Secure Web Hosting with AlmaLinux 9

Hardened Apache, Nginx, Firewall, Fail2Ban, and SSL Setup Guide

Preface

Every day, thousands of web servers are compromised—not because attackers are brilliant, but because defenders left the door open. Default configurations, unpatched software, misconfigured firewalls, and neglected logs create an attack surface that automated bots exploit around the clock. If you've ever deployed a Linux server and wondered whether it was truly **secure**, this book was written for you.

Why This Book Exists

Secure Web Hosting with AlmaLinux 9 was born from a simple observation: most web hosting guides teach you how to get a server *running*, but very few teach you how to keep it *secure*. There is a vast and dangerous gap between a functioning web server and a hardened one, and that gap is precisely where attackers operate. This book exists to close it.

The focus here is singular and deliberate—**secure configuration at every layer of your hosting stack**. From the operating system kernel to TLS cipher suites, from file permissions to intrusion detection, every chapter is built around one central question: *How do we make this secure?*

What You Will Learn

This book takes a ground-up approach to secure web hosting on AlmaLinux 9, one of the most trusted enterprise Linux distributions available today. You will begin by

understanding **why web servers get hacked**—the real-world attack vectors, common misconfigurations, and the attacker mindset that makes unsecured servers such easy targets.

From there, you will build a **hardened AlmaLinux 9 base**, configure `firewalld` for disciplined network access control, and apply network-level hardening techniques that reduce your exposure before a single web service is installed. You will then deploy and secure both **Apache** and **Nginx** with production-grade configurations designed to resist common exploits.

The journey continues through securing the full application stack—**PHP-FPM**, **MySQL/MariaDB**, and the file system itself—because a secure web server means nothing if the layers behind it are vulnerable. You will implement **Fail2Ban** to automatically block brute-force attacks, establish **log monitoring and alerting** to detect suspicious activity in real time, and deploy **SSL/TLS certificates with Let's Encrypt** alongside advanced TLS hardening to protect data in transit.

Finally, you will learn how to maintain security over time through **continuous security maintenance** practices and how to evolve from secure hosting into a broader **production security architecture**.

Who This Book Is For

This book is for system administrators, DevOps engineers, web developers, and anyone responsible for deploying or maintaining Linux-based web servers. Whether you manage a single VPS or a fleet of production servers, the secure configurations and hardening strategies presented here will immediately strengthen your infrastructure. Prior experience with Linux command-line administration is assumed, but every secure configuration is explained with clarity and context.

How This Book Is Structured

The sixteen chapters follow a logical progression—**secure the foundation first, then secure each service layer, then maintain and evolve your security posture**. The five appendices provide ready-to-use templates and checklists, including hardened configuration files for Apache and Nginx, firewall rule examples, Fail2Ban jail templates, and a comprehensive VPS security hardening checklist. These are designed to be practical references you will return to long after your first read.

Acknowledgments

This book owes a debt of gratitude to the open-source community—the developers behind AlmaLinux, Apache, Nginx, Fail2Ban, Let's Encrypt, and the countless security tools that make secure hosting possible. I also want to thank the security researchers and practitioners whose published work, vulnerability disclosures, and shared knowledge continue to raise the bar for all of us. Finally, to every reader who chooses to take security seriously rather than leaving it to chance: the internet is better because of you.

Security is not a product you install—it is a discipline you practice. Let this book be your guide to building, configuring, and maintaining web servers that are secure by design and resilient by habit.

Bas van den Berg

Table of Contents

Chapter 1: Why Web Servers Get Hacked

Every day, thousands of web servers across the globe fall victim to attacks that could have been prevented. The reality of modern web hosting is both sobering and instructive: the majority of successful breaches do not result from sophisticated zero-day exploits or genius-level hackers breaking through military-grade encryption. Instead, they stem from mundane oversights, default configurations left unchanged, unpatched software, weak passwords, and a general lack of security awareness among system administrators. Understanding why web servers get hacked is the essential first step toward building a hosting environment that resists attack, and that understanding forms the foundation upon which every subsequent chapter of this book is built.

When we talk about securing a web server running AlmaLinux 9, we are not merely discussing the installation of a firewall or the generation of an SSL certificate. We are talking about adopting a mindset, a disciplined approach to every decision made during server setup, configuration, and ongoing maintenance. Before we can harden Apache, configure Nginx securely, deploy Fail2Ban, or implement SSL with confidence, we must first appreciate the threat landscape. We must understand the adversary, their motivations, their methods, and the weaknesses they exploit. This chapter provides that critical context.

The Modern Threat Landscape

The internet has evolved dramatically since the early days of static HTML pages served from university basements. Today, web servers host complex applications, process financial transactions, store sensitive personal data, and serve as the backbone of businesses large and small. This evolution has made web servers extraordinarily valuable targets. Attackers are no longer motivated solely by curiosity or the desire for notoriety. The modern threat landscape is driven by financial gain, political activism, espionage, and even warfare.

Consider the scale of the problem. According to multiple industry reports, a new cyberattack occurs approximately every 39 seconds. Web applications are the single most targeted attack vector, accounting for a significant percentage of all confirmed data breaches. The average cost of a data breach has climbed into the millions of dollars, and for small businesses, a single successful attack can mean permanent closure.

AlmaLinux 9, as a community-driven enterprise Linux distribution that emerged as a replacement for CentOS, inherits the robust security architecture of Red Hat Enterprise Linux. However, no operating system is secure by default in the context of web hosting. The security of a web server depends entirely on how it is configured, maintained, and monitored. A freshly installed AlmaLinux 9 server connected to the internet without hardening is an open invitation to attackers, and they will find it faster than most administrators realize.

Automated scanning tools continuously sweep entire ranges of IP addresses, probing for open ports, identifying running services, and testing for known vulnerabilities. Within minutes of a new server coming online, it will begin receiving connection attempts from bots and automated attack frameworks. This is not a theoretical concern; it is an observable, measurable reality that any administrator can verify by examining the authentication logs of a newly deployed server.

Common Reasons Web Servers Are Compromised

Understanding the specific reasons web servers fall to attackers requires examining the most frequently exploited weaknesses. The following table provides a comprehensive overview of the primary attack vectors and their relationship to server security.

Attack Vector	Description	Impact Level	Prevention Approach
Default Configurations	Services installed with factory settings, including default ports, enabled modules, and sample pages	High	Hardening configurations immediately after installation, removing unnecessary modules and default content
Unpatched Software	Operating system, web server, or application software running with known vulnerabilities	Critical	Establishing a regular patch management schedule using dnf update on AlmaLinux 9
Weak Authentication	Simple passwords, shared credentials, or lack of multi-factor authentication for administrative access	Critical	Enforcing strong password policies, implementing key-based SSH authentication, disabling root login
Misconfigured Firewalls	Firewall rules that are too permissive, allowing unnecessary traffic to reach the server	High	Configuring firewalld on AlmaLinux 9 to allow only required ports and services

Missing SSL/TLS	Serving content over unencrypted HTTP, exposing data in transit to interception	High	Implementing SSL/TLS certificates using Let's Encrypt or commercial certificate authorities
Directory Traversal	Web server configuration allowing access to files outside the intended document root	Medium	Proper Apache and Nginx configuration with restrictive directory permissions
Brute Force Attacks	Automated attempts to guess passwords for SSH, web application logins, or database access	High	Deploying Fail2Ban to detect and block repeated failed authentication attempts
Information Disclosure	Server headers, error messages, or directory listings revealing software versions and server architecture	Medium	Configuring Apache and Nginx to suppress version information and disable directory listing
SQL Injection	Malicious SQL statements inserted through web application input fields	Critical	Input validation at the application level, web application firewalls, least-privilege database users
Cross-Site Scripting	Injection of malicious scripts into web pages viewed by other users	High	Proper output encoding, Content Security Policy headers, web application firewall rules

Each of these vectors represents a doorway that attackers actively probe. The critical insight is that most of these vulnerabilities are not inherent flaws in the technology itself. They are configuration failures and maintenance oversights. A properly configured and maintained AlmaLinux 9 server running hardened Apache or Ng-

inx, protected by firewalld and Fail2Ban, and serving content over properly configured SSL, eliminates the vast majority of these attack surfaces.

Let us examine several of these vectors in greater detail to understand exactly how they are exploited in practice.

The Danger of Default Configurations

When Apache HTTP Server or Nginx is installed on AlmaLinux 9, the default configuration is designed for functionality, not security. Apache, for example, ships with its default welcome page enabled, server signature and version information visible in HTTP response headers, and a set of loaded modules that far exceeds what most deployments require. Each loaded module represents additional code that could contain vulnerabilities, and each piece of information disclosed to visitors gives attackers valuable reconnaissance data.

Consider what happens when an attacker sends a simple HTTP request to a server running default Apache on AlmaLinux 9. The response headers might reveal something like this:

```
Server: Apache/2.4.57 (AlmaLinux)
X-Powered-By: PHP/8.1.14
```

This single response tells the attacker exactly which web server software is running, its precise version number, the operating system, and the PHP version. With this information, the attacker can immediately search vulnerability databases for known exploits targeting those specific versions. The attacker has gone from knowing nothing about the server to having a targeted attack plan in seconds, all because the default configuration freely volunteered that information.

The same principle applies to Nginx. A default Nginx installation on AlmaLinux 9 will include the server version in response headers and may have configurations

that are more permissive than necessary. The process of hardening these services, which we will cover in detail in later chapters, begins with understanding that defaults are starting points, never endpoints.

The Unpatched Software Epidemic

Perhaps no single factor contributes more to successful web server compromises than unpatched software. When a vulnerability is discovered in Apache, Nginx, OpenSSL, PHP, or any other component of the web hosting stack, a race begins. Security researchers and vendors work to develop and release patches. Simultaneously, attackers work to develop exploits and identify vulnerable servers. The window between public disclosure of a vulnerability and the availability of a patch is dangerous, but the window between patch availability and actual patch application by administrators is where the real damage occurs.

On AlmaLinux 9, the package management system provides a straightforward mechanism for keeping software current:

```
sudo dnf check-update
```

This command checks for available updates across all installed packages. To apply all available security updates, an administrator would execute:

```
sudo dnf update --security
```

To apply all available updates, including bug fixes and enhancements:

```
sudo dnf update -y
```

The simplicity of these commands makes the prevalence of unpatched servers all the more frustrating. Administrators who fail to establish regular patching routines leave their servers vulnerable to attacks that exploit well-documented, publicly known weaknesses. Automated tools like Metasploit contain modules for exploit-

ing hundreds of known vulnerabilities, and script kiddies with minimal technical knowledge can use these tools to compromise unpatched servers.

Note: AlmaLinux 9 supports automatic security updates through the dnf-automatic package. Configuring this service is one of the first steps in establishing a secure baseline, and we will walk through its configuration in a later chapter. However, automatic updates must be implemented thoughtfully, as updates can occasionally introduce compatibility issues with running applications.

Brute Force and the Importance of Authentication Hardening

SSH is the primary method by which administrators manage their AlmaLinux 9 servers remotely. It is also one of the most targeted services by automated attack tools. A server with SSH exposed on the default port 22 with password authentication enabled will receive thousands of login attempts per day. These brute force attacks use dictionaries of common usernames and passwords, cycling through combinations at high speed.

To illustrate the scale of this problem, consider examining the authentication log on a newly deployed server:

```
sudo journalctl -u sshd | grep "Failed password" | wc -l
```

On a server that has been online for even a few hours, this command frequently returns hundreds or thousands of results. Each line represents a failed login attempt, an attacker or automated bot trying to guess credentials.

The defense against brute force attacks is multi-layered. First, password authentication for SSH should be disabled entirely in favor of key-based authentication. On AlmaLinux 9, this is configured in the SSH daemon configuration file:

```
sudo vi /etc/ssh/sshd_config
```

The relevant settings include:

```
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
MaxAuthTries 3
```

After making these changes, the SSH service must be restarted:

```
sudo systemctl restart sshd
```

Second, Fail2Ban should be deployed to monitor authentication logs and automatically block IP addresses that exhibit brute force behavior. Fail2Ban watches log files for patterns indicating repeated failed login attempts and creates temporary firewall rules to block the offending IP addresses. This dramatically reduces the noise in authentication logs and provides an active defense layer.

Third, the firewall itself should be configured to limit SSH access. If administrative access is only needed from specific IP addresses or ranges, firewalld can be configured to restrict SSH accordingly:

```
sudo firewall-cmd --permanent --add-rich-rule='rule family="ipv4"
source address="203.0.113.50" service name="ssh" accept'
sudo firewall-cmd --permanent --remove-service=ssh
sudo firewall-cmd --reload
```

This combination of key-based authentication, Fail2Ban monitoring, and firewall restrictions transforms SSH from a vulnerable attack surface into a well-defended administrative channel.

Who Are the Attackers

Understanding the motivations and capabilities of attackers helps administrators prioritize their defensive efforts. The following table categorizes the primary types of threat actors that target web servers.

Threat Actor	Motivation	Capability Level	Typical Targets	Common Methods
Script Kiddies	Curiosity, bragging rights, minor vandalism	Low	Any vulnerable server, targets of opportunity	Automated scanning tools, publicly available exploits, default credential lists
Cybercriminals	Financial gain through data theft, ransomware, cryptomining	Medium to High	E-commerce sites, databases with personal information, any server with processing power	Sophisticated phishing, custom malware, exploitation of unpatched vulnerabilities
Hacktivists	Political or social messaging	Low to Medium	Government sites, corporate targets aligned with their cause	Website defacement, denial of service, data leaks
Nation-State Actors	Espionage, sabotage, strategic advantage	Very High	Critical infrastructure, government systems, defense contractors	Advanced persistent threats, zero-day exploits, supply chain attacks

Insiders	Revenge, financial gain, negligence	Varies	Their own organization's systems	Abuse of legitimate access, data exfiltration, intentional misconfiguration
Automated Bots	Programmatic exploitation at scale	Low individually, massive in aggregate	Every server connected to the internet	Continuous scanning, credential stuffing, vulnerability probing

The most important observation from this table is that automated bots represent the most common and persistent threat to web servers. These bots do not discriminate. They do not care whether a server hosts a Fortune 500 company's website or a personal blog. They scan, they probe, and when they find a weakness, they exploit it. This is why the argument that "my server is too small or unimportant to be targeted" is dangerously wrong. Every server is a target simply by virtue of being connected to the internet.

The Real Cost of a Compromise

When a web server is compromised, the consequences extend far beyond the immediate technical problem. The following represents a realistic sequence of events following a typical server breach:

First, the attacker gains initial access, often through a brute-forced SSH password or an unpatched vulnerability in a web application. They establish persistence by creating additional user accounts, installing backdoors, or modifying system services to maintain access even if the original vulnerability is patched.

Next, the attacker surveys the compromised system. They examine databases for valuable data, check for stored credentials that might provide access to other systems, and evaluate the server's resources. A server with significant processing power might be enrolled in a cryptocurrency mining botnet. A server with access to customer data might have that data exfiltrated for sale on dark web marketplaces.

The compromised server may also be used as a platform for attacking other targets. It might be enrolled in a distributed denial-of-service botnet, used to send spam or phishing emails, or used as a proxy to obscure the attacker's true location during attacks against other systems.

For the server owner, the consequences include data loss, reputational damage, potential legal liability under data protection regulations such as GDPR, the cost of incident response and forensic investigation, and the operational disruption of taking systems offline for remediation. For businesses, customer trust, once lost, is extraordinarily difficult to rebuild.

Note: Under regulations like GDPR, organizations that fail to adequately protect personal data can face fines of up to 4 percent of annual global revenue or 20 million euros, whichever is greater. The argument that security is too expensive or too time-consuming collapses entirely when weighed against these potential penalties.

Building a Security Mindset

The purpose of this chapter is not to create fear but to establish the foundation for informed, deliberate security practices. Every chapter that follows in this book addresses specific, actionable measures for securing a web hosting environment on

AlmaLinux 9. But those measures are only effective when implemented by administrators who understand why they matter.

Security is not a product that can be purchased and installed. It is not a single configuration change or a one-time audit. Security is a continuous process of assessment, implementation, monitoring, and adaptation. The threat landscape evolves constantly, and defensive measures must evolve with it.

As we move forward through this book, we will systematically address each layer of the security stack. We will harden the AlmaLinux 9 operating system itself, configure Apache and Nginx with security as the primary design criterion, deploy and configure firewalld to control network access, implement Fail2Ban to provide active defense against brute force attacks, and establish SSL/TLS encryption to protect data in transit.

Each of these measures addresses specific attack vectors described in this chapter. Each reduces the attack surface available to adversaries. Together, they create a defense-in-depth architecture that makes successful compromise dramatically more difficult and provides the monitoring and alerting capabilities needed to detect and respond to attacks that do occur.

The journey toward a secure web hosting environment begins with understanding the threat. You now have that understanding. In the next chapter, we will begin the practical work of building that environment, starting with the installation and initial hardening of AlmaLinux 9 itself.

Practical Exercise: Assessing Your Current Exposure

Before moving to the next chapter, perform the following exercise on a test server or virtual machine running AlmaLinux 9. This exercise is designed to demonstrate the reality of the threats discussed in this chapter.

Step 1: Install AlmaLinux 9 on a virtual machine with a minimal installation profile. Connect it to a network with internet access.

Step 2: After installation, check for available security updates:

```
sudo dnf check-update --security
```

Document the number of security updates available. Even a freshly installed system may have pending security patches if the installation media is not the most current version.

Step 3: Examine the default SSH configuration:

```
sudo grep -E "PermitRootLogin|PasswordAuthentication|
PubkeyAuthentication" /etc/ssh/sshd_config
```

Note the default values. Consider what each setting means in terms of security exposure.

Step 4: Check which ports are currently open and which services are listening:

```
sudo ss -tulnp
```

Document every listening service. For each service, ask yourself: Is this service necessary for the server's intended function? If not, it represents unnecessary attack surface.

Step 5: Examine the current firewall configuration:

```
sudo firewall-cmd --list-all
```

Note which services and ports are permitted through the firewall by default. Consider whether each permitted service is actually required.

Step 6: If the server has been connected to the internet for any period of time, check for failed SSH login attempts:

```
sudo journalctl -u sshd --since "1 hour ago" | grep -c "Failed"
```

The results of this exercise will provide concrete, personal evidence of the threats discussed throughout this chapter. They will also serve as a baseline against which you can measure the security improvements made as you work through the remaining chapters of this book.

Exercise Step	Command	What It Reveals	Security Implication
Check security updates	dnf check-update --security	Number of known vulnerabilities in installed packages	Unpatched systems are vulnerable to known exploits
Review SSH configuration	grep relevant settings in sshd_config	Default authentication and access settings	Default SSH settings often permit password-based and root authentication
List listening services	ss -tulnp	All network services accepting connections	Each listening service is a potential attack vector
Review firewall rules	firewall-cmd --list-all	Current network access control rules	Overly permissive rules expose services unnecessarily
Count failed SSH logins	journalctl grep for Failed entries	Volume of brute force attempts against SSH	Demonstrates the constant automated attack pressure on internet-facing servers

This exercise transforms the abstract concepts discussed in this chapter into tangible, observable data on your own system. It is the first step in developing the hands-on security skills that will be built upon throughout the remainder of this book.

Chapter 2: Building a Hardened AlmaLinux 9 Base

When constructing a secure web hosting environment, the foundation upon which everything rests is the operating system itself. No amount of application-level security, firewall rules, or intrusion detection systems can compensate for a poorly configured base operating system. Think of it this way: you would never build a fortress on sand. The same principle applies to your server infrastructure. AlmaLinux 9, as an enterprise-grade Linux distribution, provides an excellent starting point, but out of the box, it is configured for general-purpose use rather than hardened web hosting. This chapter walks you through every critical step of transforming a fresh AlmaLinux 9 installation into a security-hardened platform ready to host web services with confidence.

We will begin with the installation process itself, making deliberate choices about disk partitioning, package selection, and initial configuration. From there, we will move into post-installation hardening, covering everything from kernel parameters and filesystem permissions to user account policies and audit logging. By the end of this chapter, you will have a base system that follows industry best practices for security, aligned with benchmarks published by the Center for Internet Security and recommendations from Red Hat's own security guides.

Choosing the Right Installation Profile

The very first security decision you make happens before AlmaLinux 9 is even fully installed. During the installation process, the Anaconda installer presents you with several options that have profound implications for the security posture of your server. The most important of these is the software selection screen, often called the "Base Environment" selection.

For a secure web hosting server, you should always select the "Minimal Install" option. This is not merely a suggestion; it is a fundamental security principle known as "attack surface reduction." Every package installed on your system represents potential vulnerabilities, additional network services that might listen on ports, and more code that must be kept updated. A minimal installation includes only the core operating system components required to boot and operate the system.

```
# After installation, verify the minimal install by checking
installed packages
dnf list installed | wc -l
```

On a truly minimal installation, you should see roughly 300 to 400 packages. Compare this to a "Server with GUI" installation, which can include over 1,200 packages. Each additional package is a potential entry point for an attacker.

The following table outlines the recommended installation choices and their security implications:

Installation Option	Recommended Setting	Security Rationale
Base Environment	Minimal Install	Reduces attack surface by eliminating unnecessary packages and services
Software Selection Add-ons	None selected	Prevents installation of development tools, GUI components, and unnecessary daemons

Root Password	Strong, unique password	First line of defense for privileged access; use at least 16 characters with mixed complexity
User Creation	Create a non-root administrative user	Enables principle of least privilege; root login will be disabled later
Network Configuration	Configure static IP if possible	Predictable network configuration aids in firewall rule creation
Security Policy	CIS AlmaLinux 9 Benchmark (if available)	Applies automated hardening during installation
Kdump	Disabled	Reduces memory footprint and eliminates a service that is unnecessary for production web hosting

Note: If you are installing on a cloud provider such as AWS, DigitalOcean, or Linode, you may not have access to the Anaconda installer directly. In that case, start with the provider's AlmaLinux 9 minimal image and proceed with post-installation hardening as described in the following sections.

Disk Partitioning for Security

Disk partitioning is often treated as a mundane administrative task, but from a security perspective, it is one of the most consequential decisions you will make. Proper partition layout allows you to apply mount options that restrict what can happen on each filesystem, which directly mitigates several classes of attacks.

The principle here is separation of concerns. By placing different types of data on separate partitions, you can apply restrictive mount options that would be impractical on a single root partition. For example, you can prevent executable files

from running in `/tmp`, which is a common location where attackers drop and execute malicious payloads.

Here is the recommended partition layout for a secure web hosting server:

Mount Point	Suggested Size	Filesystem	Mount Options	Purpose
/boot	1 GB	xfs	defaults,nosuid,nodev	Boot loader and kernel images
/boot/efi	512 MB	vfat	defaults,nosuid,nodev	EFI system partition (UEFI systems only)
/	20 GB	xfs	defaults	Root filesystem; kept small to limit exposure
/home	10 GB	xfs	defaults,nosuid,nodev	User home directories
/tmp	5 GB	xfs	defaults,nosuid,nodev,noexec	Temporary files; noexec prevents script execution
/var	20 GB	xfs	defaults,nosuid,nodev	Variable data including logs and mail spools
/var/log	10 GB	xfs	defaults,nosuid,nodev,noexec	System logs; separate partition prevents log flooding from filling root
/var/log/audit	5 GB	xfs	defaults,nosuid,nodev,noexec	Audit logs; critical for forensic analysis
/var/tmp	5 GB	xfs	defaults,nosuid,nodev,noexec	Persistent temporary files

/var/www	Remaining space	xfs	defaults,nosuid,nodev	Web content; sized according to your hosting needs
swap	2x RAM (up to 8 GB)	swap	defaults	Virtual memory

After installation, you can verify and modify mount options by editing the `/etc/fstab` file:

```
# View current mount options
mount | column -t

# Edit fstab to add security mount options
vi /etc/fstab
```

An example `/etc/fstab` entry with hardened mount options for `/tmp`:

```
/dev/mapper/almalinux-tmp  /tmp  xfs
defaults,nosuid,nodev,noexec  0  0
```

After modifying `/etc/fstab`, remount the affected partitions without rebooting:

```
# Remount /tmp with new options
mount -o remount /tmp

# Verify the new mount options are active
mount | grep /tmp
```

Note: The `nosuid` option prevents set-user-identifier and set-group-identifier bits from taking effect, which stops privilege escalation through SUID binaries placed in that location. The `nodev` option prevents the creation of device files, and `noexec` prevents the execution of any binary on that partition. Together, these three options form a powerful defense against common attack techniques.