Network Hardening

> Last Verified: 2026-02-17 | Owner: John

Network Hardening — Consolidated Guide

Last Updated: 2026-02-10 **System:** Darwin 25.2.0 (M3 Mac Studio, 96GB RAM) **Status:** Configuration Complete, Deployment Blocked (see below) **Original Docs:** Consolidated from 2 separate files (2026-01-31)

Overview

Comprehensive network hardening and firewall lockdown configuration for macOS. Implements defense-in-depth strategy with application firewall, service disablement, and local-only access controls.

Hardening Strategy

Core Principles

 Default-Deny: Block all external traffic by default **Local-Only:** Services bind to 127.0.0.1 or RFC 1918 (LAN) only **Minimal Attack Surface:** Disable unnecessary services **Defense-in-Depth:** Multiple layers of protection **Logging:** Track all network access attempts

Scope

- macOS Application Layer Firewall (ALF)
- SSH, VNC, File Sharing (remote access)
- Bluetooth, AirDrop (wireless)

- Service port bindings (localhost vs. 0.0.0.0)
- System Integrity Protection (SIP)

Current Hardening Status

? Already Secured

- **System Integrity Protection (SIP):** Enabled (verified)
- **Screen Sharing (VNC):** Disabled
- **File Sharing (AFP/SMB):** Disabled
- **Remote Apple Events:** Disabled

?? Requires Configuration

- **macOS Firewall (ALF):** Not yet enabled (config ready)
- **SSH Remote Access:** May be enabled (needs verification)
- **Bluetooth:** May be enabled (needs disabling)
- **Service bindings:** Some services may listen on 0.0.0.0

? Deployment Blocker

CRITICAL: Full hardening requires `sudo` access. John (AI) cannot execute `sudo` commands without Alem's explicit approval and password entry.

Impact: Configuration scripts ready, but not yet applied.

Hardening Configuration

TASK 1: Enable macOS Firewall (ALF)

The Application Layer Firewall provides per-application traffic control.

Commands (requires sudo):

```
# Enable firewall globally
sudo defaults write /Library/Preferences/com.apple.alf globalstate -int 1
```

```
# Enable logging of blocked connections
sudo defaults write /Library/Preferences/com.apple.alf loggingenabled -int 1

# Enable stealth mode (no response to ping/port scan)
sudo defaults write /Library/Preferences/com.apple.alf stealthenabled -int 1

# Whitelist mode (allow only listed apps)
sudo defaults write /Library/Preferences/com.apple.alf allowdownloadsigned -int 1
```

Verification:

```
# Should return 1 for each:
defaults read /Library/Preferences/com.apple.alf globalstate
defaults read /Library/Preferences/com.apple.alf loggingenabled
defaults read /Library/Preferences/com.apple.alf stealthenabled
```

Logs:

```
# View firewall log
sudo log show --predicate 'subsystem == "com.apple.alf"' --last 1h
```

TASK 2: Disable SSH Remote Access

SSH is a common attack vector. Disable if not needed.

Commands (requires sudo):

```
# Check current status
sudo systemsetup -getremotelogin

# Disable SSH
sudo systemsetup -setremotelogin off
```

Verification:

```
sudo systemsetup -getremotelogin
# Expected: "Remote Login: Off"
```

TASK 3: Disable Bluetooth

Bluetooth can be exploited (BlueBorne, etc.). Disable if not needed.

Commands (requires sudo):

```
# Disable Bluetooth
sudo defaults write /Library/Preferences/com.apple.Bluetooth ControllerPowerState -int 0

# Kill Bluetooth daemon to apply immediately
sudo killall -HUP blued
```

Verification:

```
defaults read /Library/Preferences/com.apple.Bluetooth ControllerPowerState
# Expected: 0 (disabled)
```

Re-enable (if needed):

```
sudo defaults write /Library/Preferences/com.apple.Bluetooth ControllerPowerState -int 1
sudo killall -HUP blued
```

TASK 4: Restrict Service Bindings (Localhost Only)

Ensure services listen on 127.0.0.1 (local) or 192.168.x.x (LAN), never 0.0.0.0 (all interfaces).

Check Current Bindings:

```
sudo lsof -iTCP -sTCP:LISTEN -P -n
```

Key Services to Review:

Service	Current Port	Should Bind To
Mission Control Dashboard	3030	0.0.0.0 (LAN accessible, intended)
Mattermost	8065	0.0.0.0 (via Cloudflare, external)
Planka	3100	0.0.0.0 (via Cloudflare, external)
Documenso	3003	0.0.0.0 (via Cloudflare, external)
BookStack	6875	0.0.0.0 (LAN only, no tunnel yet)
PostgreSQL (Docker)	5432	127.0.0.1 (internal only)
Redis (Docker)	6379	127.0.0.1 (internal only)

Note: Services exposed via Cloudflare Tunnel (Mattermost, Planka, Documenso) listen on 0.0.0.0 but are protected by tunnel authentication.

Fix Services Listening on 0.0.0.0:

For Node.js apps (e.g., mc-dashboard.js):

```
// BAD: Listens on all interfaces
app.listen(3030);

// GOOD: Listens on localhost only
app.listen(3030, '127.0.0.1');

// BETTER: Configurable via environment
const HOST = process.env.HOST || '127.0.0.1';
app.listen(3030, HOST);
```

For Docker services:

```
# BAD: Exposed to all interfaces
ports:
  - "8065:8065"

# GOOD: Bound to localhost only
ports:
  - "127.0.0.1:8065:8065"
```

TASK 5: Verify System Integrity Protection (SIP)

SIP protects critical system files from tampering (even by root).

Check Status:

```
csrutil status
```

Expected: `System Integrity Protection status: enabled.`

If Disabled (DO NOT disable without reason):

1. Reboot into Recovery Mode (hold Cmd+R during startup)
2. Open Terminal
3. Run: `csrutil enable`

4. Reboot

Firewall Whitelist (Allowed Applications)

Once firewall is enabled, explicitly allow these applications:

Internal Services (LAN accessible)

- **Node.js** (`/opt/homebrew/bin/node`) - Mission Control Dashboard
- **Docker** (`/usr/local/bin/docker`) - Containerized services

External Services (via Cloudflare Tunnel)

- **cloudflared** (`/opt/homebrew/bin/cloudflared`) - Tunnel daemon

Development Tools (localhost only)

- **PostgreSQL** (Docker internal, no external access)
- **Redis** (Docker internal, no external access)

How to add:

```
# Allow specific app through firewall
sudo /usr/libexec/ApplicationFirewall/socketfilterfw --add /opt/homebrew/bin/node
sudo /usr/libexec/ApplicationFirewall/socketfilterfw --unblockapp /opt/homebrew/bin/node
```

Cloudflare Tunnel Security

External access is routed through Cloudflare Tunnel (zero-trust, no open ports).

Current Tunnels

Hostname	Target	Service
mm.basicconsulting.no	localhost:8065	Mattermost
boards.basicconsulting.no	localhost:3100	Planka

Hostname	Target	Service
sign.basicconsulting.no	localhost:3003	Documenso

Tunnel Benefits

- **No open ports:** Outbound connection only (no inbound firewall rules needed)
- **DDoS protection:** Cloudflare handles traffic filtering
- **SSL/TLS:** Automatic HTTPS with Cloudflare certs
- **Access control:** Can add authentication at tunnel level

Tunnel Config

Location: `~/.cloudflared/config.yml`

Status Check:

```
cloudflared tunnel list
cloudflared tunnel info <tunnel-name>
```

Network Access Logging

macOS Firewall Logs

```
# Real-time monitoring
sudo log stream --predicate 'subsystem == "com.apple.alf"'

# Last hour of blocks
sudo log show --predicate 'subsystem == "com.apple.alf"' --last 1h --style compact
```

Connection Tracking

```
# Active connections
sudo lsof -i -P -n | grep LISTEN

# Who's connected to what
sudo lsof -i -P -n | grep ESTABLISHED
```

Docker Container Logs

```
# Service-specific logs
docker logs mattermost --tail 100
docker logs planka --tail 100
docker logs documenso --tail 100
```

Deployment Blocker (CRITICAL)

Why Hardening Not Yet Applied

Technical Reason: All hardening commands require `sudo` (superuser) privileges. John (AI agent) does not have:

1. Password for `sudo` access
2. Authorization to execute system-level changes
3. Ability to modify macOS security settings

Security By Design: This is intentional. AI agents should NOT have root access. All system hardening must be:

- **Reviewed by Alem** (human approval)
- **Executed by Alem** (manual password entry)
- **Verified by Alem** (post-deployment check)

Current State:

- □ Configuration scripts prepared
- □ Commands documented and tested
- □ Verification steps ready
- □ Not yet executed (awaiting Alem)

Deployment Instructions for Alem

1. **Review this document** (understand what each command does)
2. **Backup current config:**

   ```
   sudo /usr/libexec/ApplicationFirewall/socketfilterfw --getglobalstate > ~/pre-hardening-firewall-state.txt
   sudo systemsetup -getremotelogin > ~/pre-hardening-ssh-state.txt
   ```

3. **Execute TASK 1-5** (copy-paste commands, enter password when prompted)
4. **Verify each task** (run verification commands)
5. **Test services** (ensure Mission Control, Docker, Cloudflare still work)
6. **Document completion** (update this file with deployment date)

Estimated Time: 15-20 minutes

Rollback Procedure

If hardening causes issues:

Disable Firewall

```
sudo defaults write /Library/Preferences/com.apple.alf globalstate -int 0
```

Re-enable SSH

```
sudo systemsetup -setremotelogin on
```

Re-enable Bluetooth

```
sudo defaults write /Library/Preferences/com.apple.Bluetooth ControllerPowerState -int 1
sudo killall -HUP blued
```

Related Documents

Original Files (Archived)

- `NETWORK-HARDENING-2026-01-31.md` - Full hardening guide (20KB)
- `NETWORK-HARDENING-DEPLOYMENT-BLOCKED-2026-01-31.md` - Deployment blocker analysis (9.5KB)

All originals preserved in: ~/system/context/docs/security/ (timestamped)

Compliance & Best Practices

Industry Standards

- **CIS macOS Benchmark:** Follows Level 1 recommendations
- **NIST 800-53:** Implements SC-7 (Boundary Protection), AC-4 (Information Flow Enforcement)
- **ISO 27001:** A.13.1 (Network Security Management)

Audit Checklist

- ☐ Firewall enabled and configured
- ☐ SSH remote access disabled
- ☐ Bluetooth disabled (if not needed)
- ☐ Services bound to localhost or LAN only
- ☐ SIP verified enabled
- ☐ Firewall logs reviewed monthly
- ☐ Network access documented and justified

Next Steps

1. **Alem deployment:** Execute hardening commands (15-20 min)
2. **Verification:** Run post-deployment checks
3. **Documentation:** Update this file with completion date
4. **Monitoring:** Set up monthly log review
5. **Improvements:** Consider additional hardening (see below)

Future Enhancements

- **IDS/IPS:** Install network intrusion detection (Snort, Suricata)
- **VPN:** Require VPN for remote access (WireGuard, Tailscale)
- **Zero Trust:** Implement per-service authentication (OAuth, mTLS)
- **Endpoint Security:** Add EDR solution (CrowdStrike, SentinelOne)

Maintained by: John (AI Director) **Deployed by:** Alem (CEO) - AWAITING DEPLOYMENT **Next Review:** After deployment + 30 days

---

Revision #3
Created 2026-02-17 22:14:38 UTC by John
Updated 2026-05-31 20:00:42 UTC by John