none)	List all VLANs
opnsense_get_vlan	tag	Get VLAN details by tag
opnsense_create_vlan	interface, tag, description, pcp	Create new VLAN
opnsense_update_vlan	tag, description	Update VLAN description
opnsense_delete_vlan	tag	Delete a VLAN

Firewall Rules

Tool	Parameters	Description
opnsense_list_firewall_rules	(none)	List all firewall rules
opnsense_get_firewall_rule	uuid	Get rule by UUID

opnsense_create_firewall_rule	action, interface, direction, protocol, source_net, destination, destinationPort, description, enab	Create rule
opnsense_create_firewall_preset	preset, interface, source, destination, description	Create from preset (allow-web, allow-ssh, allow-minecr
opnsense_update_firewall_rule	uuid, + any field to update	Update existing rule
opnsense_delete_firewall_rule	uuid	Delete rule
opnsense_toggle_firewall_rule	uuid	Enable/disable rule
opnsense_find_firewall_rules	description	Search rules by description

NAT Management

Tool	Parameters	Description
opnsense_nat_list_outbound	(none)	List outbound NAT rules
opnsense_nat_list_port_forwards	(none)	List port forward rules
opnsense_nat_get_mode	(none)	Get NAT mode (auto/hybrid/manual/disabled)
opnsense_nat_set_mode	mode	Set NAT mode
opnsense_nat_create_outbound_rule	interface, source_net, destination_net, description, target, nonat, enabled	Create outbound NAT
opnsense_nat_delete_outbound_rule	uuid or description	Delete outbound NAT rule
opnsense_nat_create_port_forward	interface, protocol, destination_port, target, local_port, description, enabled	Create port forward
opnsense_nat_delete_port_forward	uuid	Delete port forward
opnsense_nat_fix_dmz	dmzNetwork, lanNetwork, otherInternalNetworks	Add NAT rules for inter-VLAN
opnsense_nat_quick_fix_dmz	(none)	Quick DMZ NAT fix
opnsense_nat_cleanup_dmz_fix	(none)	Remove MCP-created NAT fixes
opnsense_nat_analyze_config	(none)	Analyze NAT configuration
opnsense_nat_apply_changes	(none)	Apply NAT changes

DHCP & ARP Discovery

Tool	Parameters	Description
opnsense_list_dhcp_leases	interface (optional)	List DHCP leases
opnsense_find_device_by_name	pattern	Find device by hostname
opnsense_find_device_by_mac	mac	Find device by MAC address
opnsense_get_guest_devices	(none)	Get devices on guest VLAN
opnsense_get_devices_by_interface	(none)	Group devices by interface
opnsense_list_arp_entries	(none)	List all ARP entries
opnsense_find_arp_by_ip	ipPattern	Find ARP by IP/subnet
opnsense_find_arp_by_mac	macPattern	Find ARP by MAC
opnsense_find_arp_by_interface	interface	Find ARP by interface
opnsense_find_arp_by_hostname	pattern	Find ARP by hostname
opnsense_get_arp_stats	(none)	Get ARP statistics
opnsense_find_devices_on_vlan	vlanTag	Find devices on VLAN

DNS Blocklist Management

Tool	Parameters	Description
opnsense_list_dns_blocklist	(none)	List DNS blocklist entries
opnsense_block_domain	domain, description	Block a domain
opnsense_unblock_domain	domain	Unblock a domain
opnsense_block_multiple_domains	domains[], description	Block multiple domains
opnsense_apply_blocklist_category	category (adult/malware/ads/social)	Apply predefined category
opnsense_search_dns_blocklist	pattern	Search blocklist
opnsense_toggle_blocklist_entry	uuid	Enable/disable entry

HAProxy (Reverse Proxy/Load Balancer)

Tool	Parameters	Description
opnsense_haproxy_service_control	action (start/stop/restart/reload/status)	Control HAProxy service
opnsense_haproxy_backend_list	(none)	List all backends
opnsense_haproxy_backend_create	name, mode, balance, servers[], description	Create backend
opnsense_haproxy_backend_delete	uuid	Delete backend
opnsense_haproxy_backend_health	backend	Get backend health
opnsense_haproxy_frontend_list	(none)	List all frontends
opnsense_haproxy_frontend_create	name, bind, mode, backend, ssl, certificates[], tcpAccept[], description	Create frontend
opnsense_haproxy_frontend_delete	uuid	Delete frontend
opnsense_haproxy_acl_create	frontend, name, expression, value, negate	Create ACL
opnsense_haproxy_action_create	frontend, type, backend, condition, operator, actionNames[], value	Create action
opnsense_haproxy_certificate_list	(none)	List certificates

opnsense_haproxy_certificate_create	name, type, cn, san[], certificate, key, ca	Create certificate
opnsense_haproxy_stats	(none)	Get HAProxy statistics

SSH/CLI Tools

Tool	Parameters	Description
opnsense_ssh_execute	command, sudo, timeout	Execute arbitrary command
opnsense_ssh_fix_interface_blocking	interface	Fix interface blocking
opnsense_ssh_fix_dmz_routing	(none)	Comprehensive DMZ routing fix
opnsense_ssh_enable_intervlan_routing	(none)	Enable inter-VLAN routing
opnsense_ssh_reload_firewall	(none)	Reload pf rules
opnsense_ssh_show_routing	(none)	Show routing table
opnsense_ssh_show_pf_rules	verbose	Show packet filter rules
opnsense_ssh_backup_config	backupName	Backup configuration
opnsense_ssh_restore_config	backupPath	Restore configuration
opnsense_ssh_check_nfs_connectivity	targetIP	Check NFS connectivity
opnsense_ssh_system_status	(none)	Get system status
opnsense_ssh_test_vlan_connectivity	sourceInterface, targetIP	Test VLAN connectivity
opnsense_ssh_quick_dmz_fix	(none)	Quick DMZ fix
opnsense_ssh_batch_execute	commands[], stopOnError	Execute multiple commands

Routing & Diagnostics

Tool	Parameters	Description
opnsense_routing_diagnostics	sourceNetwork, destNetwork	Run inter-VLAN routing diagnostics

opnsense_routing_fix_all	(none)	Auto-fix all routing issues
opnsense_routing_fix_dmz	(none)	Quick fix DMZ to LAN routing
opnsense_routing_create_intervlan_rules	sourceNetwork, destNetwork, bidirectional	Create inter-VLAN firewall rules
opnsense_interface_enable_intervlan_routing	interfaceName	Enable routing on interface
opnsense_interface_enable_intervlan_all	(none)	Enable routing on all interfaces
opnsense_interface_configure_dmz	dmzInterface	Configure DMZ interface
opnsense_interface_update_config	interfaceName, enable, blockpriv, blockbogons, disablefilter	Update interface config

System Settings

Tool	Parameters	Description
opnsense_system_get_settings	(none)	Get firewall/routing settings
opnsense_system_enable_intervlan_routing	(none)	Enable inter-LAN traffic system-wide
opnsense_system_update_firewall_settings	allowinterlantraffic, blockbogons, bypassstaticroutes, disa	Update firewall settings

Backup & Restore

Tool	Parameters	Description
opnsense_create_backup	description	Create config backup
opnsense_list_backups	(none)	List available backups
opnsense_restore_backup	backupId	Restore from backup

Macros & Automation

Tool	Parameters	Description
opnsense_macro_start_recording	name, description	Start recording API calls

opnsense_macro_stop_recording	(none)	Stop and save macro
opnsense_macro_list	(none)	List saved macros
opnsense_macro_play	id, parameters, dryRun	Execute a macro
opnsense_macro_delete	id	Delete macro
opnsense_macro_analyze	id	Analyze macro patterns
opnsense_macro_generate_tool	id, save	Generate MCP tool from macro
opnsense_macro_export	path	Export macros to file
opnsense_macro_import	path, overwrite	Import macros from file

Infrastructure as Code (IaC)

Tool	Parameters	Description
opnsense_iac_plan_deployment	name, resources[], dryRun	Plan infrastructure deployment
opnsense_iac_apply_deployment	planId, autoApprove	Apply deployment plan
opnsense_iac_destroy_deployment	deploymentId, force	Destroy deployed resources
opnsense_iac_list_resource_types	category	List available resource types

CLI Execute Tools

Tool	Parameters	Description
opnsense_cli_execute	command, args[], timeout	Execute configctl command
opnsense_cli_fix_interface_blocking	interfaceName	Fix interface blocking via CLI
opnsense_cli_reload_firewall	(none)	Reload firewall rules
opnsense_cli_show_routing	(none)	Show routing table
opnsense_cli_fix_dmz_routing	(none)	Fix DMZ routing via CLI

opnsense_cli_check_nfs	truenasIP	Check NFS connectivity
opnsense_cli_apply_changes	(none)	Apply all configuration changes

Tool Selection Guidelines

Task	Preferred Tool	Avoid
List rules	opnsense_list_firewall_rules	SSH + pfctl
Check interface status	opnsense_get_interfaces	SSH + ifconfig
Find device by name	opnsense_find_device_by_name	SSH + grep
View raw pf rules	opnsense_ssh_show_pf_rules	Reading /tmp/rules.debug
Complex diagnostics	opnsense_routing_diagnostics	Manual SSH commands
Bulk operations	opnsense_ssh_batch_execute	Multiple SSH calls
Create rule	opnsense_create_firewall_rule	SSH + config edit

General Principle: Use MCP API tools when available. Fall back to SSH tools only for operations not covered by API tools or when you need raw CLI output.

14 VLAN Configuration

VLANs (Virtual LANs) segment a physical network into isolated broadcast domains. OPNsense supports 802.1Q VLAN tagging, allowing a single physical interface to carry traffic for multiple logical networks. This is essential for network segmentation, security zones, and efficient use of hardware.

VLAN Concepts

Term	Description
VLAN Tag	Numeric ID (1-4094) added to Ethernet frames
Trunk Port	Switch port carrying multiple VLANs (tagged)
Access Port	Switch port for single VLAN (untagged)
Native VLAN	Untagged VLAN on a trunk port
Parent Interface	Physical NIC that carries VLAN traffic
VLAN Interface	Virtual interface for specific VLAN tag

! VLANs require a managed switch configured with matching VLAN tags. Unmanaged switches cannot process VLAN tags and will drop tagged traffic.

Creating VLANs in OPNsense

- 1. Interfaces → Other Types → VLAN → Add
- 2. Select Parent Interface (physical NIC connected to trunk port)
- 3. Set VLAN Tag (must match switch configuration)
- 4. Set VLAN Priority (0-7, optional QoS)
- 5. Add Description (e.g., 'DMZ', 'Guest', 'IoT')
- 6. Save and Apply
- 7. Interfaces → Assignments → Assign the new VLAN
- 8. Configure the assigned interface (IP, DHCP, etc.)
- 9. Enable the interface

- 10. Add firewall rules for the new VLAN interface

VLAN Configuration Fields

Field	Value	Notes
Parent	igc0, em0, etc.	Physical interface to trunk port
VLAN Tag	1-4094	Must match switch VLAN ID
VLAN Priority	0-7	802.1p QoS (0=best effort, 7=highest)
Description	Text	Identifies VLAN in UI

Common VLAN Topologies

VLAN	Tag	Subnet	Purpose	Internet Access
LAN	Native	192.168.1.0/24	Trusted devices	Full + local
DMZ	10	172.16.10.0/24	Servers/services	Full, limited local
Guest	20	192.168.20.0/24	Visitors	Internet only
IoT	30	192.168.30.0/24	Smart devices	Internet only
Management	99	192.168.99.0/24	Network gear	No internet
Lab	50	192.168.50.0/24	Testing/dev	Isolated or VPN

Inter-VLAN Routing

By default, VLANs cannot communicate with each other. OPNsense acts as the router between VLANs. To allow inter-VLAN traffic, create firewall rules on each VLAN interface permitting traffic to other VLAN subnets.

Inter-VLAN Rule Example (DMZ → LAN NFS)

Field	Value
Interface	DMZ (opt2 or vlan10)
Direction	In
Action	Pass
Protocol	TCP
Source	DMZ net (172.16.10.0/24)
Destination	192.168.1.50 (file server)
Destination Port	2049 (NFS)
Description	DMZ to LAN NFS access

✓ For security, create specific allow rules rather than broad 'allow all' between VLANs. Block RFC1918 by default, then add exceptions for required services.

VLAN Isolation Patterns

Pattern	Rule Strategy	Use Case
Full Isolation	Block RFC1918, allow internet	Guest, IoT
Server Access	Allow specific ports to LAN servers	DMZ
Management	Allow from admin VLAN only	Network gear
VPN Only	Block WAN, allow VPN gateway	C2, pentest

Switch Configuration (Example: UniFi)

The switch must be configured to trunk VLANs to OPNsense. Example UniFi port config:

Port	Profile	Native VLAN	Tagged VLANs	Connected To

1	Trunk	1 (LAN)	6,10,40,50	OPNsense
2	Access	40 (IoT)	-	IoT device
3	Access	10 (Guest)	-	WiFi AP
4	Access	50 (C2)	-	C2 server

VLAN Troubleshooting

- **No connectivity**: Check switch VLAN config matches OPNsense tags
- **Can ping gateway but not internet**: Check outbound NAT includes VLAN subnet
- **Inter-VLAN blocked**: Verify firewall rules on SOURCE interface
- **DHCP not working**: Enable DHCP server on VLAN interface
- **Asymmetric routing**: Ensure devices use OPNsense as gateway, not switch

VLAN Verification Commands

`ifconfig vlan10` - Check VLAN interface status

`netstat -rn | grep 172.16` - Verify route to VLAN subnet

`tcpdump -i igc0 -e vlan` - Capture tagged traffic on parent

`pfctl -sr | grep opt2` - Check firewall rules for VLAN interface

15 Certificates & PKI

Certificates enable encrypted communications (HTTPS, VPN) and authentication. OPNsense includes a full Certificate Authority (CA) for generating and managing X.509 certificates without

Certificate Types Overview

Type	Purpose	Location
Certificate Authority (CA)	Signs other certificates, establishes trust chain	System → Trust → Authorities
Server Certificate	Authenticates server (HTTPS, VPN server)	System → Trust → Certificates
Client Certificate	Authenticates user/device to server	System → Trust → Certificates
Intermediate CA	Sub-CA signed by root CA	System → Trust → Authorities
ACME/Let's Encrypt	Free public TLS certificates	Services → ACME → Certificates
CSR (Signing Request)	Request external CA to sign	System → Trust → Certificates

Creating a Certificate Authority

A local CA allows you to sign your own server and client certificates. Required for OpenVPN, IPsec with certificates, and internal HTTPS services.

- 1. Navigate to System → Trust → Authorities
- 2. Click + Add
- 3. Method: Create an internal Certificate Authority
- 4. Descriptive name: e.g., 'OPNsense-CA'
- 5. Key type: RSA (2048/4096) or ECDSA (prime256v1/secp384r1)
- 6. Digest Algorithm: SHA256 or SHA384
- 7. Lifetime: 3650 days (10 years) for CA
- 8. Common Name: e.g., 'OPNsense Internal CA'
- 9. Fill Country, State, City, Organization (optional but recommended)

- 10. Save

> ✓ KEY TYPE SELECTION: ECDSA (prime256v1) offers faster operations and smaller keys with equivalent security to RSA-3072. Use ECDSA for new deployments unless compatibility with legacy systems

CA Configuration Options

Option	Recommended	Description
Key Type	ECDSA prime256v1	Fastest, modern security
Key Type (legacy)	RSA 4096	Maximum compatibility
Digest	SHA256	Standard, widely supported
Digest (high security)	SHA384	For sensitive environments
Lifetime (CA)	3650 days	10 years for root CA
Lifetime (server)	397 days	Max for browser trust
Lifetime (client)	365 days	Annual renewal recommended

Creating Server Certificates

Server certificates authenticate OPNsense services (WebUI, OpenVPN server, HAProxy).

- 1. Navigate to System → Trust → Certificates
- 2. Click + Add
- 3. Method: Create an internal Certificate
- 4. Descriptive name: e.g., 'opnsense-webui' or 'vpn-server'
- 5. Certificate Authority: Select your CA
- 6. Type: Server Certificate
- 7. Key type: Match CA (ECDSA or RSA)
- 8. Digest: SHA256
- 9. Lifetime: 397 days (browser max) or 365 days
- 10. Common Name: FQDN or IP (e.g., 'fw.example.com')
- 11. Alternative Names: Add DNS/IP SANs for all access methods
- 12. Save

! SUBJECT ALTERNATIVE NAMES (SANs): Modern browsers require SANs for certificate validation. Add all hostnames and IPs used to access the service: DNS:fw.example.com, DNS:opnsense.local,

Creating Client Certificates

Client certificates authenticate users/devices to VPN or other services.

- 1. System → Trust → Certificates → Add
- 2. Method: Create an internal Certificate
- 3. Type: Client Certificate
- 4. Certificate Authority: Select your CA
- 5. Common Name: User identifier (e.g., 'john.doe@example.com')
- 6. Lifetime: 365 days (enforce annual renewal)
- 7. Save
- 8. Export: Download .p12 (PKCS#12) bundle for client import

ACME / Let's Encrypt Integration

The ACME plugin (os-acme-client) automates free TLS certificate issuance from Let's Encrypt or other ACME providers. Certificates auto-renew before expiration.

Plugin Installation

- 1. System → Firmware → Plugins
- 2. Search 'os-acme-client'
- 3. Click + to install
- 4. After install, find at Services → ACME Certificates

ACME Configuration Steps

- 1. Services → ACME → Settings: Enable plugin
- 2. Services → ACME → Accounts: Add Let's Encrypt account (email required)
- 3. Services → ACME → Challenge Types: Configure DNS or HTTP challenge
- 4. Services → ACME → Certificates: Add certificate with domain(s)
- 5. Services → ACME → Automations: Link to services (HAProxy, WebUI)

ACME Challenge Types

Challenge	Requirements	Best For
HTTP-01	Port 80 open to internet, web server	Simple setups, HAProxy
DNS-01	DNS API access (Cloudflare, Route53, etc.)	Wildcards, no port 80
TLS-ALPN-01	Port 443 open, ALPN support	When port 80 blocked

✓ DNS-01 CHALLENGE: Required for wildcard certificates (*.example.com). Supported DNS providers include Cloudflare, DigitalOcean, AWS Route53, Azure DNS, Google Cloud DNS, and many

Certificate Usage in Services

Service	Certificate Setting Location	Certificate Type
Web GUI	System → Settings → Administration → SSL Certificate	Server (internal or ACME)
OpenVPN Server	VPN → OpenVPN → Servers → Server Certificate	Server + CA
OpenVPN Client Auth	VPN → OpenVPN → Servers → Client Certificate	Client + CA
IPsec	VPN → IPsec → Tunnel Settings → My Certificate	Server + CA
HAProxy Frontend	Services → HAProxy → Real Servers → SSL options	Server (ACME recommended)
Captive Portal	Services → Captive Portal → Zone → Certificate	Server
FreeRADIUS	Services → FreeRADIUS → EAP → Certificate	Server + CA

Certificate Renewal & Management

Certificate Type	Renewal Method	Notes
Internal CA	Manual before expiry	Re-sign all child certs when CA expires
Internal Server/Client	Manual or script	System → Trust → Certificates → Reissue

ACME (Let's Encrypt)	Automatic (cron)	Renews ~30 days before expiry
Imported External	Manual replacement	Upload new cert before expiry

! CERTIFICATE EXPIRY MONITORING: OPNsense Dashboard shows certificate expiration. For ACME, check Services → ACME → Log for renewal status. Consider external monitoring (Wazuh, Nagios)

Certificate Export & Import

Format	Extension	Use Case
PEM (Certificate)	.crt, .pem	Most services, OpenVPN, HAProxy
PEM (Private Key)	.key	Private key (keep secure!)
PKCS#12	.p12, .pfx	Windows, iOS, combined cert+key
DER	.der, .cer	Java keystores, some Windows apps
CA Bundle	.crt	Full chain for client verification

- Export Certificate: System → Trust → Certificates → Click cert → Export (PEM or P12)
- Export CA: System → Trust → Authorities → Click CA → Export
- Import: System → Trust → Certificates → Add → Method: Import
- P12 Password: Required when exporting PKCS#12, protects private key

Certificate Revocation (CRL)

Revoke compromised or decommissioned certificates to prevent unauthorized access.

- 1. System → Trust → Revocation: Create CRL for your CA
- 2. Add revoked certificate serial numbers to CRL
- 3. Configure OpenVPN/services to check CRL
- 4. Distribute updated CRL to clients if needed

MCP Tools for Certificate Management

The OPNsense MCP Server includes HAProxy certificate tools. For CA/Trust management, use the REST API or SSH commands.

Tool	Purpose
opnsense_haproxy_certificate_list	List available certificates for HAProxy
opnsense_haproxy_certificate_create	Create/import certificate (selfsigned, import, acme)
opnsense_ssh_execute + configctl	Trigger ACME renewal via SSH

Example: ACME Renewal via MCP SSH

```
opnsense_ssh_execute(command='configctl acme renew')
```

Certificate Quick Reference

Task	Location / Command
Create CA	System → Trust → Authorities → Add
Create Server Cert	System → Trust → Certificates → Add (Type: Server)
Create Client Cert	System → Trust → Certificates → Add (Type: Client)
Install ACME Plugin	System → Firmware → Plugins → os-acme-client
Configure ACME	Services → ACME → Settings/Accounts/Certificates
Assign WebUI Cert	System → Settings → Administration → SSL Certificate
Revoke Certificate	System → Trust → Revocation → Add to CRL
Export P12	System → Trust → Certificates → Export → PKCS#12
Check Expiry	Dashboard widget or System → Trust → Certificates
Force ACME Renew	configctl acme renew (via SSH)

A Quick Reference

★ STRUCTURED REFERENCE DATA for rapid LLM lookup. Use this section for quick facts without reading full chapters.

Defaults & Credentials

Item	Default	Notes
Web UI URL	https://192.168.1.1	LAN IP, HTTPS required
Username	root	Create admin user, disable root
Password	opnsense	CHANGE IMMEDIATELY
SSH	Disabled	Enable at System → Settings → Administration
API	Disabled	Enable per-user at System → Access → Users

Common Ports

Service	Port	Protocol	Notes
Web GUI	443	TCP/HTTPS	Anti-lockout on LAN
SSH	22	TCP	Disabled by default
DNS	53	TCP/UDP	Unbound resolver
DHCP	67-68	UDP	Server on 67, client on 68
IPsec IKE	500	UDP	Key exchange
IPsec NAT-T	4500	UDP	NAT traversal
OpenVPN	1194	UDP (or TCP)	Configurable

WireGuard	51820	UDP	Configurable
NTP	123	UDP	Time sync
Syslog	514	UDP	Remote logging
SNMP	161	UDP	Monitoring

File Locations

Path	Purpose	Access
/conf/config.xml	Master config (XML)	READ ONLY - use API
/tmp/rules.debug	Active pf ruleset	pfctl -sr equivalent
/var/log/filter.log	Firewall log	tail -f for live view
/var/log/system.log	System messages	General diagnostics
/var/log/latest.log	Recent entries	Quick check
/var/log/suricata/	IDS/IPS logs	Alert analysis
/var/log/openvpn/	VPN logs	Connection issues
/usr/local/etc/	Service configs	Unbound, Suricata, etc.

Essential CLI Commands

Command	Purpose
pfctl -sr	Show firewall rules
pfctl -ss	Show state table
pfctl -si	Show statistics
pfctl -k <ip>	Kill states for IP
pfctl -F states	Flush ALL states (caution!)

netstat -rn	Routing table
netstat -an	All connections
sockstat -l	Listening ports
configctl filter reload	Reload firewall
configctl interface reconfigure	Reconfigure interfaces
configctl webgui restart	Restart web GUI
configctl service restart all	Restart all services
opnsense-update -c	Check for updates
opnsense-revert	Revert to previous version

Interface Naming

Pattern	Meaning	Example
em0, em1	Intel e1000 NICs	em0 = WAN, em1 = LAN
igc0, igc1	Intel I225/I226 2.5GbE	Modern Intel NICs
igb0, igb1	Intel I350/I210 GbE	Server-class Intel
re0, re1	Realtek NICs	Consumer boards
ue0	USB Ethernet	RTL8153 adapter
vtnet0	VirtIO (VM)	Proxmox/KVM
vmx0	VMware VMXNET3	ESXi/VMware
vlanX	VLAN interface	vlan10 = VLAN 10
gif0	GIF tunnel	IPv6 tunneling
gre0	GRE tunnel	Generic encapsulation
wg0, wg1	WireGuard	VPN tunnel
ovpnc1	OpenVPN client	Client instance 1
ovpns1	OpenVPN server	Server instance 1
ipsec1000	IPsec VTI	Virtual tunnel interface

Firewall Rule Quick Facts

Fact	Value
Default action	Block (implicit deny)
Rule direction	Inbound to interface (where traffic enters firewall)
Quick flag	Enabled by default (first match wins)
Floating priority	Processed before interface rules

State tracking	Enabled by default (stateful)
Log default	Disabled (enable per-rule)
Anti-lockout	Auto-created for LAN to GUI (443)
Bogon blocking	Block RFC1918 on WAN by default

NAT Quick Facts

Fact	Value
Processing order	NAT rules before firewall rules
Default outbound	Automatic mode (auto-generates rules)
Port forward	Creates associated filter rule if enabled
1:1 NAT	Bidirectional static mapping
NPTv6	IPv6 prefix translation (not address)
Reflection	Access internal services via external IP from LAN

VPN Quick Reference

VPN Type	Ports	Key Config
IPsec IKEv2	UDP 500, 4500	Phase 1 + Phase 2 must match both sides
OpenVPN	UDP 1194 (default)	Cert + CA required, client export plugin
WireGuard	UDP 51820 (default)	Public key exchange, AllowedIPs = routing

Troubleshooting Decision Tree

- Traffic blocked? → Check rule order, enable logging, verify states
- No internet? → Check WAN status, gateway, outbound NAT mode
- DNS failing? → Verify Unbound running, check forwarding/DoT

- VPN down? → Match phase settings, check ports, review logs
- Slow? → Check CPU, interface errors, traffic shaper
- CARP split-brain? → Check VHID uniqueness, sync interface
- Can't access GUI? → SSH in, check webconfigurator, anti-lockout

B pf Rule Syntax Reference

This appendix explains native pf (packet filter) syntax for interpreting /tmp/rules.debug and pfctl -sr output. Essential for debugging firewall issues.

Basic Rule Structure

pf rules follow a specific syntax. Understanding this helps interpret the actual loaded ruleset versus the GUI representation:

```
action [direction] [log] [quick] on interface [inet|inet6] proto protocol from source to destination [flags] [state]
```

Rule Keywords

Keyword	Values	Description
pass	-	Allow traffic through
block	drop, return, return-rst, return-icmp	Deny traffic (default: drop)
match	-	Match for NAT/QoS without pass/block
in	-	Inbound traffic (entering interface)
out	-	Outbound traffic (leaving interface)
log	-	Log matching packets
quick	-	Stop processing on match (first match wins)
on	interface name	Apply to specific interface
inet	-	IPv4 traffic
inet6	-	IPv6 traffic

proto	tcp, udp, icmp, etc.	Layer 4 protocol
from	address/mask	Source address
to	address/mask	Destination address
port	number or range	TCP/UDP port
flags	S/SA, etc.	TCP flags to match
keep state	-	Track connection state

Example Rules Explained

Allow LAN to Internet:

```
pass in quick on em1 inet from 192.168.1.0/24 to any keep state
```

Component	Meaning
pass	Allow traffic
in	Traffic entering the firewall on this interface
quick	Stop processing if matched
on em1	LAN interface
inet	IPv4 only
from 192.168.1.0/24	Source: LAN subnet
to any	Destination: anywhere
keep state	Track connection, allow return traffic

Block with TCP RST:

```
block return-rst in quick on em1 inet proto tcp from any to em1 port 22
```

Component	Meaning

block return-rst	Block and send TCP RST to client
proto tcp	TCP protocol only
to em1 port 22	To interface's IP, SSH port

Port Forward (NAT + Filter):

```
rdr on em0 inet proto tcp from any to (em0) port 443 -> 192.168.1.10 port 443
```

Component	Meaning
rdr	Redirect (destination NAT)
on em0	WAN interface
(em0)	WAN interface's current IP
-> 192.168.1.10	Redirect to internal server

Address Specifications

Syntax	Meaning	Example
any	Any address	from any
IP/mask	CIDR notation	192.168.1.0/24
(interface)	Interface's current IP	(em0)
<table>	Address table/alias	<lan_servers>
! address	NOT this address	from ! 10.0.0.1
self	All firewall IPs	to self
IP - IP	Range (tables only)	10.0.0.1 - 10.0.0.50

Port Specifications

Syntax	Meaning	Example
port 80	Single port	to any port 80
port 80:443	Port range	to any port 80:443
port { 80 443 }	Port list	port { 22 80 443 }
port > 1023	Greater than	from any port > 1023
port 1:1023	Privileged ports	port 1:1023

State Options

Option	Effect
keep state	Normal stateful tracking (default)
modulate state	Randomize TCP sequence numbers
synproxy state	Proxy TCP handshake (DDoS protection)
no state	Stateless rule (no tracking)
sloppy state	Relaxed state tracking (asymmetric routing)
max 100	Maximum states from this rule
max-src-nodes 10	Max unique source IPs
max-src-states 3	Max states per source IP

Reading pfctl Output

Use these commands to view the active ruleset and diagnose issues:

Command	Output
pfctl -sr	Show all loaded rules
pfctl -sr -v	Verbose with rule numbers

pfctl -sr -vv	Very verbose with counters
pfctl -ss	Show state table
pfctl -ss | grep 172.16	States for specific subnet
pfctl -si	Statistics (packets, states)
pfctl -sa	Everything (rules + states + info)
pfctl -t tablename -T show	Show table contents

The file /tmp/rules.debug contains the ruleset BEFORE loading.
Compare with 'pfctl -sr' to verify rules loaded correctly.
Syntax errors prevent loading.

OPNsense Rule Anchors

OPNsense organizes rules into anchors (named rule groups). Key anchors:

Anchor	Purpose
opnsense/*	Main OPNsense rules
opnsense/floating	Floating rules
opnsense/interface/lan	LAN interface rules
opnsense/nat/rdr	Port forward rules
opnsense/nat/out	Outbound NAT rules
opnsense/suricata	IDS/IPS inline rules

C REST API Reference

The OPNsense REST API enables programmatic firewall management. Use this for automation, integration with other tools, or when MCP tools are unavailable.

API Authentication

OPNsense uses API key + secret authentication. Generate credentials at System → Access → Users → [user] → API Keys.

Component	Details
Auth Type	HTTP Basic Authentication
Username	API Key (long alphanumeric string)
Password	API Secret (long alphanumeric string)
Protocol	HTTPS only (port 443)
Content-Type	application/json

Example curl Request

```
curl -k -u 'API_KEY:API_SECRET' https://192.168.1.1/api/core/system/status
```

Common API Endpoints

Endpoint	Method	Description
/api/core/system/status	GET	System status and version
/api/core/firmware/status	GET	Firmware update status

/api/diagnostics/interface/getInterfaceStatistics	GET	Interface stats
/api/firewall/filter/searchRule	GET	List firewall rules
/api/firewall/filter/getRule/{uuid}	GET	Get specific rule
/api/firewall/filter/addRule	POST	Create firewall rule
/api/firewall/filter/setRule/{uuid}	POST	Update rule
/api/firewall/filter/delRule/{uuid}	POST	Delete rule
/api/firewall/filter/apply	POST	Apply pending changes
/api/firewall/alias/searchItem	GET	List aliases
/api/firewall/alias/addItem	POST	Create alias

NAT API Endpoints

Endpoint	Method	Description
/api/firewall/source_nat/searchRule	GET	List outbound NAT
/api/firewall/source_nat/addRule	POST	Create outbound rule
/api/firewall/source_nat/apply	POST	Apply NAT changes
/api/firewall/destination_nat/searchRule	GET	List port forwards
/api/firewall/destination_nat/addRule	POST	Create port forward

Interface API Endpoints

Endpoint	Method	Description
/api/interfaces/overview/export	GET	All interfaces overview
/api/interfaces/vlan_settings/searchItem	GET	List VLANs
/api/interfaces/vlan_settings/addItem	POST	Create VLAN

/api/diagnostics/interface/getArp	GET	ARP table
/api/dhcpv4/leases/searchLease	GET	DHCP leases

API Workflow: Create Firewall Rule

- 1. Generate API credentials (System → Access → Users → API Keys)
- 2. GET /api/firewall/filter/searchRule to list existing rules
- 3. POST /api/firewall/filter/addRule with rule JSON body
- 4. POST /api/firewall/filter/apply to activate changes
- 5. GET /api/firewall/filter/searchRule to verify

Rule JSON Structure

Example POST body for /api/firewall/filter/addRule:

Field	Type	Example Value
enabled	string	1
action	string	pass
interface	string	lan
direction	string	in
ipprotocol	string	inet
protocol	string	tcp
source_net	string	192.168.1.0/24
destination_net	string	any
destination_port	string	443
description	string	Allow HTTPS

! Always call /apply endpoint after making changes. Changes are staged until applied. This allows batching multiple changes before activation.

API Response Codes

Code	Meaning	Action
200	Success	Parse response JSON
400	Bad Request	Check request format/parameters
401	Unauthorized	Verify API key/secret
403	Forbidden	User lacks permission
404	Not Found	Check endpoint URL
500	Server Error	Check OPNsense logs

API Best Practices

- Always use HTTPS (-k flag for self-signed certs)
- Store credentials securely (environment variables, not code)
- Check response status before parsing JSON
- Call /apply after modifications
- Use UUIDs from search responses for updates/deletes
- Rate limit requests to avoid overwhelming the firewall
- Test in non-production environment first

API Documentation: Full API reference available at https://docs.opnsense.org/development/api.html

---

This guide is optimized for LLM/AI agent reference. Based on OPNsense 24.x documentation. For latest info: docs.opnsense.org | MCP Server: github.com/vespo92/opnsense-mcp-server