EN

Quick Reference Guide

Automated OPNsense Backup

3-2-1 strategy, Git, S3 and DR < 15 min

Mars 2026

Table of Contents

Why Back Up Your OPNsense Config

OPNsense stores its entire configuration in a single XML file: /conf/config.xml. This file contains your firewall rules, VLANs, VPN tunnels, SSL certificates, users, and aliases — potentially days of work condensed into a few kilobytes.

Without a backup, a disk crash, failed update, system corruption, or human error can wipe your entire network configuration in seconds. With 200+ firewall rules and a dozen VLANs, you're looking at an entire day of work.

What to back up:

* config.xml — Complete configuration: rules, interfaces, VPN, certificates, users

* Installed packages — Plugin list (os-zerotier, os-wireguard, os-acme...) for quick reinstall

* RRD data — Network performance graphs (optional)

* Additional certificates — Included in config.xml but export separately if used elsewhere

i OPNsense exposes a complete REST API to fully automate this — no third-party plugins, no SSH required.

Manual Backup via the OPNsense Interface

In the OPNsense interface, navigate to Diagnostics -> Backup. Three main options:

* Download configuration as XML: downloads the complete config.xml, encrypted or not

* Include extra data: includes RRD data (performance graphs)

* Include package info: lists installed packages for documentation

```
# One-shot backup via OPNsense API
curl -k -u 'your-apikey:your-apisecret' \
  -o opnsense-backup-$(date +%Y%m%d).xml \
  'https://192.168.1.1/api/core/backup/download/this'
```

Automating with the OPNsense REST API

First, create an API key under System -> Access -> Users -> [your user] -> API keys. The secret is only shown once — store it securely.

```
# Essential endpoints
GET  /api/core/backup/download/this      # download current config
GET  /api/core/backup/list               # list internal backups
POST /api/core/backup/revertto/{id}      # restore a backup by ID
POST /api/core/backup/deleteback/{id}    # delete a backup by ID

# Authentication: HTTP Basic Auth (apikey:apisecret)
# Never use the admin password — API keys are individually revocable
```

Daily Backup Script to Private Git Repo

Storing OPNsense config in a private Git repo provides: full history, readable diffs between versions, instant rollback, and compliance audit trail.

```
#!/bin/bash
# /usr/local/bin/opnsense-backup.sh
set -euo pipefail

OPNSENSE_HOST='192.168.1.1'
API_KEY='your-api-key'
API_SECRET='your-api-secret'
BACKUP_DIR='/opt/opnsense-backups'
GIT_REPO='git@github.com:yourorg/opnsense-configs.git'
DATE=$(date +%Y-%m-%d_%H-%M)

if [ ! -d "$BACKUP_DIR/.git" ]; then
  git clone "$GIT_REPO" "$BACKUP_DIR"
fi
cd "$BACKUP_DIR" && git pull --rebase origin main

curl -sk -u "${API_KEY}:${API_SECRET}" \
  "https://${OPNSENSE_HOST}/api/core/backup/download/this" \
  -o "config-${DATE}.xml"

if ! grep -q '<opnsense>' "config-${DATE}.xml" 2>/dev/null; then
  echo "ERROR: invalid config.xml"; rm -f "config-${DATE}.xml"; exit 1
fi

ln -sf "config-${DATE}.xml" config-latest.xml
git add -A
if ! git diff --staged --quiet; then
  git commit -m "backup: ${DATE} [auto]"
  git push origin main
fi

# Rotate: keep last 30 XML files
ls -t config-20*.xml 2>/dev/null | tail -n +31 | xargs -r rm -f
git add -A && git diff --staged --quiet || git commit -m "rotation: cleanup"
git push origin main 2>/dev/null || true

# crontab -e (as root)
0 2 * * * /usr/local/bin/opnsense-backup.sh >> /var/log/opnsense-backup.log 2>&1
```

Remote Storage Backup

Option A — Amazon S3 (or Wasabi, MinIO)

```
apt install awscli -y && aws configure

# Add to opnsense-backup.sh after the Git commit:
aws s3 cp "config-${DATE}.xml" \
  s3://my-backup-bucket/opnsense/config-${DATE}.xml \
  --storage-class STANDARD_IA
# S3 Lifecycle policy: 90-day retention
```

Option B — SFTP to NAS or Remote Server

```
ssh-keygen -t ed25519 -f ~/.ssh/backup_opnsense -N ''
ssh-copy-id -i ~/.ssh/backup_opnsense.pub backup@nas.local

# In opnsense-backup.sh:
sftp -i ~/.ssh/backup_opnsense -b - backup@nas.local << EOF
  put config-${DATE}.xml /backups/opnsense/
  bye
EOF
```

Option C — Proxmox Backup Server (PBS)

If OPNsense runs as a Proxmox VM, PBS can back up the entire VM via Proxmox snapshots. Full restore in 5 minutes.

```
# Proxmox: configure backup job for the OPNsense VM
# Datacenter -> Backup -> Add job
# Storage: PBS, Mode: snapshot, Schedule: daily 03:00
```

3-2-1 Strategy Applied to OPNsense

The 3-2-1 strategy is the industry standard for professional backups:

* 3 copies: local config.xml + private Git repo + S3 bucket

* 2 different media: local disk + cloud object storage

* 1 offsite: S3 in a different region (or remote SFTP server)

```
# Recommended schedule
Local Git backup    : daily at 2:00 AM (cron)
S3 push             : embedded in Git script (automatic)
PBS/Proxmox snapshot: weekly (Sunday 3:00 AM)
Restore test        : quarterly

# Estimated monthly cost: < $3/month on S3 Standard-IA for 90 days
```

Testing the Restore: Complete Procedure

An untested backup is a useless backup. Test a complete restore at least once per quarter on a test OPNsense instance.

Method 1 — Via the Web Interface

1. Diagnostics -> Backup -> Restore section

2. Browse -> select the saved config.xml file

3. Check "Reboot after restore" -> Restore configuration

4. Wait for reboot (~2 min) -> verify rules, VLANs, VPN

Method 2 — Via REST API

```
curl -k -u 'apikey:apisecret' \
  -X POST \
  -F 'conffile=@/opt/opnsense-backups/config-latest.xml' \
  'https://192.168.1.1/api/core/backup/restore'

curl -k -u 'apikey:apisecret' \
  'https://192.168.1.1/api/core/firmware/status'
```

Post-restore checklist: firewall rules active, VLANs routed, WireGuard tunnels active, DHCP working, Unbound DNS, SSL certificates valid.

Ansible Integration — Disaster Recovery < 15 min

Combining automated backups with Ansible playbooks enables complete disaster recovery in under 15 minutes, even on new hardware.

```yaml
# playbook-opnsense-dr.yml
---
- name: OPNsense Disaster Recovery
  hosts: opnsense_new
  gather_facts: false
  vars:
    backup_source: 'git@github.com:yourorg/opnsense-configs.git'
    backup_file: 'config-latest.xml'

  tasks:
    - name: Clone backup repo
      git:
        repo: '{{ BACKUP_SOURCE }}'
        dest: /tmp/opnsense-backup
        version: main

    - name: Restore OPNsense configuration
      uri:
        url: 'https://{{ INVENTORY_HOSTNAME }}/api/core/backup/restore'
        method: POST
        url_username: '{{ API_KEY }}'
        url_password: '{{ API_SECRET }}'
        body_format: form-multipart
        body:
          conffile:
            content: '{{ LOOKUP_FILE }}'
            filename: 'config.xml'
        validate_certs: false

    - name: Wait for OPNsense reboot
      wait_for:
        host: '{{ INVENTORY_HOSTNAME }}'
        port: 443
        delay: 30
        timeout: 300

    - name: Verify firewall rules
      uri:
        url: 'https://{{ INVENTORY_HOSTNAME }}/api/firewall/filter/searchrule'
        url_username: '{{ API_KEY }}'
        url_password: '{{ API_SECRET }}'
        validate_certs: false
      register: rules_check

    - name: Final report
      debug:
        msg: "OPNsense restored with {{ RULES_COUNT }} firewall rules"
```

DR timeline: 0-5 min new OPNsense install | 5-8 min ansible-playbook restore | 8-12 min automated verification | 12-15 min manual tests. Target: < 15 minutes.

Conclusion: Your Network Safety Net

Automated XML backup via REST API + Git versioning + S3 replication + 3-2-1 strategy + Ansible DR: your OPNsense configuration becomes resilient infrastructure.

Implementation time: 2-3 hours. The payoff: even in a disaster scenario, the network is back online in under 15 minutes.

OPNsense Enterprise Stack — Navigation

* Article 1 : Install OPNsense in Proxmox
* Article 2 : VLANs & Zero Trust
* Article 3 : WireGuard VPN & LTE failover
* Article 4 : WiFi & APs per VLAN
* Article 5 : CrowdSec + fail2ban IDS/IPS
* Article 6 : NAC with FreeRADIUS & 802.1X
* Article 7 : Suricata IDS/IPS
* Article 8 : AdGuard Home + DNS over HTTPS
* Article 9 : Monitoring Grafana + InfluxDB
* Article 10 : CARP & High Availability
* Article 11 : Lightweight SIEM with Wazuh
* Article 12 : Ansible as Code
* Article 13 (this article) : Automated OPNsense Backup

i Hub: blog.botum.ca/opnsense-enterprise-network-stack-overview/

Full article: blog.botum.ca/opnsense-automated-config-backup

Website: www.botum.ca • contact@botum.ca