Vulhub jako nástroj pro výuku zranitelností

Bc. Nikita Zaykov

Diplomová práce | 2026

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav elektroniky a měření

Akademický rok: 2025/2026

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Nikita Zaykov**
Osobní číslo:	**A24371**
Studijní program:	**N1032A020003 Bezpečnostní technologie, systémy a management**
Specializace:	**Bezpečnostní technologie**
Forma studia:	**Prezenční**
Téma práce:	**Vulhub jako nástroj pro výuku zranitelností**
Téma práce anglicky:	**Vulhub as a Tool for Learning about Vulnerabilities**

Zásady pro vypracování

1. Popište současný stav kybernetické bezpečnosti webových aplikací se zaměřením na zranitelnosti dle projektu OWASP Top 10.
2. Analyzujte možnosti zneužití těchto zranitelností a jejich dopady na informační systémy.
3. Navrhněte a vytvořte laboratorní prostředí pro demonstraci vybraných zranitelností s využitím Kali Linux, Docker a Vulhub.
4. Navrhněte a otestujte sadu laboratorních úloh vhodných pro výuku předmětu Počítačové viry a bezpečnost.
5. Vytvořte podrobné návody pro studenty i vyučující k realizaci laboratorních úloh.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. KOLOUCH, Jan. *CyberCrime*. Praha: CZ.NIC, z.s.p.o., 2016. ISBN 978-80-88168-18-8.
2. KOLOUCH, Jan; KROPÁČOVÁ, Andrea; KUNC, Martin a BAŠTA, Pavel. *CyberSecurity*. Praha: CZ.NIC, z.s.p.o., 2019. ISBN 978-80-88168-34-8.
3. STUTTARD, Dafydd a PINTO, Marcus. *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws.* Indianapolis: Wiley Publishing, 2008. ISBN 978-0-470-17077-9.
4. WEIDMAN, Georgia. *Penetration testing: a hands-on introduction to hacking.* San Francisco: No Starch Press, 2014. ISBN 978-1-59327-564-8.
5. KASPERSKI, Kris. *Komputernye virusy iznutri i snaruzhi.* Piter, 2013. ISBN 5-469-00982-3.

Vedoucí diplomové práce: **Ing. Lukáš Králík, Ph.D.**
Ústav bezpečnostního inženýrství

Datum zadání diplomové práce: **26. listopadu 2025**
Termín odevzdání diplomové práce: **18. května 2026**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.
DĚKAN

doc. Ing. Milan Navrátil, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 26. listopadu 2025

Prohlášení autora závěrečné kvalifikační práce

Beru na vědomí, že

- odevzdáním závěrečné práce souhlasím se zpřístupněním své práce podle zákona č. 111/1998 Sb., v platném znění bez ohledu na výsledek obhajoby;
- závěrečná práce bude uložena v elektronické podobě v univerzitním informačním systému;
- jedno vyhotovení závěrečné práce v listinné podobě bude ponecháno Univerzitě Tomáše Bati ve Zlíně k uložení;
- na moji závěrečnou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- podle § 60 odst. 2 a 3 mohu užít své dílo – závěrečnou práci – nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- pokud bylo k vypracování závěrečné práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tj. k nekomerčnímu využití), nelze výsledky závěrečné práce využít ke komerčním účelům;
- pokud je výstupem závěrečné práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá; neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji, že

- jsem na závěrečné práci pracoval(a) samostatně a použitou literaturu jsem řádně citoval(a); v případě publikace výsledků budu uveden(a) jako spoluautor;
- odevzdaná verze závěrečné práce a verze elektronická nahraná do IS/STAG jsou obsahově totožné.
- Prohlašuji, že při tvorbě této práce jsem použil nástroj generativního modelu AI ChatGPT; https://chatgpt.com za účelem jazykové korektury a stylistické úpravy textu. Po použití tohoto nástroje jsem provedl kontrolu obsahu a přebírám za něj plnou zodpovědnost.

Ve Zlíně, dne ..... Podpis autora .....

Abstrakt

Diplomová práce se zabývá návrhem a vytvořením laboratorních úloh pro výuku penetračního testování webových aplikací se zaměřením na zranitelnosti dle OWASP Top 10:2025. Práce shrnuje teoretická východiska problematiky webové bezpečnosti, popisuje vybrané kategorie zranitelností a analyzuje jejich možné dopady na informační systémy. Součástí práce je návrh a implementace laboratorního prostředí založeného na technologiích Kali Linux, Docker a Vulhub. Na tomto základě byla vytvořena sada deseti laboratorních úloh, z nichž osm vychází z připravených scénářů projektu Vulhub a dvě byly vytvořeny autorsky pro pokrytí kategorií Cryptographic Failures a Security Logging and Alerting Failures. Výsledkem práce je ucelený soubor výukových materiálů využitelný při výuce předmětu Počítačové viry a bezpečnost.

Klíčová slova
penetrační testování, webové aplikace, OWASP Top 10, laboratorní úlohy, kybernetická bezpečnost, red teaming

Abstract

This diploma thesis focuses on the design and development of laboratory exercises for teaching penetration testing of web applications with an emphasis on vulnerabilities according to OWASP Top 10:2025. The thesis summarizes the theoretical background of web application security, describes selected vulnerability categories, and analyzes their potential impact on information systems. An important part of the thesis is the design and implementation of a laboratory environment based on Kali Linux, Docker, and Vulhub. Based on this environment, a set of ten laboratory exercises was created; eight of them are based on scenarios from the Vulhub project, while two were developed by the author to cover the categories Cryptographic Failures and Security Logging and Alerting Failures. The result of the thesis is a coherent set of teaching materials applicable in the course Computer Viruses and Security.

Keywords
penetration testing, web applications, OWASP Top 10, laboratory exercises, cybersecurity, red teaming

Rád bych poděkoval svému vedoucímu diplomové práce, Ing. Lukáši Králíkovi, Ph.D., za odborné vedení, cenné rady, připomínky a vstřícný přístup při zpracování této práce. Jeho podpora a doporučení mi pomohly při návrhu, realizaci i dokončení celé diplomové práce.

Obsah

Seznam tabulek

Seznam použitých symbolů a zkratek

OWASP	Open Worldwide Application Security Project
SaaS	Software as a Service
API	Application Programming Interface
ERP	Enterprise Resource Planning
CRM	Customer Relationship Management
DBIR	Data Breach Investigations Report
GDPR	General Data Protection Regulation
DevOps	Development and Operations
CI/CD	Continuous Integration / Continuous Delivery
ASVS	Application Security Verification Standard
A01–A10	označení kategorií zranitelností dle OWASP Top 10
SQL	Structured Query Language
HTTP	Hypertext Transfer Protocol
URL	Uniform Resource Locator
PHP	Hypertext Preprocessor
XML	Extensible Markup Language
XXE	XML External Entity
NIST	National Institute of Standards and Technology
PTES	Penetration Testing Execution Standard
OSSTMM	Open-Source Security Testing Methodology Manual
WAF	Web Application Firewall
SIEM	Security Information and Event Management
CMS	Content Management System
DBMS	Database Management System
CVE	Common Vulnerabilities and Exposures
RCE	Remote Code Execution

XSS	Cross-Site Scripting
AES	Advanced Encryption Standard
ECB	Electronic Codebook
MD5	Message-Digest Algorithm 5
SHA1	Secure Hash Algorithm 1

Seznam příloh

Přílohy vložené v závěrečné práci

Příloha A: Seznam přiložených souborů

Ostatní elektronické přílohy:

Příloha A: Zaykov_DP_Přílohy .zip

Úvod

Ochrana webových aplikací patří mezi klíčové oblasti moderní kyberbezpečnosti. Webové technologie jsou dnes běžně využívány v podnikání, školství i státní správě a webové aplikace často představují přístupový bod k databázím, interním systémům a uživatelským účtům. Jejich zranitelnost proto může vést k úniku dat, narušení provozu služeb i k významným finančním a reputačním ztrátám.

Významným zdrojem poznatků v této oblasti je projekt OWASP Top 10, který shrnuje nejzávažnější kategorie zranitelností webových aplikací. Tento přehled slouží nejen jako bezpečnostní standard, ale i jako vhodný základ pro výuku a analýzu rizik. Pro efektivní výuku však nestačí pouze teoretický popis zranitelností. Je nutné doplnit jej praktickou demonstrací, protože teprve při práci s reálnými scénáři si studenti dokážou lépe představit, jak zranitelnosti vznikají, jak mohou být zneužity a jaké mohou mít důsledky pro informační systémy.

Cílem této diplomové práce je návrh a vytvoření laboratorních úloh zaměřených na výuku penetračního testování webových aplikací se zaměřením na kategorie OWASP Top 10:2025. Hlavním úkolem je připravit soubor praktických cvičení pro předmět Počítačové viry a bezpečnost, který studentům umožní pochopit principy vzniku zranitelností, způsoby jejich zneužití a jejich dopady na bezpečnost systémů. Součástí práce je také návrh laboratorního prostředí založeného na technologiích Kali Linux, Docker a Vulhub a zpracování návodů pro studenty i vyučující.

Práce je rozdělena do několika vzájemně navazujících částí. Nejprve jsou zpracována teoretická východiska zaměřená na bezpečnost webových aplikací a vybrané kategorie OWASP Top 10. Následně je popsán návrh a implementace laboratorního prostředí a poté samotný návrh, realizace a testování laboratorních úloh. Důraz je kladen nejen na technickou stránku, ale i na didaktickou použitelnost výsledných scénářů. Významnou součástí práce je také autorský přínos, protože pro některé kategorie bylo nutné vytvořit vlastní zranitelné aplikace.

Hlavním přínosem této práce je vytvoření prakticky využitelného souboru laboratorních úloh, který propojuje teorii s reálnou demonstrací útoků v bezpečném prostředí. Výsledkem by měl být materiál, který studentům usnadní pochopení problematiky webové bezpečnosti a vyučujícím poskytne připravený základ pro laboratorní výuku.

1 Současný stav kybernetické bezpečnosti webových aplikací

V dnešním světě jsou webové aplikace ústředním prvkem informačních systémů a slouží jako nástroj pro řadu podnikových funkcí – od řízení vztahů se zákazníky a online prodeje až po přístup k interním systémům a komunikaci s partnery. Vývoj digitálních technologií, včetně cloudových služeb, SaaS, architektury API-first a mikroslužeb, vedl k výraznému nárůstu počtu i technické složitosti dostupných webových aplikací. Bohužel toto rozšíření funkcionality a dostupnosti s sebou nese i větší potenciální útočnou plochu pro kyberzločince. [1]

Webové aplikace jsou jedním z nejzranitelnějších míst pro kybernetické útoky. Jejich dostupnost na internetu, zpracování citlivých dat a integrace s klíčovými firemními systémy (například databázemi, ERP, CRM nebo cloudovou infrastrukturou) z nich činí kriticky důležité prvky. Narušení jedné takové aplikace může mít dalekosáhlé následky, které se dotknou nejen samotného serveru, ale i značné části celého informačního systému. Webové aplikace jsou pro útočníky obzvláště lákavé, protože často umožňují přímou manipulaci s mechanismy autentizace, uživatelskými účty a datovými úložišti. [1][2]

Analýza současné situace v oblasti kybernetických hrozeb ukazuje, že útočníci stále častěji dosahují úspěchu kombinací známých zranitelností s automatizovanými nástroji a obecně nízkou úrovní řízení bezpečnosti. Výroční zpráva Verizon 2025 DBIR zdůrazňuje tuto komplexnost: využití zranitelností jako hlavního vektoru útoku vzrostlo o 34 % a podíl třetích stran na narušeních bezpečnosti dosáhl 30 %. Tato čísla jasně ukazují, že úspěšné útoky jsou málokdy výsledkem jediné technické chyby; častěji jsou způsobeny souhrnem faktorů, jako je neefektivní řízení dodavatelského řetězce softwaru, opožděné aktualizace nebo slabá kontrola přístupu.[3]

Pro hodnocení bezpečnosti webových aplikací je velmi důležitý projekt OWASP (Open Worldwide Application Security Project). Jedná se o globální otevřenou komunitu, jejímž posláním je zvyšovat úroveň bezpečnosti softwaru. Jeho nejznámějším výsledkem je OWASP Top 10, dokument shrnující nejkritičtější hrozby pro webové aplikace. Nejedná se pouze o seznam častých chyb, ale také o cenného praktického průvodce pro vývojáře, bezpečnostní specialisty, auditory a organizace, které usilují o systematické řízení rizik. OWASP Top 10 je široce používán v praxi pro testování bezpečnosti, tvorbu standardů a školení specialistů.[2]

OWASP Top 10:2025, nejnovější verze, byla podstatně přepracována, aby odpovídala aktuálnímu stavu hrozeb. OWASP oznamuje zavedení nových kategorií a změnu předchozí struktury, které reagují na aktuální výzvy. To znamená, že dokument nyní zahrnuje nejen obvyklé problémy, jako jsou chyby řízení přístupu nebo injekce, ale také rostoucí význam rizik spojených s dodavatelským řetězcem softwaru a zajištěním integrity dat a komponent. Tato skutečnost jasně ukazuje, že bezpečnost webových aplikací již nelze považovat za

izolovaný problém chyb programátorů; jedná se o komplexní otázku, která vyžaduje pozornost věnovanou architektuře, provozu, aktualizacím, závislostem a řízení bezpečnosti ve všech fázích životního cyklu aplikace.[2]

Omezit se pouze na výčet kategorií zranitelností, například z OWASP Top 10, nestačí k adekvátnímu posouzení rizik. Je nutné provést hlubší analýzu, která zahrnuje mechanismy vzniku těchto zranitelností, potenciální vektory útoků a jejich vliv na důvěrnost, integritu a dostupnost informačních systémů. Právě takový komplexní přístup bude realizován v praktické části práce, kde budou vypracovány laboratorní úkoly napodobující reálné scénáře penetračního testování webových aplikací.

1.1 Digitalizace společnosti a rostoucí význam webových aplikací

V posledních desetiletích jsme svědky rychlé digitalizace, která mění všechny oblasti života: od podnikání a státní správy až po osobní záležitosti. Digitalizace je proces převodu analogových procesů a služeb do digitální podoby, který otevírá možnosti pro efektivnější správu, automatizaci a vzdálený přístup. Proto moderní společnosti stále aktivněji využívají informační systémy a webové aplikace pro správu svých operací, komunikaci se zákazníky, práci s daty a poskytování služeb online.[4]

Webové aplikace jsou dnes klíčovým prvkem digitální infrastruktury. Používají se pro širokou škálu úkolů, jako je elektronický obchod, online bankovnictví, správa zákaznických účtů, rezervační systémy, firemní portály a přístup k interním systémům. Díky tomu, že jsou dostupné přes internetový prohlížeč, mohou je uživatelé používat z různých zařízení a z jakéhokoli místa, aniž by museli instalovat speciální software. Právě tato snadná dostupnost se stala silným hnacím motorem jejich rychlého rozšíření v korporátní a státní sféře.[5]

Moderní webové aplikace nabývají na významu díky rozvoji technologií, jako jsou cloudové výpočty, mikroslužby, kontejnery a API. Tyto nástroje výrazně urychlují vývoj, zvyšují flexibilitu nasazení a zjednodušují škálování. Zároveň však přinášejí další technickou složitost, protože aplikace nyní sestávají z mnoha propojených prvků a služeb třetích stran. Rozšíření modelu „software jako služba“ (SaaS), kdy aplikace běží v cloudu a jsou dostupné přes web, umožňuje společnostem využívat složité systémy bez nákladů na vlastní infrastrukturu. To však znamená, že kriticky důležité obchodní procesy se stávají závislými na dostupnosti a bezpečnosti těchto webových aplikací.[4]

S rostoucím počtem webových aplikací se zvětšuje i „plocha útoku“ pro útočníky. Každý bod interakce – ať už se jedná o rozhraní, formulář nebo API – představuje potenciální zranitelnost. Kompromitace webové aplikace může vést nejen k jejímu napadení, ale také k narušení bezpečnosti širších částí informačního systému, včetně databází a interních sítí. Právě proto jsou webové aplikace již dlouhou dobu jedním z hlavních cílů kybernetických útoků. Útočníci používají řadu technik, včetně injekcí, zneužití nedostatků v řízení přístupu, zkreslení uživatelských vstupů a využití chyb v systémových nastaveních. Takové útoky mohou mít vážné důsledky: neoprávněný přístup k citlivým údajům, průnik do účtů, změnu informací nebo narušení dostupnosti služeb.[1][3]

Dnes je více než kdy jindy kriticky důležité si uvědomit rostoucí význam bezpečnosti webových aplikací. Organizace by měly přehodnotit svůj přístup a upřednostnit nejen vytváření funkcí, ale také komplexní testování penetrace, proaktivní správu zranitelností, včasné opravy a nepřetržité sledování kybernetických hrozeb. Slabě chráněná webová aplikace je otevřenou bránou pro katastrofické kybernetické útoky, které mohou způsobit nenapravitelné škody na financích, reputaci a právním postavení společnosti. Vývoj řady metodik, standardů a doporučení byl podmíněn nutností zvýšit bezpečnost webových aplikací. V této souvislosti získala zvláštní význam iniciativa OWASP, která systematicky analyzuje nejčastější hrozby a navrhuje účinné strategie jejich prevence. Nejznámějším výsledkem její činnosti je projekt OWASP Top 10, který představuje konsolidovaný seznam kritických zranitelností a slouží jako referenční základna pro testování bezpečnosti a vývoj zabezpečeného softwaru. Tento přehled také tvoří základ pro následnou analýzu zranitelností, jejíž podrobné posouzení je uvedeno v následujících kapitolách.[2][3]

1.2 Současné trendy kybernetických hrozeb

V posledních letech byl zaznamenán výrazný nárůst počtu kybernetických útoků, které se vyznačují zvýšenou sofistikovaností a organizovaností. V současné době představují kybernetické hrozby značné riziko nejen pro fyzické osoby, ale především pro právnické osoby, státní orgány a objekty kritické infrastruktury. Útočníci používají komplexní přístup, kombinující využití technických zranitelností, metod sociálního inženýrství a automatizovaných prostředků, což vede ke zvýšení účinnosti útoků při současném snížení jejich složitosti.[1]

V současné době je jedním z klíčových trendů v oblasti kybernetické bezpečnosti eskalace útoků pomocí ransomwaru. Tyto programy představují druh škodlivého softwaru, který po infikování systému zašifruje data oběti a požaduje platbu za jejich dešifrování. Je zajímavé, že moderní vyděračské kampaně často fungují na principu „Ransomware as a Service“. V rámci tohoto modelu vývojáři škodlivého softwaru nabízejí své nástroje jiným útočníkům a získávají podíl z vybraných výkupných. Tento obchodní model hrál rozhodující roli v masovém šíření a zvyšování úrovně složitosti útoků pomocí ransomwaru.[6]

Významným trendem je zneužívání zranitelností v softwarových produktech a webových aplikacích. Útočníci aktivně využívají známé bezpečnostní chyby v systémech, které nebyly včas aktualizovány nebo opraveny. Automatizované skenovací nástroje jim umožňují efektivně identifikovat zranitelné systémy přístupné přes internet a následně je využít pro své účely. Podle analytických zpráv je zneužití zranitelností jedním z nejčastějších vektorů útoků, které vedou k úspěšnému napadení systémů.[1][3]

Jedním z klíčových problémů je nárůst počtu útoků zaměřených na dodavatelské řetězce softwaru. Podstatou těchto útoků je napadení dodavatelů softwaru nebo jejich závislých komponent a jejich následné využití k šíření škodlivého kódu mezi dalšími společnostmi. Tento přístup umožňuje útočníkům nepřímo zasáhnout mnoho organizací tím, že zneužijí zranitelnost v jedné komponentě nebo procesu aktualizace. Incident se společností SolarWinds, kdy byla kompromitována aktualizace používaná tisíci organizacemi, je ukázkovým příkladem této hrozby. [7]

Rozvoj cloudových technologií a webových služeb koreluje s nárůstem počtu kybernetických hrozeb zaměřených na cloudovou infrastrukturu a aplikační programové rozhraní (API). Nesprávná konfigurace cloudových služeb, nedostatečná úroveň kontroly přístupu nebo zranitelnosti v API mohou být příčinou neoprávněného přístupu k důvěrným informacím nebo narušení funkčnosti služeb. Vzhledem k vysoké míře propojenosti komponent v cloudovém prostředí mohou bezpečnostní incidenty mít závažné negativní důsledky pro provozní činnost organizace. [8]

Dalším významným aspektem je rozšíření používání automatizovaných nástrojů při kybernetických útocích. Kyberzločinci stále častěji využívají hotová automatizovaná řešení ke skenování zranitelností, provádění útoků a šíření škodlivého obsahu. Automatizace jim umožňuje provádět útoky v obrovském rozsahu a zároveň snižovat náklady. Tento trend otevírá dveře méně zkušeným útočníkům a dává jim možnost využívat složité nástroje k provádění relativně komplexních útoků. [7]

Na základě těchto trendů roste potřeba posílit opatření v oblasti testování bezpečnosti a řízení zranitelností. Organizace by měly neustále hodnotit zabezpečení svých informačních systémů, provádět pravidelné audity a vytvářet účinné mechanismy pro rychlé odhalování a odstraňování zjištěných zranitelností. Pro úspěšné řízení rizik jsou nesmírně důležité ověřené metodiky a bezpečnostní standardy, které umožňují identifikovat nejzávažnější hrozby a stanovit priority při jejich odstraňování. V kontextu bezpečnosti webových aplikací vyniká zejména projekt OWASP a jeho publikace OWASP Top 10, která systematizuje nejkritičtější rizika a nabízí praktická doporučení pro jejich minimalizaci. [2] [8]

1.3 Webové aplikace jako častý cíl kybernetických útoků

Vzhledem k trendům v oblasti digitální transformace a zkomplikování prostředí kybernetických hrozeb představují webové aplikace jeden z nejčastějších cílů kybernetických útoků. Základní příčinou této zranitelnosti je jejich přímá dostupnost z internetu, která útočníkům umožňuje provádět vzdálené útoky bez nutnosti fyzického proniknutí do infrastruktury organizace. Webové aplikace tak představují typický vstupní bod do informačního systému, který může být potenciálně využit k dalšímu šíření destruktivní aktivity uvnitř organizace.[1]

Významnou roli hraje také skutečnost, že webové aplikace zpracovávají citlivé informace, včetně osobních údajů uživatelů, přihlašovacích údajů, platebních informací, obchodních údajů a interních firemních dokumentů. Úspěšné zneužití zranitelností webové aplikace může vést k neoprávněnému přístupu a úniku citlivých údajů, což má za následek značné právní, finanční a reputační důsledky. Je třeba poznamenat, že v evropském právním prostoru je ochrana osobních údajů upravena legislativními akty, jako je nařízení GDPR, které v případě bezpečnostních incidentů může vést k uložení podstatných finančních sankcí.[9]

Webové aplikace jsou často terčem útoků kvůli své komplexní architektuře, která výrazně rozšiřuje „útočnou plochu“. Moderní aplikace se skládají z mnoha propojených komponent – od uživatelského rozhraní po databáze, které komunikují prostřednictvím různých protokolů. Každá taková interakce a každá komponenta představuje potenciální místo pro

vznik zranitelností, ať už kvůli nesprávné validaci vstupů, chybám v bezpečnostních mechanismech, nesprávnému nastavení nebo použití zranitelných externích knihoven.[4][5]

Rychlý vývoj, který je typický pro přístupy DevOps a CI/CD, také přispívá k vzniku bezpečnostních problémů. Ačkoli tyto metodiky urychlují vydávání nových verzí, mohou vést k tomu, že bezpečnostní otázky nebudou řádně integrovány do vývojového cyklu. Nedostatečné testování nebo absence bezpečnostních kontrol v raných fázích může vést k tomu, že zranitelnosti budou odhaleny až v již fungující aplikaci.[4][5]

Významnou roli při využívání zranitelností webových aplikací hraje dostupnost specializovaných nástrojů pro testování bezpečnosti. Nástroje vyvinuté pro účely penetračního testování a výzkumu v oblasti kybernetické bezpečnosti mohou být zneužity útočníky k odhalení slabých míst. Automatizované skenery zajišťují vysokou rychlost a škálovatelnost detekce zranitelností ve webových aplikacích, což umožňuje útočníkům provádět systematické vyhledávání aplikací se známými zranitelnostmi v globální síti.[5]

Specifickým aspektem webových aplikací je jejich úzká provázanost s ostatními komponenty informační infrastruktury. Webové aplikace často fungují jako rozhraní pro přístup k databázím, cloudovým službám, interním aplikacím nebo identifikačním systémům organizace. Po získání kontroly nad webovou aplikací může útočník tuto pozici využít k dalšímu postupu v infrastruktuře organizace, například k získání přístupu k databázím nebo interním systémům. Tento typ útoku je často klasifikován jako laterální pohyb po síti (lateral movement).[1][2][4]

Vzhledem k zásadnímu významu bezpečnosti webových aplikací v kontextu kybernetické bezpečnosti organizace systematicky zavádějí komplexní testovací strategie, které zahrnují penetrační testování, analýzu zranitelností a bezpečnostní audity. Pro účinné odhalování a prioritizaci nejvýznamnějších zranitelností se široce používá metodika navržená projektem OWASP. Tento seznam kritických rizik slouží jako referenční příručka pro provádění bezpečnostních testů a jako koncepční základ pro následnou podrobnou analýzu zranitelností, která bude představena v následující kapitole.[2]

1.4 Projekt OWASP a jeho význam v praxi

Vzhledem k rostoucímu významu webových aplikací a vývoji kybernetických hrozeb vznikla objektivní potřeba vyvinout systémové metodiky pro identifikaci a řízení bezpečnostních rizik softwaru. V kontextu této výzvy představuje projekt OWASP jednu z nejvýznamnějších iniciativ zaměřených na zvýšení úrovně zabezpečení webových aplikací a softwarových systémů. OWASP je mezinárodní otevřená komunita sdružující odborníky v oblasti kybernetické bezpečnosti, vývojáře, výzkumníky a zástupce organizací, jejichž posláním je zlepšovat bezpečnost softwaru prostřednictvím výměny odborných znalostí, provádění vědeckého výzkumu a vytváření aplikovaných nástrojů a metodik.[2]

Projekt OWASP, založený v roce 2001, se postupně etabloval jako jeden z nejuznávanějších hráčů v oblasti bezpečnosti webových aplikací. Organizace funguje jako nezisková iniciativa a opírá se převážně o dobrovolnou spolupráci odborníků z různých jurisdikcí. Klíčovou charakteristikou projektu je jeho otevřenost, která zajišťuje širokou dostupnost

výsledků činnosti OWASP pro akademický a komerční sektor. Hlavním cílem projektu OWASP je zvýšit povědomí o existujících bezpečnostních rizicích v oblasti vývoje softwaru a poskytnout praktické nástroje a doporučení pro jejich neutralizaci. V rámci této iniciativy bylo realizováno mnoho projektů zaměřených na různé aspekty bezpečnosti aplikací, včetně metodik bezpečného vývoje, testování zranitelnosti, analýzy rizik a školení bezpečnostních specialistů. Uvedené projekty jsou aktivně využívány vývojáři, bezpečnostními analytiky, odborníky na testování penetrace a také vedoucími pracovníky odpovědnými za správu bezpečnosti informačních systémů.[2]

Dokument „OWASP Top 10“ je jedním z nejznámějších výsledků činnosti organizace OWASP. Jedná se o konsolidovaný seznam nejzávažnějších bezpečnostních hrozeb a typických zranitelností charakteristických pro webové aplikace. Tento seznam je sestaven na základě empirické analýzy skutečných incidentů a agregovaných údajů od předních subjektů v oblasti kybernetické bezpečnosti a jeho pravidelné aktualizace zajišťují jeho relevanci v kontextu vývoje hrozeb a technologií. Význam OWASP se však neomezuje pouze na zveřejnění tohoto seznamu. Organizace také vyvíjí a poskytuje širokou škálu dalších metodik a nástrojů, které se aktivně používají v oblasti bezpečnosti aplikací. Mezi nimi vyniká OWASP Application Security Verification Standard (ASVS), který nabízí strukturovaný základ pro ověřování bezpečnosti, a OWASP Testing Guide, který podrobně popisuje metodiky testování bezpečnosti webových aplikací. Tyto dokumenty slouží jako základní praktické pokyny pro implementaci bezpečnostních opatření v průběhu celého životního cyklu vývoje softwaru. [2]

Praktická hodnota OWASP se projevuje v široké škále použití. Organizace integrují doporučení OWASP do svých bezpečnostních politik, používají je pro audity a zvyšování kvalifikace zaměstnanců v oblasti bezpečnosti aplikací. Vývojáři se při navrhování zabezpečené architektury řídí zásadami OWASP a odborníci na bezpečnost používají metodiky OWASP k provádění penetračních testů a analýzy zranitelností. To činí OWASP základním zdrojem pro efektivní identifikaci a řízení bezpečnostních rizik. Kromě toho se OWASP aktivně podílí na vzdělávacích aktivitách, organizuje konference, semináře a setkání, která podporují výměnu zkušeností mezi odborníky z různých odvětví a stimulují pokrok v oblasti kybernetické bezpečnosti. [2]

Vzhledem ke své otevřenosti, praktickému zaměření a širokému uplatnění v praxi se projekt OWASP stal jedním z nejdůležitějších referenčních zdrojů v oblasti bezpečnosti webových aplikací. Jeho metodiky a doporučení se používají při vývoji zabezpečeného softwaru, při testování bezpečnosti aplikací a také při řízení bezpečnostních rizik v organizacích. Vzhledem k tomu, že tato diplomová práce se zabývá zranitelností webových aplikací a jejich praktickým testováním, představuje projekt OWASP a jeho výsledky důležitý teoretický základ pro další část práce. [2]

1.5 OWASP Top 10 – struktura a metodika

V předchozích částech jsem podrobně rozebral rostoucí význam webových aplikací, současné trendy v oblasti kybernetických hrozeb a důvody, proč se webové aplikace tak často

stávají terčem útočníků. S tímto kontextem přímo souvisí dokument OWASP Top 10. Jedná se o jeden z nejvýznamnějších zdrojů pro identifikaci a diskusi o kritických bezpečnostních hrozbách webových aplikací. Podle oficiálních údajů OWASP má tento standardizovaný dokument zvýšit povědomí o bezpečnosti webových aplikací a představuje konsensuální názor na nejvýznamnější rizika v této oblasti. Jeho význam přesahuje rámec pouhého technického seznamu zranitelností; především vytváří společný jazyk pro vývojáře, bezpečnostní experty, auditory a vedoucí pracovníky společností. [2]

Z metodologického hlediska je důležité zdůraznit, že OWASP Top 10 si neklade za cíl být úplným katalogem všech možných zranitelností webových aplikací. Jeho úkolem není nahradit podrobné bezpečnostní standardy, penetrační testování nebo specializované metodiky bezpečného vývoje. Místo toho se dokument zaměřuje na ty typy rizik, které se pravidelně vyskytují, mají významný dopad a jsou dostatečně univerzální, aby byly relevantní pro různé aplikace a technologická prostředí. Právě tato kombinace srozumitelnosti, praktického významu a široké použitelnosti vysvětluje, proč je OWASP Top 10 tak často používán v profesionální praxi jako výchozí základ pro diskusi o bezpečnosti aplikací. [2]

Význam dokumentu OWASP Top 10 je umocněn jeho zaměřením na kategorie rizik, nikoli na izolované technické chyby. To je strategicky důležité pro řízení bezpečnosti, protože organizace se obvykle potýkají nikoli s jednotlivými softwarovými vadami, ale s celými skupinami problémů, které vznikají v podobných kontextech vývoje, architektury nebo provozu. Kategorie, jako jsou chyby v řízení přístupu, nedostatečná kryptografická ochrana, zranitelnosti injekcí nebo nesprávná nastavení zabezpečení, představují rozsáhlé problémové oblasti, ve kterých se konkrétní technické chyby mohou projevovat různými způsoby. Díky tomuto přístupu je OWASP Top 10 nejen vzdělávacím zdrojem, ale také nepostradatelným praktickým průvodcem pro efektivní stanovení priorit v oblasti zabezpečení. [2]

Metodický přístup k vytvoření OWASP Top 10 se vyznačuje tím, že je založen na reálných datech a rozsáhlých zkušenostech komunity bezpečnostních specialistů. Oficiální materiály k verzi 2025 upřesňují, že zůstává „založena na datech“, ale není „slepě orientována na data“. To znamená, že konečná struktura dokumentu není čistě mechanickým produktem statistické analýzy, ale představuje syntézu kvantitativních údajů, odborných hodnocení a výsledků průzkumů komunity. OWASP přímo uvádí, že 12 kategorií bylo vytvořeno na základě dostupných dat, zatímco dvě další mohly být upraveny nebo přesunuty s ohledem na názory komunity. Tento přístup je nesmírně důležitý, protože samotná data odrážejí především minulost, zatímco výzkum v oblasti bezpečnosti musí být proaktivní a zohledňovat nové hrozby a metody útoků. Předložená metodika dokazuje, že OWASP Top 10 by neměl být interpretován výhradně jako statistický žebříček nejčastěji se vyskytujících zranitelností. Ve své podstatě je výsledkem integrace analýzy dat, odborného posouzení a komplexního zvážení bezpečnostních aspektů. Tato skutečnost má zásadní význam pro akademickou a praktickou činnost, protože hodnocení bezpečnosti aplikací nelze redukovat na pouhé zaznamenávání četnosti výskytu jednotlivých zranitelností. Některé problémy se mohou vyznačovat nižší četností, ale jejich potenciální důsledky mohou být mimořádně závažné; jiné se projevují po dlouhou dobu v různých formách a vyžadují systematický přístup

k jejich odstranění. OWASP Top 10 se tedy zaměřuje nejen na odrážení aktuálního stavu známých problémů, ale také na jejich význam v kontextu rizik pro organizace a jejich informační systémy. [2]

Tento dokument byl vypracován s ohledem na jeho praktickou hodnotu v každé fázi životního cyklu softwaru. Ve fázi návrhu slouží jako nástroj pro včasné odhalení kriticky důležitých aspektů, které je třeba zohlednit při přijímání architektonických rozhodnutí. V průběhu vývoje pomáhá určit oblasti, které vyžadují přísnější ověřování vstupních dat, spolehlivé řízení přístupových práv nebo bezpečné zpracování důvěrných informací. Ve fázi testování slouží dokument jako základ pro cílené provádění penetračních testů a bezpečnostních auditů se zaměřením na nejrizikovější komponenty aplikace. A konečně, ve fázi provozu pomáhá pochopit, jaké typy zranitelností mohou mít významný dopad na důvěrnost, integritu nebo dostupnost poskytovaných služeb. [2]

Praktická významnost OWASP Top 10 je dána jeho integrací do širšího komplexu metodik OWASP. Pro provedení hloubkové a přesné verifikace bezpečnosti webových aplikací má zásadní význam například OWASP Application Security Verification Standard (ASVS). Tento standard podrobně popisuje požadavky na testování technických prostředků ochrany a zároveň funguje jako soubor kritérií pro vývojáře v kontextu bezpečného vývoje. Zatímco OWASP Top 10 plní především funkci zvyšování povědomí a stanovení priorit, ASVS umožňuje transformovat obecné kategorie rizik na konkrétní a ověřitelné bezpečnostní požadavky. V praxi tedy OWASP Top 10 často slouží jako vstupní bod a ASVS jako následná, podrobnější metodika. [2]

Z profesionálního hlediska přispívá OWASP Top 10 k podstatnému zlepšení standardizace komunikačních procesů mezi technickými a netechnickými stakeholdery v organizaci. Účinně vyrovnává rozdíly v úrovni detailů a terminologii, které jsou charakteristické pro interakci bezpečnostních specialistů, vývojářů, projektových manažerů a vrcholového vedení při hodnocení rizik. Univerzálnost formulací kategorií OWASP Top 10 zajišťuje jejich srozumitelnost na manažerské úrovni a zároveň zachovává dostatečnou technickou použitelnost pro analýzu zranitelností. Jeho hodnota tak přesahuje rámec čistě odborného dokumentu a činí z něj klíčový nástroj strategického řízení bezpečnosti. [2]

Další důležitou vlastností je pravidelné aktualizování dokumentu. Vzhledem k neustálým změnám ve webových technologiích, přístupech k vývoji a metodách útoků je třeba přehodnotit také klasifikaci nejvýznamnějších bezpečnostních hrozeb. Oficiální web projektu nabízí aktuální verzi OWASP Top 10:2025 a také starší verze pro srovnání. To umožňuje analyzovat, jak se mění priority v souvislosti s různými riziky a jak nové technologické trendy ovlivňují reálné postupy zajišťování bezpečnosti. Pro akademický výzkum je tato kontinuita obzvláště cenná, protože jasně ukazuje, že bezpečnost webových aplikací je živá, vyvíjející se oblast, a nikoli statický soubor problémů, které lze vyřešit jednou provždy. [2]

V kontextu této diplomové práce představuje OWASP Top 10 významný prvek, protože tvoří systematický základ pro následný analytický výzkum. Jednotlivé kategorie této klasifikace umožňují provést strukturovanou analýzu mechanismů vzniku zranitelností, jejich potenciálních vektorů zneužití a jejich dopadu na informační systémy organizací. Zároveň je

tato struktura díky svému širokému rozšíření a uznání v odborné praxi vhodná pro sestavení praktické části práce zaměřené na laboratorní úkoly a simulaci penetračního testování. Následující kapitola bude věnována podrobnému rozboru každé z kategorií OWASP Top 10 a jejich roli v zajištění bezpečnosti moderních webových aplikací. [2]

1.6 Shrnutí kapitoly

Současná situace v oblasti kyberbezpečnosti webových aplikací ukazuje, že tyto aplikace jsou zároveň základem moderních informačních systémů i nejčastějším terčem kyberútoků. Jejich význam roste díky digitalizaci, cloudovým službám, architektuře API a širokému využití v podnikové, státní i soukromé sféře. Tento vývoj však s sebou nese zvětšení útočné plochy a nárůst závažnosti hrozeb, které mohou ohrozit důvěrnost, integritu a dostupnost dat a služeb.

Analýza aktuálních hrozeb ukazuje, že úspěšné útoky na webové aplikace jsou málokdy způsobeny jedinou izolovanou bezpečnostní mezerou. Místo toho útočníci kombinují různé faktory: od obecně známých zranitelností a chyb v konfiguraci až po nedostatečné řízení přístupu, opožděné aktualizace a bezpečnostní problémy v dodavatelském řetězci softwaru. Automatizace útoků jim výrazně usnadňuje vyhledávání a využívání zranitelných systémů ve velkém měřítku. Je proto zřejmé, že ochrana webových aplikací by neměla být jednorázovou akcí, ale nepřetržitým a komplexním procesem.

Jedním z klíčových závěrů této kapitoly bylo potvrzení, že projekt OWASP, a zejména „OWASP Top 10“, představuje nenahraditelný praktický a metodický průvodce pro identifikaci nejkritičtějších rizik webových aplikací. „OWASP Top 10“ přesahuje rámec pouhého výčtu typických chyb a nabízí strukturovaný přístup k hodnocení bezpečnostních problémů, který zohledňuje jejich dopad a aktuálnost v reálných podmínkách. To z něj činí ideální výchozí bod pro další fázi práce, která bude věnována hloubkové analýze konkrétních zranitelností a vývoji praktických laboratorních cvičení pro jejich názornou demonstraci.

2 Detailní analýza zranitelností dle OWASP Top 10

Předchozí kapitola byla věnována konceptu OWASP Top 10, který je prezentován jako referenční model pro identifikaci nejkritičtějších bezpečnostních rizik webových aplikací. Byla popsána jeho metodika, architektura a relevance pro praxi zajištění bezpečnosti. Jelikož však samotný seznam kategorií nabízí pouze souhrnný přehled rizik, je pro dosažení skutečného porozumění problémům bezpečnosti webových aplikací zapotřebí hlubší analýza jednotlivých kategorií zranitelností, včetně vysvětlení mechanismů jejich vzniku, způsobů zneužití a jejich dopadů na informační systémy organizací. [2][5]

Složitost moderních webových aplikací, daná jejich vícesložkovou strukturou a interakcí prostřednictvím různých protokolů, vytváří mnoho potenciálních míst pro vznik zranitelností. Tyto zranitelnosti se mohou projevovat na různých úrovních: od vnitřní logiky a zpracování uživatelských vstupů až po mechanismy autorizace a konfigurace serverové infrastruktury. Nečestné osoby mohou tyto zranitelnosti zneužít k získání neoprávněného přístupu k důvěrným informacím, jejich zkreslení nebo úplnému vyřazení služby z provozu. Proto je kriticky důležité analyzovat každý typ zranitelnosti nejen z technického hlediska, ale také s ohledem na jeho potenciální dopad na celkovou bezpečnost organizace.[5][10]

Při analýze zranitelností webových aplikací je třeba přemýšlet o tom, jak je lze využít v praxi. Útočníci se často neomezují na jednu chybu, ale kombinují několik slabých míst, aby provedli sofistikovanější útoky. Například zranitelnost v systému řízení přístupu může otevřít cestu k citlivým údajům a jiná zranitelnost související s nesprávnou konfigurací může umožnit zvýšení úrovně přístupu nebo proniknutí do jiných částí podnikové infrastruktury. Takový přístup založený na postupném využití několika zranitelností je charakteristický pro moderní kybernetické hrozby.[10][11]

Důležitým aspektem je, že značná část zranitelností webových aplikací vyplývá z opakujících se chyb, ke kterým dochází ve fázích návrhu nebo implementace softwaru. Příklady takových chyb zahrnují nedostatečnou validaci vstupních dat, nesprávnou správu přístupových oprávnění a nevhodné zpracování citlivých informací. Tyto problémy se vyskytují s vysokou mírou opakovatelnosti v různých technologických stackech a aplikačních prostředích, což je důvodem jejich zařazení do OWASP Top 10 jako nástroje pro systematickou identifikaci a minimalizaci rizik. [2][5]

Cílem této kapitoly je komplexní zkoumání každé ze zranitelností klasifikovaných v rámci OWASP Top 10. U každé kategorie bude uvedeno vysvětlení její podstaty a mechanismu vzniku. Dále budou posouzeny typické scénáře zneužití, které demonstrují praktické metody využití těchto zranitelností. Analýza bude zahrnovat také hodnocení potenciálního dopadu na informační aktiva organizace se zaměřením na aspekty důvěrnosti, integrity a dostupnosti informací. Tento přístup zajistí hlubší pochopení jak technické povahy konkrétních zranitelností, tak jejich strategického významu pro kybernetickou bezpečnost organizací. Získané znalosti budou sloužit jako teoretický základ pro praktickou část diplomové

práce, která spočívá ve vývoji laboratorních úkolů simulujících reálné scénáře penetračního testování webových aplikací.

2.1 Broken Access Control

Broken Access Control popisuje zranitelnost, při které systém neprovádí řádnou validaci oprávnění uživatele k přístupu k údajům nebo funkcím. To otevírá prostor pro neoprávněné čtení, úpravu nebo odstranění informací, stejně jako pro získání přístupu k privilegovaným funkcím. Tato třída zranitelností se stabilně umisťuje na předních pozicích v žebříčcích hrozeb pro webové aplikace. [2][5]

Definice zranitelnosti

Zranitelnost Broken Access Control se projevuje v případě nedostatečné implementace mechanismů řízení přístupových práv v aplikaci. To vede k tomu, že uživatelé získají možnost provádět akce, které přesahují rámec jejich přidělených rolí nebo oprávnění. V důsledku toho může dojít ke kompromitaci důvěrných dat, změně záznamů patřících jiným uživatelům nebo k použití funkcí s vyššími oprávněními bez řádné autorizace. [2][5]

Mechanismus vzniku zranitelnosti

Hlavní bezpečnostní chyba spočívá v nedostatečné implementaci mechanismů řízení přístupu na straně serveru. Ačkoli aplikace pracuje s identifikátory objektů a uživatelskými rolemi, neprovádí ověření skutečných oprávnění oprávněného subjektu k přístupu k požadovanému zdroji. To umožňuje útočníkovi manipulovat s parametry v URL, formulářích nebo HTTP požadavcích, čímž získá neoprávněný přístup k důvěrným údajům. [1]

Častou chybou je porušení zásady oddělení rolí, kdy jsou běžným uživatelům poskytovány privilegované administrativní funkce. K takové chybě může dojít v důsledku nedostatečné validace požadavků na straně serveru, a to i v případě, že jsou na straně klienta implementována bezpečnostní opatření, jako je skrytí prvků uživatelského rozhraní. [5][10]

Scénáře zneužití

Častým scénářem je možnost manipulace s identifikátory objektů nebo uživatelů obsaženými v URL adresách a HTTP požadavcích. To útočníkovi umožňuje získat neoprávněný přístup k datům jiných uživatelů, ať už jde o jejich prohlížení nebo úpravu. Klasickým příkladem je změna hodnoty parametru id ve webové adrese nebo v rámci API požadavku. [2]

Alternativní scénář předpokládá neoprávněný přístup k administrativním funkcím. Pokud na serveru chybí řádná kontrola uživatelských rolí, útočník získá možnost prohlížet seznam uživatelů, měnit systémová nastavení nebo vytvářet nové účty s rozšířenými oprávněními. [10][11]

Tato zranitelnost je často zneužívána v kombinaci s dalšími útočnými vektory. Po kompromitaci účtu běžného uživatele může útočník využít nedostatky v mechanismech řízení přístupu k eskalaci oprávnění nebo k neoprávněnému přístupu k důvěrným informacím. [5]

Dopady na informační systémy

Problémy spojené s nedostatečnou kontrolou přístupu přímo ohrožují tři klíčové aspekty bezpečnosti informačních systémů: důvěrnost, integritu a dostupnost. Typickým důsledkem těchto zranitelností je neoprávněný přístup k datům patřícím jiným uživatelům nebo k důvěrným informacím organizace. Může se jednat o osobní údaje, obchodní tajemství nebo záznamy z databází. Takové incidenty nejen porušují důvěrnost, ale také představují značná právní a reputační rizika. [5]

Významným rizikem spojeným s tímto problémem je možnost neoprávněného zásahu do integrity dat. Útočníci mohou provádět úpravy, mazání nebo vytváření záznamů, stejně jako si udělovat rozšířená oprávnění nebo měnit konfiguraci systému. Takové činnosti narušují spolehlivost a důvěryhodnost informací, na nichž je založeno fungování systému. [2][10]

Umožnění neoprávněného přístupu k administrativním funkcím vytváří podmínky pro narušení stability systému. Takové činnosti mohou zahrnovat změnu konfiguračních parametrů, blokování ověřených uživatelů nebo vyřazení jednotlivých modulů aplikace z provozu. Kromě toho je narušení mechanismů řízení přístupu často výchozím bodem pro následné útoky, zejména pro eskalaci oprávnění a hlubší proniknutí do systému. V souvislosti s tím patří správná implementace řízení přístupu k nejzásadnějším požadavkům na bezpečnost webových aplikací. [10][11]

2.2 Security Misconfiguration

Security Misconfiguration představuje skupinu zranitelností způsobených nesprávným nebo neúplným nastavením různých komponent informačního systému, včetně aplikací, serverů, databází, frameworků a cloudových služeb. Je důležité poznamenat, že i software vyvinutý v souladu s vysokými bezpečnostními standardy se může stát terčem útoku, pokud je nasazen v neoptimálně nakonfigurovaném prostředí. Z tohoto důvodu tento typ hrozeb stabilně zaujímá vedoucí pozice mezi riziky spojenými s webovými aplikacemi. [2]

Definice zranitelnosti

Podle OWASP, je Security Misconfiguration definována jako skupina zranitelností, které vznikají v důsledku nesprávného nebo neúplného nastavení bezpečnostních parametrů v rámci softwaru nebo jeho infrastruktury. Mezi běžné projevy této zranitelnosti patří použití výchozích přihlašovacích údajů, přítomnost neaktivních, ale zapnutých služeb, poskytování nadměrných přístupových práv, absence implementovaných ochranných mechanismů a také dostupnost administrativních rozhraní pro širokou veřejnost. Podobné nedostatky

v konfiguraci mohou vést ke kompromitaci dat, neoprávněnému vniknutí do systému nebo jiným formám zneužití. [2][12]

Mechanismus vzniku zranitelnosti

Hlavní příčinou vzniku zranitelností tohoto typu je nesprávná správa konfiguračních parametrů. Moderní webové aplikace představují víceúrovňové systémy, které zahrnují webové servery, databáze, frameworky, kontejnery a cloudové služby. Každá z těchto komponent vyžaduje zajištění bezpečné konfigurace. Nedodržování standardních nastavení nebo absence adekvátních opatření k posílení bezpečnosti (hardening) u kteréhokoli z prvků vytváří předpoklady pro vznik zranitelností a následných útoků. [2]

Často se zdrojem problémů stává použití standardních nastavení, která byla původně navržena pro rychlé nasazení nebo testování. Taková konfigurace zpravidla nesplňuje požadavky na bezpečný provoz a může zahrnovat nechráněné síťové porty, přednastavené přihlašovací údaje, přístupné rozhraní pro správu nebo nedostatečně přísné bezpečnostní zásady. [2][12]

Scénáře zneužití

Častým vektorem útoku je únik důvěrných údajů prostřednictvím příliš podrobných chybových hlášení. Takové informace umožňují útočníkovi identifikovat používané technologie, verze softwaru nebo architekturu aplikace, což může být následně využito k provedení cílených útoků. [2]

Další možný způsob útoku souvisí s neoprávněným přístupem k veřejně dostupným administračním panelům, testovacím funkcím, otevřeným síťovým portům nebo konfiguračním souborům. Absence adekvátních bezpečnostních opatření pro tyto komponenty může útočníkům poskytnout přímou cestu ke kompromitaci systému. Kromě toho představuje neméně závažnou hrozbu i zveřejnění důvěrných konfiguračních informací, například ve veřejných repozitářích nebo nezabezpečených úložištích souborů. [2]

Dopady na informační systémy

Security Misconfiguration představuje vážnou hrozbu, která může ohrozit důvěrnost, integritu a dostupnost informačních systémů. Typickým projevem této zranitelnosti je poskytnutí neoprávněného přístupu ke kriticky důležitým komponentám, jako jsou administrátorské panely, servisní rozhraní nebo konfigurační soubory obsahující citlivé systémové informace. V důsledku toho získávají útočníci možnost kompromitovat databáze, aplikační služby a další prvky infrastruktury. [12]

Vážnou hrozbou je také možnost neoprávněné změny konfigurace nebo zásahu do fungování aplikace. Po získání přístupu k administrativním nástrojům nebo privilegovaným službám může útočník upravovat systémové parametry, deaktivovat ochranné mechanismy,

vytvářet nové uživatelské účty nebo připravovat infrastrukturu pro následné útoky. Takové akce vedou ke kompromitaci integrity systému a podstatnému zvýšení celkové úrovně bezpečnostních rizik. [2]

Nesprávná konfigurace může vést k závažným poruchám ve fungování služby, a to až k její úplné nedostupnosti. Mezi takové scénáře patří zejména zneužití otevřených rozhraní, přetížení služby nebo změna kriticky důležitých provozních parametrů. Závažnost této kategorie problémů je umocněna tím, že jejich příčina často nespočívá v chybách programového kódu, ale ve specifikách prostředí, ve kterém je aplikace provozována. V souvislosti s tím má pro minimalizaci rizik prvořadý význam systematické posilování bezpečnostních opatření, ověřování konfigurace a provádění pravidelných bezpečnostních auditů. [2][12]

2.3 Software Supply Chain Failures

Moderní vývoj webových aplikací se vyznačuje širokým využíváním externích knihoven, komponent s otevřeným zdrojovým kódem a automatizovaných systémů sestavování a nasazování. Tato praxe s sebou nese zvýšenou zranitelnost dodavatelského řetězce softwaru. V případě kompromitace kteréhokoli z těchto prvků nebo procesů mohou útočníci vložit škodlivý kód do finální verze aplikace, a to i v případě, že vlastní kód vývojáře neobsahuje žádné zranitelnosti. [2]

Definice zranitelnosti

Software Supply Chain Failures se projevují jako zranitelnosti, které ohrožují integritu a spolehlivost softwarových komponent používaných ve fázích vývoje, kompilace, aktualizace nebo provozu aplikací. Tyto zranitelnosti mohou mít podobu kompromitovaných knihoven, nedostatečně ověřených aktualizací nebo nezabezpečených kompilačních procesů. V důsledku těchto selhání může být do aplikace vložen škodlivý nebo nespolehlivý kód. [2]

Místo toho, aby útočník přímo zaútočil na cílovou organizaci, využívá kompromitaci důvěryhodné složky, jako je dodavatel, repozitář, balíček nebo mechanismus aktualizací. Tento přístup mu umožňuje obejít standardní bezpečnostní opatření a zároveň zasáhnout velké množství systémů. [5]

Mechanismus vzniku zranitelnosti

Uvedené zranitelnosti se projevují zejména v situacích, kdy aplikace využívají automatické stahování externích závislostí nebo používají procesy kontinuální integrace a kontinuálního nasazení (CI/CD) bez odpovídajících mechanismů kontroly integrity. Neověřené komponenty se mohou stát vektorem pro vniknutí škodlivého kódu do procesu sestavování. [2]

Používání knihoven s otevřeným zdrojovým kódem bez řádné kontroly jejich bezpečnosti a zdrojů s sebou nese zvýšená rizika. Útočníci mohou zneužít zranitelnosti vyplývající ze složitosti závislostí, kompromitace repozitářů nebo nespolehlivých aktualizací. Výrazným příkladem je incident se společností SolarWinds, který názorně ukázal, jak může

kompromitace mechanismu aktualizací vést k rozsáhlému šíření škodlivého kódu prostřednictvím důvěryhodného softwaru. [7][10]

Scénáře zneužití

Typický scénář předpokládá kompromitaci externí závislosti, která je následně integrována do cílové aplikace. To má za následek automatické šíření škodlivého kódu do všech postižených systémů, které používají kompromitovanou komponentu. Dalším běžným vektorem útoku je kompromitace sestavovacího pipeline nebo CI/CD. Po získání přístupu k nástrojům pro sestavení nebo nasazení může útočník provést neviditelné změny v konečném artefaktu nebo do něj vložit škodlivé moduly. [2] [11]

Dalším možným způsobem útoku je nesprávná správa procesu aktualizací. Pokud nejsou aktualizace dostatečně ověřovány, útočníci získávají možnost vkládat do softwaru škodlivé úpravy nebo kompromitovat důvěryhodnou infrastrukturu aktualizací za účelem šíření škodlivého kódu. [7] [10]

Dopady na informační systémy

Odhalené zranitelnosti s sebou nesou značná rizika, protože jejich dopad často přesahuje rámec jednotlivé aplikace či organizace. Kompromitace komponenty integrované do mnoha systémů může vyvolat kaskádové šíření škodlivého kódu, které zasáhne rozsáhlé síťové infrastruktury. V důsledku toho se lokální incident promění v událost velkého rozsahu, která zasáhne mnoho subjektů. [2]

Jedním z hlavních negativních dopadů je ohrožení důvěrnosti informací. V případě úspěšného útoku na zranitelnou komponentu získá útočník možnost získat přístup k autentizačním údajům, interní dokumentaci, záznamům v databázích a dalším informacím, které představují obchodní tajemství nebo osobní údaje. Vzhledem k tomu, že škodlivý software se často maskuje jako legitimní aplikace, je jeho včasné odhalení složitým úkolem, což může vést k dlouhodobým a nepozorovaným únikům dat. [2][5]

Dalším významným důsledkem je narušení systémové integrity. Začlenění škodlivého kódu do aplikace nebo její aktualizace může vyvolat odchylky od běžného fungování, vést k neoprávněným manipulacím s daty nebo nainstalovat skryté zadní vrátka pro následné proniknutí. Odhalení takovýchto narušení je obtížné, protože se maskují jako součást důvěryhodné komponenty nebo legitimního procesu sestavení. [10]

Neméně závažnou hrozbu představuje ohrožení dostupnosti. V případě narušení integrity jakékoli součásti ji mohou útočníci využít k provádění sabotáže, přerušení fungování služeb nebo rozšíření dopadu na další prvky infrastruktury. U systémů kritického významu může mít takový incident negativní dopad na provozní činnost společnosti, narušit plynulost poskytování služeb a podkopat důvěru uživatelů. [2][7]

Z hlediska kyberbezpečnosti představuje tato kategorie zvýšené riziko, protože využívá důvěru v legitimní softwarové produkty a pracovní postupy. Účinná ochrana proto vyžaduje

nejen přísné standardy bezpečného vývoje, ale také komplexní opatření zahrnující ověřování závislostí, kontrolu aktualizací, zajištění integrity sestavovacího konvejeru a systematický audit dodavatelského řetězce.

2.4 Cryptographic Failures

Kryptografie je základním nástrojem pro zajištění důvěrnosti citlivých informací, včetně osobních údajů, přihlašovacích údajů a finančních informací. Nesprávné použití kryptografických metod však může mít opačný účinek, kdy nejenže nezajistí náležitou ochranu, ale také způsobí značné zranitelnosti. Vzhledem k tomu klasifikuje OWASP kryptografické chyby jako jedno z nejzávažnějších bezpečnostních rizik webových aplikací. [2]

Definice zranitelnosti

Cryptographic Failures zahrnují zranitelnosti vyplývající z nedostatečné ochrany informací pomocí kryptografických metod. Může se jednat o úplnou absenci šifrování, použití zastaralých kryptografických algoritmů, použití nevhodných hashových funkcí nebo nesprávnou správu kryptografických klíčů. [2]

Z hlediska zabezpečení je klíčová nejen volba samotného algoritmu, ale také jeho bezchybná implementace a harmonická integrace do celkové architektury systému. Nedostatky ve správě klíčů, použití nestabilních hashových funkcí nebo nesprávné parametry šifrování mohou výrazně ohrozit spolehlivost ochrany informací. [5]

Mechanismus vzniku zranitelnosti

Nedostatky v oblasti bezpečnosti související s kryptografií jsou často způsobeny chybami při implementaci kryptografických protokolů nebo použitím neoptimálních nástrojů. Zvláštní riziko představuje vývoj vlastních kryptografických algoritmů, na rozdíl od používání standardizovaných a osvědčených knihoven. Problémy také často vznikají při správě přihlašovacích údajů, například při použití nedostatečně odolných hashových funkcí nebo při ukládání důvěrných informací bez adekvátních ochranných opatření. [10] [11]

Nedostatečná úroveň kryptografické ochrany komunikačního kanálu mezi klientskou aplikací a serverem rovněž představuje značné riziko. Absence šifrování nebo použití zastaralých protokolů otevírá útočníkům možnost neoprávněného zachycení a úpravy přenášených informací. [13]

Scénáře zneužití

Častým směrem útoku je narušení bezpečnosti databází, které obsahují důvěrné informace v nešifrovaném formátu nebo s použitím neúčinných metod hashování. To útočníkům

umožňuje rychle získat přístup k přihlašovacím údajům uživatelů a dalším citlivým informacím. [11]

Alternativním vektorem útoku je kompromitace komunikačního kanálu používaného k výměně dat mezi klientem a serverem. V případě nedostatečné kryptografické ochrany datového toku může útočník získat přístup k autentizačním údajům nebo jiným citlivým informacím. V závažnějších případech je také možná modifikace přenášených dat. Takové incidenty jsou typické pro útoky typu „man-in-the-middle“ (člověk uprostřed). [1]

Zranitelnosti v bezpečnostním systému mohou vzniknout nejen v důsledku úmyslných útoků, ale také v důsledku použití zastaralých nebo nedostatečně spolehlivých algoritmů. Tyto algoritmy, které lze snadno prolomit pomocí běžně dostupných nástrojů, představují zvláštní nebezpečí pro systémy, které nebyly včas aktualizovány.

Dopady na informační systémy

Porušení v oblasti kryptografie přímo ohrožují tři základní aspekty informační bezpečnosti: důvěrnost, integritu a dostupnost. Typickým důsledkem takových selhání je únik citlivých údajů, včetně přihlašovacích údajů uživatelů, osobních identifikátorů, finančních údajů nebo firemní dokumentace. Důsledky podobných incidentů mohou zahrnovat právní sankce, značné finanční ztráty a nevratné poškození obchodní reputace společnosti. [5]

Neméně závažným negativním důsledkem je narušení integrity informací. V případě úspěšného obejití nebo zneužití kryptografických protokolů získá útočník možnost nejen získat přístup k důvěrným údajům, ale také je upravovat, falšovat zprávy nebo obcházet nastavené ochranné mechanismy. To zase otevírá cestu k dalšímu neoprávněnému přístupu do systému a eskalaci útoku na sousední komponenty infrastruktury. [11]

Nedokonalé zavedení kryptografických řešení může ohrozit dostupnost služeb. K tomu může dojít například v případě, že kryptografické nástroje způsobí systémové selhání nebo přetížení a naruší tak běžný provoz. Správné použití kryptografie proto přesahuje rámec pouhé technické úlohy a představuje základní požadavek pro zajištění bezpečnosti jakéhokoli informačního systému. [2]

2.5 Injection

Zranitelnosti typu „injection“ představují jeden z nejzávažnějších a historicky nejvýznamnějších bezpečnostních problémů webových aplikací. Vznikají v důsledku nesprávného zpracování uživatelského vstupu, což útočníkům umožňuje vložit a spustit libovolný kód nebo příkazy. Mezi potenciální důsledky patří kompromitace důvěrných dat, narušení normálního fungování aplikace a provádění škodlivých operací. [2]

Definice zranitelnosti

Termín „injekce“ popisuje scénář, při kterém jsou nedůvěryhodná vstupní data vložena do příkazu, který je následně interpretován systémem (například databází, operačním systémem

nebo skriptovacím prostředím). Mezi běžné příklady patří SQL injekce, injekce příkazů a mezistránkové skriptování (XSS). Společnou slabostí je neschopnost aplikace účinně oddělit data od instrukcí. Podstatou této zranitelnosti je nedostatečná validace a čištění vstupních dat. Pokud aplikace předává uživatelský vstup přímo interním komponentám bez adekvátních bezpečnostních kontrol, vytváří to podmínky pro zneužití. [2] [5]

Mechanismus vzniku zranitelnosti

Jednou z příčin vzniku takových zranitelností je přímé začlenění uživatelských dat do prováděných příkazů. Klasickým příkladem je vytváření dotazů SQL zřetězením řetězců, na rozdíl od použití parametrizovaných dotazů. Při tomto přístupu může databázový systém chybně interpretovat data zadaná uživatelem jako součást samotného příkazu. [2]

Podobné zranitelnosti se projevují i při interakci se systémovými nástroji. Neoprávněné spuštění příkazů na serveru je možné, pokud aplikace předává uživatelský vstup do příkazového řádku bez adekvátního filtrování. Základní příčina spočívá v nedostatečném oddělení dat a spustitelného kódu. [5][14]

Scénáře zneužití

Existuje několik běžných typů útoků založených na vkládání škodlivého kódu. Nejčastěji se vyskytuje SQL-injekce, která útočníkům umožňuje manipulovat s dotazy do databází. To může vést k úniku důvěrných údajů, jejich změně nebo obcházení autentizačních mechanismů. Existuje také příkazová injekce, která umožňuje provádět systémové příkazy na serveru. V případě, že aplikace disponuje zvýšenými oprávněními, může takový útok vést k úplnému převzetí systému. Do širší kategorie útoků typu „injekce“ patří také XSS. Při takovém útoku útočník vloží na webovou stránku škodlivý skript, který se poté spustí v prohlížeči jiného uživatele. To může vést k takovým důsledkům, jako je krádež uživatelské relace nebo změna zobrazeného obsahu. [1] [2] [14]

Dopady na informační systémy

Zranitelnosti typu injection představují vážnou hrozbu pro tři klíčové aspekty informační bezpečnosti: důvěrnost, integritu a dostupnost. Typickým výsledkem zneužití těchto zranitelností je získání neoprávněného přístupu k důvěrným údajům, včetně záznamů v databázích, přihlašovacích údajů uživatelů nebo interních informací aplikací. Takové incidenty vedou k narušení důvěrnosti a mohou pro organizaci znamenat značné právní a reputační náklady. [5]

Vážnou hrozbou je také možnost neoprávněné změny dat nebo funkčnosti aplikace. Útočník může získat možnost upravovat záznamy v databázi, mazat kriticky důležité informace, obcházet mechanismy ověřování nebo zkreslovat obsah zobrazovaný koncovým uživatelům. V kontextu útoků typu Command Injection existuje riziko provedení libovolných

systémových příkazů, což podstatně rozšiřuje rozsah poškození a může vést k úplnému kompromitování serverové infrastruktury. [14]

Útoky typu injection představují hrozbu nejen pro důvěrnost a integritu dat, ale také pro dostupnost služeb. Mechanismy útoku mohou zahrnovat přetížení databází, narušení logiky provádění aplikace nebo zásah do operačních procesů. Kritičnost této kategorie zranitelností je dána tím, že jediná chyba v parsování vstupních dat může sloužit jako vektor pro eskalaci oprávnění a následné destruktivní akce, které vedou ke kompromitaci celého informačního systému. Implementace opatření na ochranu před injekcemi je tedy nezbytnou podmínkou pro zajištění bezpečnosti webových aplikací. [2] [3]

2.6 Insecure Design

Insecure Design představují zranitelnosti, které vznikají již ve fázi vývoje architektury systému. Na rozdíl od chyb souvisejících s implementací nebo konfigurací mají svůj původ v základních architektonických řešeních, nikoli v jednotlivých technických nedostatcích. Odstranění těchto problémů proto často vyžaduje provedení významných změn v celkové struktuře systému. [2]

Definice zranitelnosti

Podle údajů organizace OWASP zahrnuje kategorie Insecure Design situace, které se vyznačují nedostatečným zavedením bezpečnostních opatření nebo absencí komplexní analýzy potenciálních hrozeb ve fázi návrhu. Typickými příklady jsou zranitelnost vůči automatizovaným útokům, nedostatečné rozdělení uživatelských oprávnění nebo absence mechanismů kontroly provádění kriticky důležitých operací. [2]

Takové problémy zpravidla nevznikají náhodně, ale jsou způsobeny tím, že bezpečnostní aspekty nebyly zohledněny již v raných fázích návrhu. To vede k vytváření systémů, které sice fungují, ale jsou náchylné k předvídatelným způsobům útoku. [15]

Mechanismus vzniku zranitelnosti

Hlavním problémem je, že bezpečnost není od samého počátku zakomponována do vývoje aplikace. Pozdní odhalení hrozeb, kriticky důležitých operací a potenciálních vektorů útoků vede k tomu, že při vytváření systému mohou být nezbytná bezpečnostní opatření zcela opomenuta. [2]

Nedostatečné využívání metod, jako je modelování hrozeb, navrhování bezpečné architektury a princip nejnižších oprávnění, představuje významnou zranitelnost. Pokud nejsou kriticky důležité funkce aplikace odděleny od méně důležitých komponent nebo pokud není přístup k nim regulován na základě rolí a kontextu, otevírá se tím prostor pro zlovolné aktivity. [15] [16]

Scénáře zneužití

Často dochází k situacím, kdy útočník zneužije zranitelnosti ve funkcích, které nepočítají s odpovídající autentizací nebo autorizací. To mu umožňuje provádět neoprávněné úkony s důvěrnými informacemi, včetně jejich úpravy nebo získání přístupu k datům, která přesahují rámec jeho oprávnění. [2]

Aplikace, které nejsou vybaveny mechanismy ochrany před automatizovanými hrozbami, jako je omezení počtu požadavků, monitorování neobvyklé aktivity nebo jiná protiopatření, představují zranitelný cíl. Jsou snadno napadnutelné útoky založenými na hádání hesel nebo hromadném získávání informací. [1]

Je třeba poznamenat, že chyby v návrhu často souvisejí s dalšími zranitelnostmi. Pokud se tak například vyskytne současně nedostatečná autorizace nebo zranitelnosti typu „injekce“, může se potenciální škoda způsobená útokem mnohonásobně zvýšit. [3]

Dopady na informační systémy

Insecure Design představují pro informační systémy závažnou hrozbu, protože se nejedná o ojedinělé chyby, ale spíše o zásadní zranitelnosti, které jsou vlastní samotné architektuře aplikace. Takové problémy mají zpravidla komplexní charakter a dotýkají se mnoha komponent systému. Jejich náprava často vyžaduje značné úsilí a zdroje, protože může vést k přepracování logiky fungování aplikace, autorizačních politik nebo mechanismů zpracování kriticky důležitých transakcí. [15] [16]

Z hlediska kybernetické bezpečnosti představuje tato zranitelnost vážnou hrozbu, která může ohrozit důvěrnost, integritu a dostupnost informací. Neoprávněný přístup může útočníkovi umožnit získat důvěrné informace, provést v nich změny nebo zneužít zranitelné komponenty systému k vlastním účelům. Při použití automatizovaných metod útoku jsou možné rozsáhlé incidenty, jako je hromadný sběr dat nebo odmítnutí služby. [1] [2]

Význam této kategorie zranitelností podtrhuje skutečnost, že chyby ve fázi návrhu jsou často příčinou vzniku nebo zneužití dalších slabých míst. Pokud v architektuře systému chybí náležitá bezpečnostní opatření, může jedna nedokonalost mnohonásobně zhoršit dopad jiné. Z toho vyplývá, že správné navrhování bezpečnosti je základním předpokladem pro zajištění odolnosti moderních informačních systémů vůči kybernetickým hrozbám. [3]

2.7 Authentication Failures

Authentication Failures zahrnují zranitelnosti vznikající v důsledku nedostatečné validace identity uživatele nebo slabé kontroly procesu ověřování. Takové nedostatky mohou otevřít dveře k neoprávněnému vniknutí do uživatelských účtů a administrativních částí systému. Tyto problémy často doprovázejí další zranitelnosti, jako je slabá správa relací nebo nedostatečná ochrana důvěrných údajů uživatelů. [2][5]

Definice zranitelnosti

Kategorie Authentication Failures zahrnuje případy, kdy jsou mechanismy ověřování identity uživatele navrženy nebo implementovány s nedostatky. Mezi běžné zranitelnosti patří používání jednoduchých hesel, absence opatření proti hrubé síle, slabá ochrana uživatelských relací a nedostatečná podpora vícefaktorového ověřování. [2]

Tato zranitelnost se netýká pouze procesu ověřování, ale také následného udržování autorizovaného stavu uživatele. Nedostatečné zabezpečení při správě identifikace a relací v aplikaci otevírá možnost kompromitace bez použití sofistikovaných technických metod. [14]

Mechanismus vzniku zranitelnosti

Uvedené zranitelnosti jsou často důsledkem ignorování základních bezpečnostních zásad. Častým problémem je nedostatečná ochrana proti útokům metodou hrubé síly, což útočníkovi umožňuje automatizované testování mnoha kombinací přihlašovacích údajů. [2]

Nedostatečná správa relací je rovněž častou příčinou zranitelností. Používání předvídatelných nebo dlouhodobě platných identifikátorů relací vytváří podmínky pro jejich zneužití. Kromě toho představují významná rizika slabé mechanismy ochrany hesel, jako je použití neúčinných hashovacích algoritmů, a absence vícefaktorové autentizace v kriticky důležitých systémech. [1] [5] [17]

Scénáře zneužití

Častým způsobem útoku je metoda brute force, při které útočník využívá automatizované nástroje k systematickému testování velkého množství kombinací uživatelských jmen a hesel. Absence mechanismů omezujících počet pokusů o přihlášení nebo analyzujících neobvyklou aktivitu může přispět k úspěšnému zneužití uživatelských účtů. [2]

Dalším častým typem útoku je credential stuffing, při kterém útočníci využívají odcizené přihlašovací údaje k získání přístupu k dalším službám. Účinnost této metody se výrazně zvyšuje, pokud mají uživatelé ve zvyku používat stejná hesla pro různé platformy. [14]

Jednou z významných hrozeb je kompromitace relace, k níž dochází například prostřednictvím odcizení nebo zachycení identifikátoru relace. To umožňuje útočníkovi úspěšně se vydávat za oprávněného uživatele, aniž by disponoval jeho přihlašovacími údaji. [1]

Dopady na informační systémy

Authentication Failures představují vážné bezpečnostní riziko pro informační systémy, které ohrožuje jejich důvěrnost, integritu a dostupnost. Nejzávažnějším důsledkem je možnost neoprávněného vniknutí do uživatelských a administrátorských účtů. To útočníkům otevírá cestu ke čtení, úpravám nebo mazání cenných dat, stejně jako k provádění akcí pod záminkou

legitimních uživatelů. V nejhorším případě může prolomení účtu s administrátorskými právy vést k úplnému převzetí kontroly nad aplikací nebo její správní částí. [5]

Vážným problémem je také zneužití identifikačních údajů uživatelů. Pokud útočník získá kontrolu nad aktivní relací nebo přihlašovacími údaji, může jednat jménem legitimního uživatele. To mu umožňuje provádět akce, které systém interpretuje jako legitimní, což zase zvyšuje pravděpodobnost neoprávněné změny informací, provádění podvodných transakcí a komplikuje proces odhalování bezpečnostních incidentů. [17]

Authentication Failures se často stává výchozím bodem pro další destruktivní činnosti. Po získání neoprávněného přístupu má útočník možnost provádět průzkum systému, odhalovat další zranitelná místa a postupně rozšiřovat svá oprávnění. Tento problém tedy přesahuje rámec pouhého porušení přihlašovacího postupu a představuje značné riziko pro integritu a bezpečnost celé informační infrastruktury. [2]

2.8 Software or Data Integrity Failures

Tato kategorie zahrnuje situace, kdy panuje nejistota ohledně pravosti softwaru nebo dat zpracovávaných systémem. V kontextu moderních aplikací, které se spoléhají na automatické aktualizace, externí zdroje a sdílené knihovny, se zajištění integrity stává kriticky důležitým. Neschopnost ověřit, že v softwaru nebo datech nedošlo k neoprávněným úpravám, představuje riziko vniknutí škodlivého kódu nebo zkreslení informací. [2] [15]

Definice zranitelnosti

Ohrožení integrity softwaru a dat vzniká v případě, že nejsou zavedena ochranná opatření, která by zabránila neoprávněným změnám v kódu, aktualizacích nebo samotných datech. Mezi běžné zranitelnosti patří absence digitálních podpisů, kontrolních součtů, nedostatečná ověřování zdrojů informací nebo nechráněný přístup ke konfiguračním souborům. [2]

Tento problém přesahuje rámec samotné funkčnosti aplikace a týká se celého jejího životního cyklu, včetně vývoje, distribuce a následného provozu. Absence mechanismů kontroly integrity v organizaci představuje riziko nepozorovaných neoprávněných úprav. [1]

Mechanismus vzniku zranitelnosti

Tyto zranitelnosti mohou nastat zejména v situacích, kdy se systém spoléhá na software nebo data, aniž by provedl jejich kryptografickou nebo procesní verifikaci. Typickým příkladem takového rizika je stahování aktualizací z nedůvěryhodných zdrojů nebo používání komponent bez ověření jejich pravosti. [2]

Nedostatečná úroveň zabezpečení procesu nepřetržité integrace/nepřetržitého nasazení (CI/CD) představuje závažnou zranitelnost, která útočníkům umožňuje vkládat škodlivý software přímo do zkompilovaných aplikací. Kromě toho představuje značné riziko také nedostatečná ochrana zpracovávaných dat bez mechanismů kontroly neoprávněných úprav. [7][15]

Scénáře zneužití

Jeden z možných scénářů předpokládá zneužití mechanismu aktualizace softwaru. Útočník může vytvořit a šířit škodlivý balíček aktualizací, který zamaskuje jako oficiální software. Absence mechanismů pro ověření integrity dané aktualizace může vyvolat rozsáhlé napadení zařízení. [2]

Existuje také riziko neoprávněných změn v datech nebo konfiguračních souborech. Pokud nejsou k dispozici mechanismy pro kontrolu integrity, mohou takové úpravy zůstat bez povšimnutí a být zneužity k narušení například finančních výkazů nebo fungování systému. [1]

Existuje značné riziko spojené s kompromitací repozitářů nebo externích knihoven používaných v projektu. Vniknutí škodlivého kódu prostřednictvím těchto zdánlivě důvěryhodných kanálů představuje vážnou hrozbu pro bezpečnost systému. [7]

Dopady na informační systémy

Porušení integrity softwaru nebo dat může vést k závažným problémům s důvěrností, integritou a dostupností informačních systémů. Pokud se útočníkům podaří zasáhnout do procesů aktualizace, instalace balíčků nebo kompilace, získají možnost vložit škodlivý kód přímo do legitimních aplikací. Tento kompromitovaný kód se pak může šířit do dalších systémů, čímž se zvyšuje pravděpodobnost vzniku rozsáhlých incidentů, které postihnou mnoho organizací nebo segmentů jedné infrastruktury. [2]

Neméně závažným dopadem je zhoršení kvality a spolehlivosti dat. Neoprávněné nebo nepozorované změny v datech či konfiguračních souborech mohou způsobit nesprávné fungování softwaru, vést k přijetí chybných manažerských rozhodnutí, vyvolat poruchy ve finančních a provozních cyklech a také podkopat důvěru ve získané systémové výsledky. [15]

Nelze podceňovat dlouhodobý dopad na důvěru jak jednotlivých uživatelů, tak celých organizací. Neschopnost systému zajistit pravost a neměnnost softwaru a dat podkopává jeho základní spolehlivost. Zajištění integrity dat a programů je proto základním kamenem bezpečného fungování moderních informačních systémů. [1]

2.9 Security Logging and Alerting Failures

Klíčovými prvky zajištění bezpečnosti informačních systémů jsou správná dokumentace událostí a včasné informování o bezpečnostních incidentech. Absence záznamu nebo nemožnost upozornění na útok dává útočníkovi možnost dlouhodobé a nepozorované přítomnosti v systému. Proto se tato část nezaměřuje na samotný fakt provedení útoku, ale na nedostatky v mechanismech detekce a reakce. [2]

Definice zranitelnosti

Security Logging and Alerting Failures zahrnuje případy, kdy mechanismus auditu bezpečnosti nefunguje správně. To se může projevit nedostatečným objemem shromažďovaných údajů o událostech, neúplností informací v protokolech, absencí jednotného systému správy protokolů nebo neschopností systému včas informovat o podezřelé aktivitě. [2]

Hlavní potíž spočívá v nedostatečné informovanosti o bezpečnostních incidentech. Chybějící ucelený přehled o dění v systému brání včasnému odhalení a adekvátnímu vyhodnocení hrozeb. [1]

Mechanismus vzniku zranitelnosti

Hlavní příčinou vzniku těchto zranitelností je nedostatečný objem zaznamenávaných událostí nebo ukládání protokolů v nečitelném formátu. Typickým příkladem je opomenutí zaznamenávat neúspěšné pokusy o přihlášení, změny přístupových práv nebo podezřelé operace. [2]

Nedostatečná centralizace a analýza protokolů představuje významnou bezpečnostní mezeru. Lokální ukládání a absence korelace událostí ztěžují odhalování komplexních útoků. Navíc i v případě, že protokoly existují, systém nemusí generovat včasná upozornění na podezřelou aktivitu, což tento problém ještě zhoršuje. [15] [18]

Scénáře zneužití

Existuje pravděpodobnost, že opakované neúspěšné pokusy o přihlášení nebo jiné podezřelé aktivity zůstanou bez povšimnutí a nebudou analyzovány. To vytváří příznivé podmínky pro útočníka a umožňuje mu pokračovat v útocích, aniž by byl včas odhalen. [2]

Alternativní způsob útoku představuje sérii jednotlivých, nenápadných operací. Každá jednotlivá akce nemusí vzbudit podezření. Pokud však chybí mechanismus pro korelaci událostí, který by umožnil vytvořit ucelený obraz o hrozbě, může se útočník úspěšně vyhnout odhalení. [1]

Pokud má útočník dostatečná oprávnění, je schopen upravit nebo zničit auditní protokoly, pokud se jejich ochranné mechanismy ukáží jako nedostatečné. To zase podstatně zkomplikuje nebo znemožní následné vyšetřování incidentu. [18]

Dopady na informační systémy

Nedostatky v bezpečnostním systému vedou k tomu, že útočníci mají možnost zůstat při své činnosti déle nepozorováni. Absence řádného monitorování a analýzy incidentů zbavuje organizaci možnosti operativně reagovat a lokalizovat hrozbu. To zase zvyšuje riziko vzniku škod a přispívá k postupnému rozšiřování přítomnosti útočníka v infrastruktuře. [3]

Nedostatečné vedení protokolů událostí vede k podstatnému snížení rychlosti reakce na incidenty. Absence podrobných záznamů ztěžuje přesné určení času zahájení útoku, identifikaci postižených komponent systému a odhad skutečného rozsahu incidentu. To zase komplikuje technickou analýzu, zpomaluje proces obnovy funkčnosti a zvyšuje celkové náklady spojené s likvidací následků incidentu. [15]

Nedostatečná dokumentace událostí a absence včasných oznámení představují slabá místa v dlouhodobé bezpečnosti organizace. Nemožnost provést zpětnou analýzu incidentů ztěžuje vypracování účinných nápravných opatření a optimalizaci stávajících bezpečnostních systémů. Nedostatky v těchto oblastech proto ohrožují nejen provozní stabilitu systému, ale i jeho schopnost čelit budoucím hrozbám. [2]

2.10 Mishandling of Exceptional Conditions

Mishandling of Exceptional Conditions představuje klíčový aspekt zajištění bezpečnosti webových aplikací. Neschopnost systému adekvátně zpracovat výjimky a nepředvídané scénáře může vést k ohrožení důvěrných údajů, narušení fungování systému nebo vzniku nových vektorů útoků. Proto se tato část zaměřuje nejen na povahu chyb, ale také na mechanismy jejich řešení v rámci systému. [2] [15]

Definice zranitelnosti

Mishandling of Exceptional Conditions zahrnuje případy, kdy software reaguje nevhodným způsobem na chyby, výjimky nebo nepředvídané provozní situace. Charakteristickými znaky takového chování jsou příliš podrobné chybové zprávy, narušení integrity stavu aplikace po selhání a také pokračování v provádění procesů navzdory výskytu kritických chyb. [2]

V kontextu bezpečnosti jde o víc než jen o technickou stabilitu; týká se to zranitelností, které mohou vést k úniku důvěrných údajů nebo posloužit jako odrazový můstek pro další destruktivní akce. [19]

Mechanismus vzniku zranitelnosti

Uvedené zranitelnosti jsou často způsobeny tím, že se vývojový proces soustředí výhradně na běžné provozní režimy a opomíjí přitom analýzu chování systému v případě výskytu chyb. Nedostatečné zpracování výjimečných situací může mít za následek nepředvídatelné důsledky, únik důvěrných údajů nebo narušení integrity systému. [2]

Nedostatečná pozornost věnovaná testování hraničních a chybových scénářů představuje závažný nedostatek. Ignorování situací, jako je přerušení připojení, nesprávné zadání dat nebo selhání transakce, může vést k tomu, že kritické zranitelnosti zůstanou neodhaleny až do okamžiku uvedení systému do provozního provozu. [15] [20]

Scénáře zneužití

Existuje riziko úniku důvěrných údajů prostřednictvím chybových hlášení. Pokud aplikace odhaluje technické údaje, jako jsou konfigurační parametry, výpisy zásobníku nebo informace o použitých technologiích, může to útočníkům poskytnout cenné informace pro plánování dalších útoků. [2]

Další scénář předpokládá, že útočník může využít okamžik, kdy se aplikace po chybě nachází v nekonzistentním stavu. Pokud systém není schopen se po selhání sám vrátit do bezpečného stavu, dává to hackerovi příležitost k manipulaci s daty nebo k obcházení ochranných mechanismů. [19]

Pokračování v provozu aplikace po zjištění chyby, aniž by byl proces řádně ukončen nebo byly zablokovány nesprávné operace, představuje značné riziko. Takové chování může vyvolat neoprávněný přístup nebo vést k dalšímu narušení integrity systému. [15]

Dopady na informační systémy

Mishandling of Exceptional Conditions může vést k ohrožení důvěrnosti, integrity a dostupnosti systému. Zobrazené chybové zprávy mohou obsahovat citlivé technické podrobnosti, které by útočníci mohli využít k dalším útokům. [2]

V důsledku toho může dojít k nesynchronizovanému stavu aplikace, což s sebou nese riziko nesprávného uložení dat, neúplného provedení operací nebo obejití nastavených kontrol. Taková situace zpochybňuje integritu a spolehlivost fungování systému. [19][20]

Nezpracované výjimky představují značné riziko pro stabilitu aplikací a dostupnost služeb, zejména v kriticky důležitých systémech, kde je vyžadován nepřetržitý provoz. Navíc mohou zhoršovat negativní dopad jiných zjištěných zranitelností, což je činí obzvláště nebezpečnými. [15][19]

2.11 Shrnutí kapitoly

Druhá kapitola je věnována systematickému rozboru zranitelností webových aplikací podle aktuálního seznamu OWASP Top 10 2025. Analýza každé kategorie zahrnovala zkoumání jejích charakteristik, mechanismů vzniku a běžných scénářů zneužití, s důrazem na jejich relevanci pro moderní informační systémy. Tento strukturovaný přístup umožnil vytvořit komplexní představu o nejkritičtějších bezpečnostních hrozbách, kterým čelí moderní webové aplikace v praxi.

Provedená analýza naznačuje, že většina zjištěných zranitelností není důsledkem ojedinělých chyb, ale spíše odrazem zásadních a opakujících se problémů v návrhu, implementaci nebo konfiguraci systémů. Tyto nedostatky, které se projevují v různých technologiích a typech aplikací, potvrzují jejich univerzální a systémový charakter. Vzhledem k tomu je zřejmé, že bezpečnost webových aplikací nevyžaduje izolovaná opatření, ale integrovaný přístup, který pokrývá celý životní cyklus vývoje softwaru.

Bylo zjištěno, že v reálném prostředí různé kategorie zranitelností často působí synergicky. To znamená, že útoky se zpravidla neomezují na zneužití jediné slabiny, ale představují kombinaci několika zranitelností, což vede ke zvýšení jejich účinnosti a celkového dopadu. Například zranitelnost typu „injekce“ může být využita k získání počátečního přístupu k databázi a následná nedostatečná autorizace umožní útočníkovi zvýšit svá oprávnění nebo manipulovat s daty. Tato skutečnost jasně demonstruje nutnost vnímat bezpečnost jako propojený ekosystém, nikoli jako soubor izolovaných incidentů.

Zásadní význam mají zranitelnosti, které vznikají již ve fázi návrhu nebo jsou důsledkem nesprávné konfigurace systémů. Odstraňování těchto nedostatků je často velmi obtížné a jejich potenciální důsledky mohou mít dlouhodobý charakter. Zároveň je zřejmé, že čistě technická bezpečnostní opatření nemohou být plně účinná bez integrace s odpovídajícími organizačními procesy, jako jsou systematické testování bezpečnosti, komplexní řízení rizik a pravidelné audity konfigurací.

Analýza získaných výsledků vede k závěru, že zranitelnosti uvedené v žebříčku OWASP Top 10 nejsou pouhými teoretickými koncepty, ale reálnými, aktivně zneužívanými hrozbami, které mají přímý dopad na funkčnost informačních systémů. Proto je hluboké porozumění těmto zranitelnostem kriticky důležité pro zajištění spolehlivé ochrany moderních aplikací a je nedílnou součástí praktického testování bezpečnosti.

V této souvislosti bude následující kapitola věnována penetračnímu testování webových aplikací. Jedná se o empirický přístup zaměřený na odhalování a ověřování zranitelností v podmínkách, které se co nejvíce blíží reálným situacím. Hlavním úkolem je ukázat, jak mohou být teoreticky odhalené slabiny ve skutečnosti zneužity útočníky a jaký je jejich reálný dopad na informační systémy.

3 Penetrační testování webových aplikací

Ačkoli zranitelnosti webových aplikací, které byly popsány v předchozí kapitole na základě klasifikace OWASP Top 10 2025, tvoří koncepční základ pro pochopení potenciálních hrozeb, jejich účinné odhalování a minimalizace v praxi vyžadují komplexnější přístup. Nestačí pouze identifikovat tyto kategorie; je nutné empiricky ověřit jejich využitelnost a potenciální dopad na cílové informační systémy.

Vzhledem k výše uvedenému se tato kapitola zaměřuje na penetrační testování webových aplikací – metodu, která představuje strukturovaný proces odhalování a ověřování zranitelností v bezpečnostních systémech. Penetrační testování, které simuluje skutečné kybernetické útoky, umožňuje posoudit reakci systému v podmínkách, které se co nejvíce blíží aktuálním hrozbám. V důsledku toho nejen odhaluje technické detaily existujících nedostatků, ale také přispívá k hlubokému pochopení jejich potenciálního dopadu.

Tato kapitola začíná výkladem základních pojmů a metodických přístupů k provádění penetračního testování a podrobně popisuje postup jednotlivých fází. Následuje přehled klíčových nástrojů používaných v procesu auditu bezpečnosti webových aplikací. Tyto nástroje slouží k efektivnímu odhalování systémových zranitelností a hodnocení jejich zneužitelnosti v podmínkách kontrolovaného testování. V této části kapitoly se podrobněji zabýváme praktickým zkoumáním konkrétních zranitelností, které byly představeny dříve. Na rozdíl od jejich teoretického rozboru se zde soustředíme na podrobný popis reálných scénářů útoků, postupu při zneužití zranitelností a jejich přímého dopadu na informační systémy. Tento přístup umožňuje propojit teoretické koncepty s praktickou aplikací a ukázat, jak mohou být bezpečnostní zranitelnosti ve skutečnosti zneužity.

Provádí se hluboký výzkum vlivu podobných útoků na klíčové komponenty informačních systémů, jako jsou databáze, aplikační software a síťová infrastruktura. Hlavním cílem je prokázat, že lokální zranitelnosti nejsou samostatnými vadami, ale mohou vyvolat řetězové negativní účinky, které ovlivňují provozní činnost organizace jako celku. Tato kapitola tedy představuje klíčový spojovací článek, který propojuje teoretické základy výzkumu s nadcházející praktickou fází, zahrnující kontrolovanou realizaci scénářů zneužívání v laboratorních podmínkách.

3.1 Úvod do penetračního testování

Penetrační testování je metoda praktického ověřování bezpečnosti informačních systémů, která spočívá v simulaci reálných útoků z pohledu potenciálního útočníka. Tento přístup je účinný při odhalování zranitelností, které mohou být při statické analýze nebo při použití automatizovaných nástrojů přehlédnuty. V případě webových aplikací je jeho význam ještě větší, protože tyto aplikace jsou dostupné ve veřejné síti a často slouží jako první linie útoku pro útočníky. [14]

Na rozdíl od abstraktních teoretických konceptů v oblasti bezpečnosti poskytuje praktické penetrační testování empirická data, která lze ověřit. Tento přístup přesahuje rámec

pouhého odhalování zranitelností a zaměřuje se na posouzení jejich skutečné zneužitelnosti a potenciálního dopadu na integritu systému. Toto rozlišení je zásadní, protože ne každá odhalená slabina představuje významné riziko, zatímco některé, i když jsou relativně snadno zneužitelné, mohou mít katastrofální následky. [21]

Penetrační testy se obvykle provádějí podle určitých metodik, díky čemuž jsou lépe organizované a předvídatelné. Příkladem takových standardů jsou PTES a OSSTMM. Tyto metodiky uspořádávají proces testování tak, že jej rozdělují do několika fází, od počáteční analýzy až po závěrečné vyhodnocení, a poskytují spolehlivý základ pro provádění bezpečnostních kontrol. [22]

Z praktického hlediska je nesmírně důležité vnímat penetrační testování nikoli jako samostatnou událost, ale jako nedílnou součást komplexní strategie kybernetické bezpečnosti. Společnosti jej využívají k systematickému odhalování potenciálních zranitelností, ověřování funkčnosti svých ochranných opatření a neustálému zdokonalování celkového bezpečnostního systému. Podle organizace ENISA je pravidelné provádění těchto kontrol základním prvkem efektivního řízení kybernetických rizik v moderních informačních systémech.[1]

V případě webových aplikací se penetrační testování zaměřuje na oblasti, které byly v rámci bezpečnostní analýzy identifikovány jako kritické. Mezi tyto oblasti patří zejména místa zadávání dat, mechanismy ověřování a rozhraní pro komunikaci mezi klientskou a serverovou částí. Prozkoumání těchto oblastí umožňuje odhalit praktické projevy zranitelností a hlouběji pochopit jejich skutečný dopad na funkčnost systému.[22]

Je také třeba poznamenat, že účinnost penetračního testování není dána pouze arzenálem použitých softwarových nástrojů, ale také kvalifikací odborníka. Pro dosažení podstatných výsledků je klíčová schopnost testera identifikovat zranitelnosti a kombinovat různé metody útoků. Právě tento aspekt odlišuje penetrační testování od automatizovaného skenování zranitelností, které zpravidla není schopno pokrýt víceúrovňové a složité scénáře hrozeb.[21]

Penetrační testování tedy představuje klíčový spojovací článek, který zajišťuje přechod od teoretického uchopení potenciálních zranitelností k jejich empirickému ověření v podmínkách reálného provozu. V této souvislosti budou následující kapitoly věnovány podrobnému rozboru metodických přístupů a nástrojů používaných v rámci tohoto druhu testování, jakož i jejich praktické implementaci při zkoumání specifických zranitelností webových aplikací.

3.2 Metodika penetračního testování

Pentestování by nemělo být vnímáno jako jednorázový pokus odhalit co nejvíce chyb, ale jako řízený a dokumentovaný proces. Jeho cílem je zjistit, do jaké míry je zneužití zranitelností testovaného systému reálné a jaké mohou být jeho důsledky. Právě metodický přístup odlišuje profesionální penetrační testování od náhodných technických zkoušek. NIST ve své příručce o technickém testování bezpečnosti zdůrazňuje, že pro získání spolehlivých výsledků vhodných pro řízení rizik musí organizace k testování přistupovat systematicky:

plánovat, provádět a vyhodnocovat jej. Podobně i PTES považuje penetrační testování za proces skládající se z postupných fází, které dohromady pokrývají celý průběh testování – od počáteční dohody až po vypracování zprávy.[22][23]

Z metodologického hlediska je důležité si uvědomit, že neexistuje univerzální řešení, které by vyhovovalo všem situacím. Příručka OWASP pro testování webové bezpečnosti zaměřená na webové aplikace nabízí strukturu testování v oblastech, jako je sběr informací, kontrola konfigurace, autentizace, správa relací, validace vstupních dat a obchodní logika. PTES naopak představuje penetrační testování jako širší proces sestávající ze sedmi klíčových fází: od interakce před zahájením projektu až po vypracování zprávy, včetně průzkumu, modelování hrozeb, analýzy zranitelností a zneužití. OSSTMM se zase zaměřuje na měřitelnost, opakovatelnost a provozní bezpečnost testování. Proto je v praxi nejrozumnější neomezovat se na jednu metodiku, ale kombinovat je s ohledem na specifika testovaného systému. [2][22][24]

Zásadní význam má fáze přípravy na testování, nazývaná také „pre-engagement“. Tento aspekt, ačkoli je v technických specifikacích často opomíjen, má rozhodující vliv na kvalitu celého testovacího procesu. V této fázi je nutné přesně definovat následující parametry: rozsah testování, přípustné metodiky, komunikační protokoly, časový rámec, pravidla pro práci s důvěrnými údaji a také postup při zjištění kritických zranitelností. Standard PTES vyzdvihuje tuto fázi jako samostatnou část a považuje ji za nezbytnou podmínku pro úspěšné provedení testování. Nesprávné určení rozsahu testování může vést k opomenutí klíčových prvků systému nebo k neoprávněnému zásahu do oblastí, které nemají být testovány. [22]

Dalším krokem po přípravě je sběr informací neboli průzkum (reconnaissance/intelligence gathering). Tato fáze nezahrnuje přímý útok, ale je zaměřena na získání co nejpřesnějšího přehledu o cílové aplikaci a její infrastruktuře. V kontextu webových aplikací to zahrnuje činnosti, jako je vyhledávání domén a subdomén, identifikace použitých technologií, frameworků, knihoven, způsobů autentizace, stejně jako studium struktury URL, parametrů, API, HTTP hlaviček a reakcí na chyby. OWASP WSTG vyzdvihuje sběr informací jako klíčovou oblast testování, protože kvalita následujících fází přímo závisí na tom, jak dobře tester rozumí povrchu aplikace a jejím vstupním bodům. PTES přitom zdůrazňuje, že sběr informací by měl být nejen technický, ale i analytický: cílem není nashromáždit co největší objem dat, ale získat informace, které umožní formulovat realistické hypotézy pro provedení útoků. [2][22]

Po dokončení rozvědné fáze začíná fáze, kterou lze nazvat modelováním hrozeb a hodnocením zranitelností. V této fázi se nashromážděné informace transformují do konkrétních scénářů pro provedení testování. Bezpečnostní specialista analyzuje, kde se mohou skrývat kritické body aplikace, které role nebo pracovní procesy představují největší zájem pro potenciálního útočníka a jaké kombinace slabých míst mohou mít vážné následky. PTES vyčleňuje modelování hrozeb do samostatné fáze, čímž zdůrazňuje, že penetrační testování by nemělo být pouhým seznamem technických kontrol, ale pečlivě promyšleným procesem založeným na perspektivě útočníka. OWASP WSTG tento přístup rozšiřuje a nabízí podrobnou klasifikaci konkrétních oblastí, které je vhodné ve webových aplikacích prozkoumat. NIST

zase rozlišuje mezi fázemi sběru informací, skenování, validace a dalších testovacích metod, čímž podporuje strukturovaný přístup k odhalování zranitelností. [2][22][23]

V kontextu hodnocení bezpečnosti je zásadně důležité rozlišovat mezi automatizovaným skenováním a komplexní analýzou zranitelností. Automatizované nástroje vykazují vysokou účinnost při identifikaci známých a typických problémů, jako jsou zastaralé softwarové komponenty, otevřené síťové služby nebo standardní chyby v konfiguraci. Jejich funkčnost je však omezená, pokud jde o hodnocení obchodní logiky, odhalování složitých řetězců zneužití zranitelností a předpovídání skutečných důsledků úspěšného útoku. Doporučení NIST přímo poukazují na nutnost použití řady testovacích metod a zdůrazňují, že samotné skenování nepokrývá celé spektrum potenciálních problémů. Podobný přístup prosazuje i OWASP Testing Framework, který považuje testování webových aplikací za synergii manuálních a automatizovaných technik. Automatický skener tedy může pouze vyslovit hypotézu o existenci zranitelnosti, zatímco konečné rozhodnutí o její aktuálnosti a kritičnosti přijímá kvalifikovaný testovací specialista.[2][23]

Následuje fáze exploitace, která představuje praktické ověření skutečné možnosti zneužití odhalené zranitelnosti. PTES interpretuje exploataci jako součást testování zaměřenou na získání přístupu k systému nebo zdroji obcházením bezpečnostních opatření. Právě v této fázi je nejzřetelněji vidět rozdíl mezi teoretickou znalostí zranitelnosti a její skutečnou použitelností v konkrétním prostředí. U webových systémů to může zahrnovat kontrolu na zranitelnosti, jako jsou SQL injekce, injekce příkazů, porušení kontroly přístupu, chyby autentizace, nesprávné konfigurace nebo problémy s obchodní logikou. Je důležité zdůraznit, že cílem není způsobit systému škodu, ale shromáždit dostatečné důkazy o potenciálním zneužití, při přísném dodržování autorizovaných a bezpečných hranic testování.[22]

V kontextu bezpečnostního hodnocení není pouhé zneužití zranitelnosti vždy dostačující. Metodické přístupy, zejména PTES, proto zahrnují fázi následného zneužití, jejímž cílem je analýza důsledků získaného přístupu. V rámci této fáze tester nejen ověřuje skutečnost kompromitace systému, ale také zkoumá potenciál dalších akcí: eskalaci oprávnění, exfiltraci dat, laterální pohyb nebo narušení integrity a dostupnosti systému. Tento aspekt je kriticky důležitý pro vytvoření adekvátního posouzení rizika, protože zranitelnost, která se jeví jako izolovaná, může v kombinaci s dalšími faktory vést k mnohem závažnějším důsledkům. To podtrhuje praktickou hodnotu penetračního testování ve srovnání s čistě deklarativním popisem zranitelností. [22]

Technická stránka penetračního testování je pouze částí celého procesu. Nelze podceňovat význam reportingu, protože právě ten umožňuje proměnit výsledky penetračního testu v účinná opatření k posílení bezpečnosti. Podle PTES by měl report zahrnovat rozsah testování, shromážděné údaje, podrobný popis postupu útoku, možné důsledky a návrhy na nápravu. NIST rovněž trvá na analýze zjištěných problémů a vypracování řešení. Bez ohledu na to, zda je penetrační testování posuzováno z akademického nebo praktického hlediska, má tento aspekt prvořadý význam. Cílem penetračního testu totiž není pouze najít „díry“, ale popsat je tak, aby bylo možné je uzavřít a aby organizace plně pochopila s nimi spojená rizika.[22][23]

Analýza, provedená dříve, ukázala, že metodika testování na proniknutí nepředstavuje striktně lineární schéma, které by se dalo uplatnit výhradně mechanicky. V reálné praxi dochází k vzájemnému prolínání jednotlivých fází a také k opakovanému návratu testera k předchozím fázím s následnou korekcí hypotéz na základě nově získaných informací. Přesto je existence metodologického základu nezbytnou podmínkou, protože zajišťuje systematickost, transparentnost procesu a jeho adekvátní provázání s řízením rizik. Pro účely této diplomové práce má tento přístup zvláštní význam, protože tvoří metodologický základ jak pro následnou praktickou analýzu zranitelností, tak pro vývoj laboratorních úkolů založených na realistických scénářích útoků.

3.3 Nástroje penetračního testování

Při provádění penetračního testování se nelze obejít bez použití specializovaného softwaru, který optimalizuje průběh jednotlivých fází hodnocení zabezpečení. Tyto nástroje však nejsou samy o sobě dostačující. Jejich hodnota se projeví pouze při komplexním využití, kdy každý z nich plní svou specifickou funkci v rámci celkové testovací metodiky. [22]

V kontextu moderních webových aplikací, které se vyznačují vysokou mírou složitosti a hlubokou integrací komponent, má kompetentní výběr a efektivní využití nástrojů zásadní význam pro zajištění kvality testování. Různé nástroje jsou určeny k podpoře konkrétních fází testovacího procesu, od počátečního průzkumu a sběru dat až po analýzu síťových interakcí, identifikaci zranitelností, jejich zneužití a následné vyhodnocení dopadů.

Z hlediska praktické použitelnosti se jeví jako rozumné provést klasifikaci testovacích nástrojů na základě jejich funkční role v rámci celého testovacího cyklu. Tento přístup přispívá k lepšímu pochopení vzájemných vztahů mezi nástroji a k jejich efektivnímu využití při řešení konkrétních scénářů zajištění bezpečnosti. V této práci jsou jednotlivé nástroje analyzovány nejen z teoretického hlediska, ale také v kontextu jejich empirického použití při vývoji a provádění laboratorních cvičení zaměřených na odhalování zranitelností odpovídajících OWASP Top 10.

3.3.1 Kali Linux

Kali Linux je linuxová distribuce vyvinutá speciálně pro účely penetračního testování a bezpečnostních auditů. Jde nad rámec standardní distribuce a představuje integrovanou platformu obsahující rozsáhlou sadu nástrojů pro odhalování a využívání zranitelností. Takovéto sjednocené prostředí zajišťuje připravenost k práci „ihned po vybalení“ a minimalizuje potřebu předběžné konfigurace. [25]

Kali Linux se vyznačuje vysokou mírou flexibility a rozšiřitelnosti. Uživatelské rozhraní umožňuje integraci doplňkových nástrojů, úpravu systémových parametrů a automatizaci testovacích fází pomocí skriptů. V praxi se Kali Linux často nasazuje ve virtualizovaných prostředích, což přispívá k izolaci testovacího prostředí a minimalizaci potenciálního negativního dopadu na hostitelský systém. Tento přístup nabývá zvláštního významu při práci s aplikacemi, které mají zranitelnosti, a při modelování reálných scénářů útoků. [25]

Z metodologického hlediska představuje Kali Linux klíčový operační systém, který zajišťuje komplexní provedení všech fází penetračního testování. To zahrnuje jak počáteční fáze, jako je průzkum a sběr dat, tak i následné fáze, včetně zneužití zranitelností a vyhodnocení výsledků. Sloučení specializovaných nástrojů do jednotného prostředí výrazně usnadňuje orientaci v procesu testování a přispívá ke zvýšení produktivity odborníků na kyberbezpečnost. [22]

V rámci této práce sloužil Kali Linux jako klíčový operační systém pro provádění laboratorních výzkumů. Využití této platformy ke spuštění všech testovacích nástrojů (například Burp Suite, sqlmap, Metasploit, Gobuster) umožnilo standardizovat proces a zajistit přehledné zobrazení výsledků analýzy jednotlivých zranitelností.

3.3.2 Burp Suite

Burp Suite je komplexní řešení pro audit bezpečnosti webových aplikací. Jeho klíčovou vlastností je funkce proxy serveru, která umožňuje zachycování a manipulaci s HTTP/HTTPS provozem v reálném čase. To umožňuje bezpečnostním specialistům podrobně analyzovat interakci mezi klientem a serverem a získat tak plnou kontrolu nad přenášenými daty za účelem odhalení potenciálních hrozeb. [26]

Z hlediska praktického použití má zásadní význam možnost ruční manipulace s HTTP požadavky. To umožňuje testerovi flexibilně nastavovat parametry, upravovat hlavičky a experimentovat s různými vstupními daty, přičemž sleduje chování aplikace. Tato metoda je obzvláště cenná při odhalování zranitelností, jako jsou injekce (Injection) nebo porušení kontroly přístupu (Broken Access Control), kde se automatizované prostředky často ukážou jako nedostatečné pro úplnou analýzu. [26]

Kromě ruční analýzy nabízí Burp Suite také výkonné automatizované funkce. Modul Intruder je určen k provádění fuzz testování a zkoumání variací vstupních dat, zatímco modul Scanner se zaměřuje na odhalování obecně známých zranitelností. Propojení ručních a automatizovaných metod zaručuje komplexní a důkladný audit bezpečnosti webových aplikací. [26]

V rámci této práce byl Burp Suite využíván k zachycování a úpravám HTTP požadavků při řešení praktických úkolů. Tento nástroj se ukázal jako nepostradatelný při zkoumání zranitelností, jako jsou SQL injekce (na příkladu Joomla) a XXE, a také při cílené modifikaci požadavků za účelem zneužití zranitelností v aplikacích založených na Spring a Struts. Použití Burp Suite umožnilo identifikovat zranitelné parametry a vyhodnotit reakci aplikace na nesprávná vstupní data.

3.3.3 Nmap

Nmap je specializovaný software vyvinutý pro provádění síťových auditů a průzkumu síťové infrastruktury. Jeho hlavním úkolem je detekce aktivních uzlů a služeb, které poskytují, v rámci sítě. V rámci penetračního testování hraje Nmap klíčovou roli v počáteční fázi sběru informací (průzkumu), kdy poskytuje základní přehled o cílovém systému a jeho síťových nastaveních. [27]

Použití různých metod skenování umožňuje identifikovat dostupné síťové porty, aktivní služby a jejich konkrétní verze. Tyto informace jsou zásadně důležité pro určení potenciálních vektorů útoku. Další možnosti, které nabízí například Nmap Scripting Engine, rozšiřují spektrum analýzy a umožňují odhalit známé zranitelnosti. Tento nástroj tak slouží specialistovi na testování jako základ pro pochopení síťové architektury a určení dalších směrů testování. [27]

Z metodologického hlediska představuje Nmap základní nástroj pro získání přehledu o topologii cílového systému a stanovení strategie pro další fáze testování. Data získaná během skenování slouží jako cenný zdroj informací pro testovacího specialistu a umožňují mu fundovaně vybírat specializované nástroje a metodiky pro provedení hloubkové analýzy a následného provozu. [27]

V rámci této práce byl Nmap využíván především ve fázi předběžné analýzy laboratorní infrastruktury. Skenování umožnilo ověřit dostupnost služeb, identifikovat aktivní porty a získat celkový přehled o spuštěných aplikacích, což posloužilo jako základ pro následné podrobné testování konkrétních zranitelností.

3.3.4 Wireshark

Wireshark je výkonný nástroj pro zachycování a podrobnou analýzu síťových dat na úrovni jednotlivých paketů. Jeho pasivní režim zaručuje, že analýza síťové aktivity probíhá bez jakýchkoli změn v přenášených datech, což zajišťuje vysokou přesnost a objektivitu při hodnocení chování aplikací a protokolů. [28]

Hlavní předností nástroje Wireshark je jeho schopnost podrobně analyzovat strukturu síťových paketů, rozpoznávat použité protokoly a vizualizovat dynamiku interakce mezi klientskými a serverovými uzly. Tato funkce nabývá zvláštního významu v kontextu auditu bezpečnosti přenosu dat, například pro ověření správnosti použití kryptografických mechanismů nebo pro odhalení úniků důvěrných informací v otevřené podobě. [28]

Z metodologického hlediska nachází Wireshark své hlavní uplatnění ve fázi analýzy síťové komunikace. Tento nástroj umožňuje hluboké pochopení výměny dat mezi různými prvky systému. Jeho role je obzvláště důležitá při diagnostice problémů souvisejících s kryptografickými protokoly, stejně jako při odhalování potenciálních úniků důvěrných informací v síťovém provozu. [28]

V rámci tohoto výzkumu byl nástroj Wireshark použit jako pomocný prostředek k podrobnému zkoumání síťové komunikace v kontrolovaném laboratorním prostředí. Tento nástroj byl použit k ověření protokolů přenosu dat mezi klientskými a serverovými komponenty, jakož i k odhalení případů neoprávněného přenosu důvěrných dat, což má zvláštní význam při posuzování zranitelností kategorie „Kryptografické selhání“.

3.3.5 Sqlmap

Sqlmap je specializované řešení pro odhalování a využívání zranitelností typu SQL injection ve webových aplikacích. Jeho klíčovou vlastností je úplná automatizace procesu, od vyhledávání zranitelných míst až po extrakci informací z databází. Nástroj nabízí širokou podporu

různých systémů pro správu databází a umožňuje provádět pokročilé akce, například získávat seznamy databází, tabulek nebo extrahovat konkrétní data. [29]

V praxi slouží nástroj sqlmap jako doplněk a rozšíření výsledků ručního bezpečnostního auditu. Testovací specialista nejprve identifikuje potenciální vektory útoků pomocí nástrojů, jako je Burp Suite, a poté použije sqlmap k automatizovanému prozkoumání a systematickému sběru dat. Tato metodika kombinuje silné stránky ruční analýzy a automatizovaných procesů, což vede ke zvýšení jak spolehlivosti, tak i rychlosti provádění testování. [29]

Sqlmap jde nad rámec pouhé verifikace zranitelností a umožňuje názorně demonstrovat jejich dopad tím, že získá neoprávněný přístup k důvěrným informacím obsaženým v databázi. To umožňuje specialistovi na testování bezpečnosti prozkoumat strukturu databáze, identifikovat uživatelské účty a extrahovat hesla, což představuje významné ohrožení integrity a důvěrnosti dat. [29]

V rámci této práce byl nástroj sqlmap použit k řešení praktického laboratorního úkolu zaměřeného na odhalování a zneužití zranitelností typu SQL-injekce v systému pro správu obsahu Joomla. Po ručním odhalení zranitelného parametru byl nástroj sqlmap použit k automatizovanému procesu výčtu dostupných databází, jejich tabulek a následnému získání důvěrných informací. Tento přístup názorně demonstroval potenciální rizika pro bezpečnost dat spojená s podobnými zranitelnostmi.

3.3.6 Metasploit Framework

Metasploit Framework je výkonná platforma vyvinutá pro vytváření a využívání exploitů v rámci penetračního testování. Umožňuje odborníkům na kyberbezpečnost simulovat reálné hrozby v izolovaném prostředí, čímž posuzují skutečný dopad odhalených zranitelností. Na rozdíl od řešení, která se zaměřují pouze na hledání slabých míst, klade Metasploit důraz na jejich aktivní zneužití. [30]

Tento nástroj je postaven na modulární struktuře, která se skládá z exploitů, užitečných nákladů a podpůrných komponent. Exploit je určen k využití konkrétní zranitelnosti, zatímco užitečný náklad představuje kód, který se spustí na cílovém systému po úspěšném prolomení zabezpečení. Mezi oblíbené užitečné zatížení patří Meterpreter, který zajišťuje rozšířené ovládání napadeného systému a umožňuje pracovat se soubory, provádět příkazy a shromažďovat systémové informace. [30]

Metasploit Framework je klíčovým nástrojem, zejména ve fázích zneužití zranitelností a vyhodnocení jejich dopadů. Po úspěšném použití exploitu poskytuje testerovi možnost určit úroveň získaných oprávnění, prozkoumat architekturu systému a odhalit další cesty pro útok. Tato metoda umožňuje získat úplné pochopení skutečné hrozby, která vychází z konkrétní zranitelnosti. [30]

V rámci tohoto výzkumu byl Metasploit Framework využíván především k zneužití zranitelností umožňujících vzdálené spuštění kódu (RCE). Konkrétně v rámci laboratorního testu simulujícího útok na Drupal byl úspěšně použit připravený exploit. To umožnilo navázat vzdálený přístup k cílovému systému a provést řadu diagnostických příkazů, čímž se

názorně demonstrovala potenciální rizika spojená s touto zranitelností pro bezpečnost aplikace.

3.3.7 Gobuster

Gobuster je výkonný nástroj pro bezpečnostní audity, který se specializuje na vyhledávání skrytých webových zdrojů. Využívá techniku hrubé síly a systematicky odesílá dotazy na cílový server s využitím rozsáhlých slovníků. To umožňuje odhalit potenciálně zranitelné nebo skryté adresáře, soubory a subdomény, které mohou být nedostupné prostřednictvím standardních navigačních cest. [31]

Z hlediska praktického využití je Gobuster cenným nástrojem ve fázi průzkumu, který umožňuje testerům odhalit skryté komponenty webových aplikací. Efektivní využití tohoto nástroje může přispět k odhalení administrativních panelů, záložních kopií dat, konfiguračních souborů nebo technické dokumentace, které mohou obsahovat důvěrné informace nebo otevírat cesty pro další útoky. [31]

Z metodologického hlediska je skenování klíčovou fází při odhalování možných cest proniknutí. Data shromážděná pomocí nástroje Gobuster pak slouží jako základ pro další kontroly, jako je analýza síťové komunikace v Burp Suite nebo testování na SQL injekce pomocí nástroje sqlmap. [31]

V rámci laboratorních prací byl k detekci skrytých adresářů použit nástroj Gobuster. To umožnilo zejména zjistit verzi použité aplikace a najít soubory s cennými informacemi, jako je například CHANGELOG. Získaná data se stala výchozím bodem pro další analýzu zranitelností a jejich následné zneužití.

3.3.8 Nikto

Nikto je specializované řešení pro provádění bezpečnostních auditů webových serverů. Jeho hlavním úkolem je identifikace typických zranitelností, nesprávných konfigurací a zastaralých verzí serverového softwaru. K dosažení tohoto cíle se nástroj opírá o rozsáhlou databázi obsahující tisíce testovacích scénářů, což zajišťuje rychlost a efektivitu při odhalování běžných bezpečnostních problémů. [32]

Z hlediska praktického využití nachází Nikto své hlavní uplatnění v počáteční fázi bezpečnostního auditu. V této fázi umožňuje odborníkovi získat základní přehled o stavu zabezpečení testovaného systému. Nástroj účinně odhaluje aspekty, jako jsou nesprávně nakonfigurované HTTP hlavičky, povolené HTTP metody, přístup k citlivým souborům, stejně jako známé zranitelnosti typické pro určité verze webových serverů. Shromážděná data slouží jako základ pro další, podrobnější zkoumání a provádění hloubkového testování. [32]

Z metodologického hlediska je třeba zdůraznit, že Nikto je především nástrojem pro předběžné posouzení. Data získaná během skenování mohou obsahovat falešné poplachy, což vyžaduje ověření všech zjištěných zranitelností prostřednictvím ručního auditu. Integrace automatizované analýzy s ruční kontrolou zaručuje zvýšení přesnosti a spolehlivosti výsledků testování. [32]

V rámci tohoto výzkumu byl nástroj Nikto použit jako pomocný prostředek k rychlému odhalení klíčových bezpečnostních zranitelností v kontrolovaném laboratorním prostředí. Tento nástroj umožnil získat obecný přehled o konfiguraci webových služeb a pomohl odhalit potenciální slabá místa, která byla následně podrobena hloubkové analýze s využitím dalších specializovaných nástrojů.

3.3.9 John the Ripper

John the Ripper je specializovaný software vyvinutý pro offline analýzu kryptografických hashů hesel. V rámci bezpečnostního auditu autentizačních systémů se používá k posouzení odolnosti uložených hesel vůči různým typům útoků. Nástroj využívá řadu kryptoanalytických metod, včetně slovníkových útoků a útoků metodou úplného prohledávání, s cílem dešifrovat původní hesla z jejich hashovaných podob. [33]

V praxi se John the Ripper využívá při analýze databází, v nichž jsou uloženy hashované identifikační údaje uživatelů. To umožňuje odborníkům na testování bezpečnosti posoudit spolehlivost používaných hashovacích algoritmů a zásad pro tvorbu hesel. Odhalení zranitelných nebo snadno prolomitelných hashů poukazuje na potenciální mezery v ochraně důvěrných informací a představuje značné riziko pro celkovou bezpečnost systému. [33]

Z metodologického hlediska tento nástroj umožňuje posoudit kryptografickou odolnost autentizačního systému a odhalit slabá místa v implementaci bezpečnostních mechanismů. Údaje získané v průběhu analýzy mohou zejména poukazovat na použití zastaralých hashových funkcí nebo na nedostatečnost doplňkových bezpečnostních opatření, jako je například solování. [33]

V rámci tohoto výzkumu byl k analýze laboratorního úkolu souvisejícího s kryptografickými zranitelnostmi (Cryptographic Failures) použit nástroj John the Ripper. Použití tohoto softwaru umožnilo názorně demonstrovat slabiny hashových funkcí, zejména MD5 bez použití „soli“, a empiricky potvrdit snadnost kompromitace hesel chráněných tímto způsobem.

3.3.10 Docker

Docker je způsob, jak zabalit programy spolu se vším, co potřebují k fungování (například potřebné soubory a nastavení), a spouštět je ve vlastních „krabičkách“ (kontejnery). Tyto krabičky nijak nezasahují do hlavního operačního systému počítače a program bude fungovat stejně dobře kdekoli, ať už na vašem počítači nebo na serveru. [34]

Z hlediska praktického využití představuje Docker cenný nástroj pro provádění penetračních testů. Jeho architektura zajišťuje bezpečné prostředí pro práci s potenciálně zranitelnými aplikacemi, čímž eliminuje riziko kompromitace produkčních systémů. Nízká spotřeba systémových zdrojů, vysoká rychlost inicializace a snadné obnovení do původního stavu činí kontejnery obzvláště žádanými v laboratorních podmínkách. Tyto vlastnosti jsou kriticky důležité pro opakované provádění testů a modelování různých vektorů útoků. [34]

Docker nabízí významnou výhodu v podobě možnosti orchestrace složitých, víceúložkových prostředí. Nástroje jako Docker Compose umožňují sjednotit do jednoho systému

prvky, jako jsou webové servery, databáze a podpůrné služby. To zajišťuje přesné ztvárnění architektury moderních webových aplikací, což je zásadně důležité pro analýzu vzájemných závislostí mezi komponenty a posouzení jejich vlivu na celkovou bezpečnost systému. [34]

Z metodologického hlediska funguje Docker jako nástroj pro sjednocení laboratorních prostředí, přičemž zaručuje reprodukovatelnost testovacích scénářů. To umožňuje testerům snadno nasazovat identická prostředí na různých zařízeních, čímž zajišťuje jednotnost výsledků při provádění analýzy. [34]

V této práci Docker sloužil jako základní platforma pro inicializaci a provoz zranitelných softwarových prostředí. Orchestrace jednotlivých laboratorních scénářů, převzatých z repozitáře Vulhub, byla realizována pomocí nástroje Docker Compose. Tato metodika umožnila dosáhnout vysoké rychlosti nasazení, zjednodušit správu a zaručit reprodukovatelnost výsledků při testování specifických zranitelností.

3.3.11 Vulhub

Vulhub je open-source iniciativa, jejímž cílem je vývoj a šíření předem nakonfigurovaných aplikací s bezpečnostními chybami. Tyto aplikace jsou určeny k provádění bezpečnostních testů a vzdělávacích aktivit. Každý scénář obsahuje specifickou, reálně existující zranitelnost, zapouzdřenou v izolovaném prostředí, což zajišťuje připravenost k okamžitému použití. Díky využití technologie Docker se nasazení těchto scénářů provádí s minimálním úsilím, což vylučuje nutnost pracného nastavování. [35]

Z hlediska praktického využití Vulhub výrazně optimalizuje proces nasazování testovacích prostředí. To umožňuje odborníkům na kyberbezpečnost vyhnout se časově náročné ruční instalaci a konfiguraci zranitelného softwaru a soustředit se tak na přímé zkoumání a využívání odhalených zranitelností. Každý představený scénář pečlivě simuluje realistické pracovní situace, což výrazně zvyšuje jeho pedagogickou účinnost. [35]

Z metodologického hlediska představuje Vulhub účinný zdroj, který napomáhá propojení teoretických konceptů s praktickými dovednostmi. Platforma umožňuje simulovat reálné scénáře útoků v bezpečném prostředí, což umožňuje zkoumat reakci aplikací na konkrétní zranitelnosti. Tato metoda je zásadní pro pochopení povahy kyberútoků a jejich vlivu na bezpečnost informačních systémů. [35]

Pro účely této práce posloužila platforma Vulhub jako základ pro vytvoření laboratorních scénářů. Konkrétní úkoly byly sestaveny tak, že byly vybrány a přizpůsobeny zranitelnosti uvedené na Vulhubu do podoby strukturovaných praktických cvičení. Tento přístup umožnil vytvořit jednotné a reprodukovatelné vzdělávací prostředí zaměřené na názornou demonstraci zranitelností odpovídajících seznamu OWASP Top 10.

3.4 Praktické aspekty penetračního testování webových aplikací

Provádění penetračního testování webových aplikací v reálném provozním prostředí se podstatně liší od abstraktních teoretických modelů a metodických rámců. Metodiky poskytují strukturovaný přístup a určují posloupnost fází, avšak praktická realizace je často podmíněna specifickými charakteristikami testovaného systému, časovými omezeními a mírou

dostupnosti informací. Od testera se proto vyžaduje schopnost flexibilně přizpůsobit svůj přístup a operativně reagovat na nepředvídané okolnosti. [22]

Klíčový rozdíl mezi teorií a reálným životem spočívá v tom, že v reálných aplikacích se zranitelnosti málokdy vyskytují izolovaně. K úspěšnému prolomení zabezpečení je často nutné zkombinovat několik slabých míst, z nichž každé samo o sobě není kritické. Tento proces, nazývaný „řetězec zranitelností“, umožňuje útočníkovi postupně rozšiřovat svá oprávnění a dosáhnout hlubšího proniknutí. Taková strategie se nedá plně automatizovat a vyžaduje od testera rozvinuté analytické myšlení. [2]

Klíčovým aspektem testovacího procesu je efektivní zpracování dat získaných z automatizovaných nástrojů. Zejména skenery zranitelností často vykazují značné množství potenciálních problémů, avšak ne všechny z nich mají praktický význam. Častý výskyt falešně pozitivních výsledků představuje vážný problém, protože mohou zkreslit skutečný obraz o zabezpečení systému. Je proto nesmírně důležité, aby tester disponoval dovednostmi potřebnými k ověření těchto výsledků a přesnému určení jejich relevance. [23]

V praxi se pravidelně setkáváme s řadou omezujících faktorů, které mohou výrazně ovlivnit proces testování. Mezi tyto faktory patří zejména neúplné pokrytí testovacích scénářů, nedostatek systémové dokumentace nebo organizační překážky. Takové okolnosti mohou zkomplikovat odhalování potenciálních zranitelností a vyžadují od testovacího specialisty schopnost efektivně pracovat v podmínkách informační nejistoty.

Kromě jiného je zásadně důležité důkladné pochopení kontextu aplikace. Nejasnosti ohledně její vnitřní logiky, funkčních možností a scénářů provozu výrazně ztěžují identifikaci skrytých zranitelností. Tato skutečnost je obzvláště relevantní pro chyby v obchodní logice, které při použití standardních analytických nástrojů obvykle zůstávají nepovšimnuty.[11]

Pro dosažení optimální účinnosti testování je zásadně důležité kombinovat ruční i automatizované postupy. Automatizované nástroje účinně odhalují již známé zranitelnosti, zatímco ruční zkoumání zajišťuje hluboké porozumění architektuře systému a umožňuje odhalit skryté vektory útoků. Tato kombinovaná strategie je v současné oblasti penetračního testování běžnou praxí. [22]

Nelze opomenout ani etické, ani právní aspekty provádění testování. Pentestování vyžaduje povinné získání informovaného souhlasu a přísné dodržování stanovených hranic. Porušení těchto zásad s sebou nese riziko právních sankcí a ztráty důvěry mezi specialistou na testování a testovanou organizací.

Empirické údaje z oblasti penetračního testování tedy potvrzují, že bezpečnostní analýza není výlučně technickou disciplínou. Naopak vyžaduje propojení technických kompetencí, analytického myšlení a schopnosti interpretovat získané výsledky v širším systémovém kontextu. Tento komplexní přístup je zásadní pro účinné odhalování a adekvátní hodnocení zranitelností v operačním prostředí.

3.5 Shrnutí kapitoly

Ve třetí kapitole byla hlavní pozornost věnována praktickému využití penetračního testování webových aplikací jako účinné metody odhalování zranitelností. Na rozdíl od předchozí kapitoly, kde byla pojednána teoretická klasifikace zranitelností podle OWASP Top 10 2025, se tato část zaměřila na jejich empirickou identifikaci a analýzu v podmínkách reálného provozu. Tímto způsobem se podařilo integrovat teoretické koncepty s jejich praktickou realizací v kontextu hodnocení bezpečnosti.

V této části byly podrobně popsány základní principy penetračního testování. Zvláštní pozornost byla věnována jeho systémové metodice, která zahrnuje klíčové fáze: od počátečního průzkumu a identifikace zranitelností až po jejich praktické zneužití a následnou komplexní analýzu dopadů. Byla zdůrazněna zásadní důležitost komplexního přístupu, který předpokládá integraci různých technik, protože účinnost tohoto procesu nelze dosáhnout výhradně pomocí automatizovaných prostředků.

V této kapitole byla rovněž provedena analýza konkrétních nástrojů pro penetrační testování, které napomáhají realizaci různých fází testovacího procesu. Bylo zjištěno, že každý nástroj disponuje jedinečnými funkcemi a jejich synergické využití je pro dosažení úspěšných výsledků testování zásadně důležité. Zároveň bylo zdůrazněno, že nástroje, jakkoli dokonalé, nemohou nahradit analytické schopnosti odborníka na testování.

Klíčovým závěrem této části bylo zjištění, že penetrační testování je nepostradatelným nástrojem pro ověření zabezpečení webových aplikací. Umožňuje identifikovat zranitelnosti, které by útočníci mohli využít v reálných podmínkách, a kvantitativně vyhodnotit jejich potenciální dopad na informační systémy. Současně byly zváženy praktické aspekty provádění penetračních testů, včetně omezení použitých nástrojů, potřeby odborné ruční analýzy a významu zohlednění specifik testovaného systému.

Tato kapitola tedy slouží jako výchozí bod pro další fázi práce, v níž navrhneme laboratorní cvičení. Jejich cílem je simulovat scénáře zneužití zranitelností webových aplikací, které se co nejvíce blíží reálným situacím.

4 Návrh a implementace laboratorního prostředí

Na základě teoretických principů popsaných v předchozích kapitolách se tato část studie zaměřuje na návrh a vytvoření laboratorního prostředí. Toto prostředí má sloužit jako platforma pro praktickou demonstraci specifických zranitelností webových aplikací. Klíčovým úkolem je vytvoření kontrolovaného prostředí, které umožní bezpečně reprodukovat realistické scénáře útoků a vyhodnotit jejich dopad na informační systémy.

V této kapitole je uveden popis vyvinutého prostředí a zdůvodnění volby použitých technologických řešení. Následuje podrobný popis procesu implementace, který zahrnuje konfiguraci jednotlivých prvků a integraci všech složek do jednotného řešení. Kapitolu uzavírá popis architektury prostředí a metodiky použité při testování.

Vytvořená platforma poskytuje základ pro přípravu praktických laboratorních cvičení zaměřených na odhalování a odstraňování zranitelností, které jsou seřazeny podle seznamu OWASP Top 10. Prioritou je zajištění bezpečnosti, reprodukovatelnosti výsledků a škálovatelnosti, což ji činí vhodnou nejen pro použití v rámci aktuálního projektu, ale i pro širší vzdělávací a výzkumné účely v oblasti kyberbezpečnosti.

4.1 Návrh laboratorního prostředí

S ohledem na předchozí části studie věnované analýze zranitelností webových aplikací a metodice penetračního testování vyvstala potřeba vytvořit laboratorní prostředí pro praktické ověření získaných znalostí. Vývoj tohoto prostředí byl proto vnímán nejen jako technická příprava na plnění konkrétních úkolů, ale také jako samostatná, kriticky důležitá součást, která má podstatný vliv na bezpečnost, reprodukovatelnost výsledků a pedagogickou účinnost celého systému.

Klíčovým požadavkem bylo vytvoření bezpečného prostředí pro simulaci útoků na webové aplikace, které by zcela vyloučilo jakýkoli dopad na reálné systémy. Bylo důležité, aby toto řešení bylo vhodné i pro vzdělávací účely, bylo názorné, snadno pochopitelné a technicky nenáročné. Zaměřil jsem se tedy na vývoj prostředí, které by zajišťovalo realistické chování zranitelných aplikací a zároveň zůstalo jednoduché na používání.

Při vývoji byly stanoveny zejména následující požadavky: izolace testovaného prostředí od hostitelského systému, možnost rychlého spuštění a opakovaného resetování jednotlivých scénářů, přehledná organizace laboratorních úkolů, nízké požadavky na hardwarové vybavení a dostatečná flexibilita pro úpravy či rozšíření jednotlivých zranitelností. Důležitým požadavkem byla také reprodukovatelnost, tedy schopnost opakovaně spouštět stejné laboratorní scénáře za stejných podmínek a získávat srovnatelné výsledky.

K splnění stanovených požadavků bylo zvoleno virtualizované prostředí s operačním systémem Kali Linux jako základním operačním systémem. Toto řešení umožňuje spolehlivě izolovat testovací infrastrukturu od hostitelského systému a zároveň zajišťuje přístup ke všem nástrojům nezbytným pro penetrační testování. Výhody virtualizace přesahují rámec

bezpečnosti a zjednodušují správu: systém lze snadno zálohovat, přesouvat a obnovovat, což je obzvláště cenné při práci s aplikacemi obsahujícími zranitelnosti.

Docker je jednou ze základních technologií použitých při návrhu prostředí. Jeho volba je dána schopností spouštět izolované, zranitelné aplikace ve formě samostatných kontejnerů. To zajišťuje vzájemnou nezávislost kontejnerů při centralizované správě. V důsledku toho lze každý laboratorní úkol realizovat jako samostatný scénář, což vylučuje nutnost vytvářet samostatné virtuální stroje pro každou zranitelnost. To vede k podstatnému snížení požadavků na diskový prostor, zrychlení procesu nasazení a zjednodušení správy.

Z pohledu architektury je toto prostředí realizováno jako lokální testovací platforma, která na jednom fyzickém nosiči sdružuje útočné uzly a služby s bezpečnostními zranitelnostmi. Kali Linux plní roli řídicí a výkonné vrstvy, z níž jsou iniciovány operace penetračního testování. Zranitelné aplikace jsou zapouzdřeny v kontejnerech Docker. Přístup k jednotlivým službám je zajištěn prostřednictvím lokální adresy 127.0.0.1 nebo interního IP adresování, což usnadňuje přístup k aplikacím a zároveň zachovává zapouzdření celého prostředí.

Toto řešení má řadu významných výhod. Především zajišťuje vysokou úroveň bezpečnosti: všechny prováděné útoky jsou izolovány, což zcela vylučuje ohrožení ostatních systémů. Další výhodou je modulární architektura, která umožňuje spouštět jednotlivé úlohy nezávisle na sobě. To výrazně zjednodušuje jejich využití ve výuce a otevírá možnosti pro další rozšiřování funkcí. Třetím klíčovým aspektem je reprodukovatelnost: kontejnery lze kdykoli zastavit, znovu spustit nebo vrátit do původního stavu. Z pedagogického hlediska je obzvláště cenné, že studenti mohou pracovat s realistickými scénáři útoků, přičemž se nacházejí v kontrolovaném prostředí, které je pro učitele přehledné a snadno ovladatelné.

Při navrhování jsem bral v úvahu specifické vlastnosti zranitelností, protože jsem si uvědomoval, že rozmanitost scénářů vyžaduje odpovídající architekturu. Zatímco některé laboratorní úkoly se zaměřují na jednu webovou aplikaci, jiné předpokládají interakci několika služeb, včetně webového serveru, databáze a podpůrných komponent. Použití Dockeru mi umožnilo efektivně vytvářet jak jednoduché, tak složité scénáře s více komponenty, aniž bych musel měnit základní principy fungování prostředí.

Organizace laboratorních úkolů tvořila významnou část projektu. Každý úkol byl umístěn do samostatného adresáře, který obsahoval všechny potřebné konfigurační soubory, specifikace Docker Compose a podpůrné materiály. Tato struktura usnadňuje efektivní orientaci a přehlednou správu rozsáhlé sady scénářů. Z hlediska dalšího využití ve vzdělávacím procesu zajišťuje tento přístup vysokou míru přizpůsobitelnosti a umožňuje snadno integrovat, vylučovat nebo upravovat jednotlivé úkoly bez dopadu na ostatní komponenty systému.

Při navrhování laboratorního prostředí jsem dbál na to, aby sloužilo nejen k demonstraci technických zranitelností, ale také jako cenný výukový nástroj. To znamenalo, že studenti měli mít možnost nejen provádět útoky, ale také pochopit jejich kontext, důsledky a metody ochrany. Proto jsem se rozhodl pro architekturu, která zajišťuje dostatečnou míru realismu, ale zároveň nevytváří zbytečné překážky pro studenty s omezenými předchozími zkušenostmi.

Vyvinuté laboratorní prostředí je výsledkem vyváženého přístupu, který zohledňuje realističnost scénářů, úroveň bezpečnosti, technickou proveditelnost a pedagogickou vhodnost. Umožňuje názornou demonstraci různých zranitelností webových aplikací, přičemž využívá jednotný systém správy a nasazení. To vytváří pevný základ pro další fáze výzkumu, kde bude toto prostředí použito k vytváření a provádění konkrétních laboratorních prací.

4.2 Implementace prostředí

Navržený koncept laboratorního prostředí vyžadoval praktickou realizaci. Cílem této fáze bylo vytvoření pracovní platformy založené na dané architektuře. Platforma měla umožňovat spouštění zranitelných aplikací, zkoumání jejich chování a vytváření laboratorních úkolů na jejich základě. Zavedení prostředí přesáhlo rámec pouhé instalace softwaru a zahrnovalo organizaci pracovního prostoru, nastavení síťového přístupu a standardizaci spouštění scénářů.

Jako základ pro naši laboratorní infrastrukturu byl zvolen virtuální stroj s operačním systémem Kali Linux. Tento výběr byl podmíněn jak dostupností specializovaných nástrojů pro analýzu bezpečnosti, tak i jeho vhodností pro vytváření izolovaných testovacích prostředí. Po inicializaci virtuálního stroje bylo provedeno testování kriticky důležitých komponent, včetně kontroly síťové konektivity, funkčnosti emulátoru terminálu a dostupnosti plánovaných nástrojů. Zvláštní pozornost byla věnována zajištění stability a reprodukovatelnosti konfigurace, což je nezbytnou podmínkou pro správné provedení všech laboratorních scénářů.

Základem pro realizaci těchto zranitelných scénářů byla technologie Docker. Ta funguje jako spouštěcí prostředí, ve kterém se umisťují a spouštějí jednotlivé aplikace. Klíčovou výhodou Dockeru oproti tradiční virtualizaci je možnost spouštět mnoho izolovaných aplikací na jednom systému, namísto vytváření samostatného virtuálního stroje pro každý úkol. To vede ke snížení nároků na výkon a úložný prostor, a také k jednodušší správě celého prostředí.

Pro úspěšné zavedení nestačilo pouze nainstalovat Docker, ale bylo také nutné naučit se efektivně spravovat kontejnery. To zahrnovalo ověření spolehlivosti operací s kontejnery (vytváření, spouštění, zastavování, mazání) a zajištění plynulé interakce mezi hostitelským systémem a kontejnery. Nástroj Docker Compose sehrál významnou roli, protože zjednodušil definici vícekomponentních prostředí pomocí jediného konfiguračního souboru. Tento přístup se ukázal jako obzvláště efektivní pro scénáře, kde musí webová aplikace a databáze spolupracovat.

Pro zajištění strukturovaného přístupu k plnění úkolů byl vyvinut specializovaný hierarchický katalog. V rámci této struktury byl každý laboratorní úkol umístěn do samostatného adresáře. Každý takový adresář obsahoval kompletní sadu artefaktů nezbytných pro inicializaci a provedení příslušného scénáře. Klíčovými komponenty byly konfigurační soubor docker-compose.yml a související soubory specifické pro konkrétní aplikaci nebo scénář zneužití zranitelnosti. Taková organizace umožnila výrazně zvýšit pohodlí při navigaci v prostředí a zaručila jednotný přístup ke všem laboratorním materiálům.

Všechny scénáře byly spouštěny podle jednotného standardu přímo ze svých adresářů. V praxi to znamenalo, že po otevření složky konkrétního úkolu mohli uživatelé spustit aplikaci pomocí Docker Compose a stejně snadno ji také zastavit. Tento přístup měl řadu klíčových výhod. Hlavním z nich byla sjednocení pracovního postupu pro studenty i učitele, díky čemuž byla práce s různými zranitelnostmi stejně jednoduchá. Kromě toho to výrazně usnadnilo údržbu prostředí, protože každý scénář mohl být konfigurován odděleně od ostatních.

V rámci implementace bylo provedeno nastavení síťového přístupu k aplikacím. Přístup k jednotlivým kontejnerům probíhal přes porty přesměrované na lokální rozhraní, zpravidla na adresu 127.0.0.1 s uvedením portu. Tato metoda byla zvolena záměrně, aby byl zajištěn pohodlný přístup jak z webového prohlížeče, tak z bezpečnostních nástrojů fungujících v prostředí Kali Linux. Současně toto řešení minimalizuje riziko vystavení zranitelných služeb mimo laboratorní prostředí, což odpovídá požadavkům na izolaci a bezpečné používání zranitelných aplikací.

V průběhu implementace vyvstala řada praktických úkolů. Mezi nimi byla nutnost ověřit stav kontejnerů a dostupnost cílové služby na zadaném portu. V některých situacích bylo nutné počkat na úplnou inicializaci aplikace nebo základní infrastruktury, protože služba nemusela být dostupná ihned po spuštění kontejneru. Z tohoto důvodu byly v laboratorních příručkách vždy podrobně popsány nejen postupy spuštění, ale i metody ověřování funkčnosti.

Jedním z klíčových praktických úkolů bylo zajistit možnost návratu do původního stavu. Právě v tomto kontextu prokázala kontejnerizace své nesporné výhody. Pokud v průběhu testování došlo k nežádoucím změnám, ať už se jednalo o poškození dat, selhání aplikace nebo nevratné úpravy prostředí, bylo možné scénář snadno zastavit a spustit znovu. To umožnilo dosáhnout úplné reprodukovatelnosti výsledků, a to i při intenzivním využívání pro vzdělávací účely. Tato funkce je obzvláště relevantní pro laboratorní práce zaměřené na aktivní výzkum a využívání zranitelností, kde je změna stavu systému přirozeným důsledkem plnění úkolů.

Zkušenosti s nasazením tohoto prostředí potvrdily, že použití Kali Linuxu v kombinaci s Dockerem a předem připravenými scénáři umožnilo dosáhnout optimální rovnováhy mezi technickou dostupností a realistickým chováním aplikací. Správa takového prostředí není nijak složitá, ale zároveň zajišťuje dostatečně přesné modelování útoků, jejich následků a metod ochrany. To má velký význam jak pro profesionální činnost, tak pro vzdělávací proces, protože studenti získávají možnost pracovat v podmínkách, které se co nejvíce blíží reálným systémům.

Výsledky zavedení tohoto prostředí jsou hodnoceny jako pozitivní, jelikož se podařilo vytvořit stabilní a standardizovanou základnu pro provádění všech následných laboratorních prací. Zvolený přístup ke správě kontejnerů, organizaci souborového systému a zajištění přístupu ke službám přispěl k efektivní přípravě a zjednodušení procesu testování jednotlivých scénářů. V budoucnu bude na základě vytvořené infrastruktury představen projekt konkrétních laboratorních úkolů určených k demonstraci zranitelností v kontextu OWASP Top 10.

4.3 Struktura laboratorního prostředí

Pro úspěšné fungování v podmínkách mnoha laboratorních scénářů nestačí pouze zajistit spolehlivost a správnost technické infrastruktury. Zásadní význam má také vnitřní architektura řešení, která určuje způsoby ukládání, spouštění a kontroly provádění jednotlivých úkolů. Absence jasné struktury nevyhnutelně povede k exponenciálnímu nárůstu složitosti správy s rostoucím počtem scénářů, čímž se prostředí stane nepraktickým pro vývojáře i studenty. Vzhledem k výše uvedenému byla implementována jednotná struktura adresářů, v níž je každý laboratorní úkol umístěn v samostatném adresáři. Tato metodika zajišťuje jasné oddělení jednotlivých scénářů a zároveň zachovává ucelenou a přehlednou organizaci celého prostředí. Každý úkol tedy funguje jako autonomní modul, který obsahuje všechny komponenty potřebné pro jeho inicializaci a následnou verifikaci.

V praxi to znamená, že kořenový adresář obsahuje hlavní složku pro provádění laboratorních prací. Uvnitř této složky je každé jednotlivé cvičení jednoznačně identifikováno, například pomocí pořadového čísla nebo tematického názvu. Typická struktura může vypadat následovně:

```
Student-Labs/
├── Cviceni1
├── Cviceni2
├── Cviceni3
├── Cviceni4
├── Cviceni5
├── ...
```

Tato struktura přináší značné provozní výhody. Především usnadňuje rychlé osvojení uživatelského prostředí, protože poskytuje přehledný obraz o dostupných scénářích a umístění pracovních dat. Kromě toho umožňuje nezávislé řízení každého úkolu, čímž se eliminuje nutnost zasahovat do ostatních komponent systému.

Každá složka představuje izolovaný kontejner pro materiály určené výhradně pro konkrétní laboratorní práci. Hlavní součástí je zpravidla soubor docker-compose.yml, který definuje konfiguraci prostředí. Navíc mohou být přítomny pomocné zdroje, jako jsou skripty pro provádění útoků (exploity), sady testovacích dat, textové soubory s informacemi nebo pokyny specifické pro daný scénář. Taková organizace zajišťuje úplnou autonomii každého laboratorního úkolu a umožňuje jejich provádění nezávisle na sobě.

Způsob organizace a správy kontejnerů je klíčovým aspektem této architektury. Každý laboratorní úkol se spouští ve vlastním kontejneru nebo ve skupině kontejnerů, které emulují cílovou aplikaci a její nezbytné služby. Může se jednat jak o samostatnou zranitelnou webovou aplikaci, tak o složitější scénáře vyžadující souběžné spuštění databáze nebo jiných závislých služeb. Z pohledu uživatele však proces zůstává jednotný: pro správu prostředí úlohy je nutné přejít do jejího adresáře a spustit příkazy pro spuštění nebo zastavení.

Tato architektura výrazně usnadňuje jak provoz, tak údržbu prostředí. V případě selhání jedné z úloh, změny konfigurace nebo nesprávného použití se negativní dopad omezuje výhradně na tento scénář, aniž by to mělo vliv na ostatní komponenty systému. Izolovanost jednotlivých úloh je tedy klíčovým faktorem pro zajištění celkové stability řešení.

Systém se vyznačuje značnou škálovatelností, což je jeho klíčová přednost. Integrace nových laboratorních úkolů probíhá vytvořením samostatných adresářů, které zachovávají vnitřní strukturu a logiku. Tento způsob vylučuje nutnost úprav stávajících scénářů nebo jejich konfiguračních souborů. Takový modulární přístup zaručuje možnost dalšího rozvoje platformy bez rizika narušení její integrity a funkčnosti.

Z pedagogického hlediska má navrhovaná struktura značné výhody. Studentům umožňuje pracovat v rámci přesně definovaných scénářů, kde každé cvičení má svůj jedinečný technický základ, sadu souborů a postup kroků. Tento přístup minimalizuje pravděpodobnost vzniku chyb a zároveň zvyšuje přehlednost výukového procesu. Navíc získává lektor možnost flexibilně integrovat jednotlivé scénáře do učebního plánu, spouštět je podle potřeby a považovat je za samostatné moduly.

Organizace laboratorního prostředí tedy přesahuje rámec pouhé správy souborů a představuje klíčovou strukturální složku. Zajišťuje modularitu, stabilitu a reprodukovatelnost celého vyvíjeného řešení. Díky své přehlednosti a samostatnosti získávají jednotlivé úkoly pevný základ pro následné navrhování, testování a integraci do vzdělávacího procesu.

4.4 Metodika práce

Cílem vytvoření laboratorních úkolů nebylo pouze vytvoření zranitelných aplikací, ale také napodobení realistického pracovního postupu, který je typický pro penetrační testování. K tomu jsem použil metodiku založenou na standardním cyklu analýzy bezpečnosti webových aplikací. Tento přístup umožňuje řešit úkoly nikoli izolovaně, ale v logickém sledu, podobně jako tomu je při skutečné kontrole bezpečnosti systému.

Celý proces je rozčleněn do tří klíčových fází: předběžný průzkum cíle útoku, samotné využití zjištěné zranitelnosti a následné vyhodnocení výsledků. Tato klasifikace prokazuje svou účinnost jak v praktickém použití, tak i ve vzdělávacím kontextu, přičemž zdůrazňuje, že každý útok začíná důkladným sběrem dat o cíli, jeho architektuře a specifikách fungování, a nikoli okamžikem použití exploitu.

V úvodní fázi se provádí průzkum, jehož cílem je shromáždit informace o cílové aplikaci nebo službě. Úkolem je zjistit, jaká aplikace se používá, na kterém portu běží, jaké funkce jsou k dispozici a kde se mohou skrývat zranitelnosti. Metody průzkumu závisí na konkrétním úkolu: může se jednat o určení verze aplikace na základě veřejně dostupných údajů nebo chyb, případně o analýzu HTTP provozu za účelem odhalení řídicích parametrů. V této fázi jsou nepostradatelné nástroje jako Burp Suite, Gobuster a Nmap, které pomáhají získat celkový přehled o systému.

V rámci této metodiky hraje průzkum klíčovou roli, protože bez něj jsou následné kroky buď neproveditelné, nebo se stávají pouhou hrou náhody. V této fázi si student uvědomí, že se nejprve nemá snažit systém zneužít, ale musí se v něm zorientovat. Navíc právě zde se

často rozhoduje o výběru dalších technik. Pokud tedy analýza odhalí zranitelný parametr HTTP, je logické pokračovat v ruční modifikaci požadavků. Pokud však bude odhalena známá zranitelnost v konkrétní verzi produktu, lze se uchýlit k hotovým exploitům nebo prověřit specifické vektory útoků.

Dalším krokem je fáze zneužití, ve které se teorie promění v praxi. Zde se student aktivně pokouší využít odhalenou zranitelnost a přechází od pasivní analýzy k přímé interakci se systémem. Způsoby zneužití se liší v závislosti na scénáři: od manipulace s HTTP požadavky a nahrávání škodlivých souborů až po padělání relačních cookies, zneužití slabých hashovacích algoritmů nebo použití hotových exploitů. Důležité je, že každá akce je podniknuta s konkrétním cílem, přičemž se ověřuje předem formulovaná hypotéza, a nikoli metodou pokusů a omylů.

Při provádění laboratorních úkolů nesloužilo zneužití zranitelností pouze k demonstraci jejich existence, ale také jako nástroj k pochopení jejich praktických důsledků. Samotné odhalení programové chyby totiž studentům ne vždy umožňuje plně posoudit její závažnost. Teprve když se podaří například získat přístup k příkazovému řádku, přečíst cizí data, získat práva správce nebo obejít autentizaci, stává se zranitelnost hmatatelnou a srozumitelnou. Právě tato praktická část metodiky má tedy největší vzdělávací hodnotu.

Závěrečná fáze výzkumu zahrnuje analýzu dopadů. V kontextu této práce má tato fáze zvláštní význam, protože cílem laboratorních prací nebylo pouze odhalení zranitelností, ale také demonstrace jejich praktického dopadu. Po úspěšném zneužití zranitelnosti bylo tedy provedeno vyhodnocení: objemu získaných informací, získaných oprávnění, dotčených komponent systému a potenciálních důsledků útoku v reálných podmínkách. V některých případech se to projevilo v přístupu k důvěrným údajům, v jiných – v získání informací z databází, úpravě obsahu aplikací, vytvoření nových účtů nebo narušení dostupnosti služeb.

Analýza z metodologického hlediska hraje důležitou roli při formování uceleného pohledu studentů na bezpečnost. Umožňuje nahlížet na útoky nejen jako na technickou výzvu, ale také jako na ohrožení důvěrnosti, integrity a dostupnosti dat. Jinými slovy, důležitý není tolik samotný fakt úspěšného zneužití zranitelnosti, jako spíše její důsledky pro fungování aplikace, práci systémových administrátorů a uživatelský zážitek. To přibližuje laboratorní úkoly reálným úkolům v oblasti zabezpečení, kde technická stránka útoku představuje pouze jeden z prvků hodnocení rizik.

Všechny laboratorní úlohy byly postaveny na jednotném metodologickém základu, a to navzdory specifikům zranitelností, které simulovaly. V závislosti na typu hrozby (ať už šlo o síťovou interakci, práci s daty, soubory nebo nastaveními aplikace) se měnily pouze detaily realizace. Postup však zůstal stejný: pochopení úkolu, ověření možnosti zneužití a následná analýza výsledků. To umožnilo vytvořit z jednotlivých úkolů jednotný, logicky propojený systém, sjednocený společnou metodikou.

Neméně důležité je, že zvolená výuková metoda přispívá k rozvoji klíčových kompetencí. Studenti se neučí pouze jednotlivé příkazy nebo nástroje, ale – a to je mnohem důležitější – učí se při řešení úkolů v oblasti bezpečnosti uvažovat systémově. Získávají schopnost klást si správné otázky: co je o cíli známo, co je ještě třeba zjistit, jak ověřit vyslovené

hypotézy a jak vyhodnotit získané výsledky. V konečném důsledku se takový přístup k výuce ukazuje jako cennější než pouhé memorování hotových pokynů, protože formuje schopnost samostatné analýzy a přijímání fundovaných rozhodnutí.

V této práci byl použit komplexní přístup, který spojuje technické aspekty penetračního testování s pedagogickými principy. Tato metoda nejen umožňuje simulovat útok, ale také přispívá k hlubokému pochopení jeho předpokladů, průběhu a výsledků. Právě synergie těchto složek umožnila vyvinout laboratorní cvičení, která přesahují rámec pouhé demonstrace zranitelností a slouží jako účinný vzdělávací zdroj pro pochopení jemných nuancí bezpečnosti webových aplikací.

5 Návrh a realizace laboratorních úloh

Vyvinuté a implementované laboratorní prostředí slouží jako základ pro následující část práce, která je věnována plnění praktických laboratorních úkolů. Cíl těchto úkolů přesahuje rámec pouhé technické demonstrace konkrétních zranitelností; je zaměřen na převedení teoretických znalostí do podoby, která je srozumitelná a vhodná pro praktické použití. V této fázi dochází ke spojení předchozí analýzy webových zranitelností s konkrétními scénáři, které umožňují bezpečně otestovat jednotlivé typy útoků a následně vyhodnotit jejich důsledky.

Cílem při přípravě laboratorních úkolů bylo zajistit jejich maximální praktickou hodnotu. Snažil jsem se, aby úkoly napodobovaly skutečné zranitelnosti, byly technicky proveditelné v rámci laboratorního prostředí a přispívaly k hlubokému porozumění látky ze strany studentů. Zvláštní pozornost byla věnována širokému pokrytí kategorií OWASP Top 10, aby si studenti mohli osvojit nejen techniky útoků, ale také si uvědomit jejich důsledky pro bezpečnost aplikací a informačních systémů.

V této kapitole se zaměříme na proces tvorby a strukturování laboratorních úkolů, jakož i na jejich praktickou realizaci. Zvláštní pozornost je věnována odůvodnění výběru představených scénářů, jejich přizpůsobení pro výukové účely a tomu, jak přispívají k rozvoji praktických kompetencí studentů. Výsledkem je soubor laboratorních prací, které tvoří jednotný systém cvičení, umožňující studentům v praxi studovat a demonstrovat zranitelnosti webových aplikací.

5.1 Návrh laboratorních úloh

Vypracování laboratorních úkolů zaujímalo v mé práci ústřední místo, protože právě zde bylo třeba převést obecné představy o bezpečnosti do konkrétních, technicky proveditelných a didakticky hodnotných scénářů. Mým cílem nebylo pouze katalogizovat známé zranitelnosti, ale také vytvořit soubor úkolů, který by se vyznačoval metodologickou celistvostí, vyvážeností a adekvátním zohledněním problémů ze seznamu OWASP Top 10:2025. Finální laboratorní práce se tak staly výsledkem profesionálního výběru, praktického ověření a neustálých úprav v souladu s reálnými možnostmi prostředí.

Výběr zranitelností proběhl na základě tří základních kritérií, jejichž cílem bylo zajistit maximální přínos. Prvním kritériem byla relevantnost pro profesionální činnost: vybíral jsem zranitelnosti, které se skutečně vyskytují v moderních webových aplikacích a odpovídají aktuálním kategoriím OWASP Top 10:2025. Druhým kritériem byla praktická použitelnost: hodnotil jsem, do jaké míry lze každý scénář spolehlivě reprodukovat a demonstrovat v laboratorních podmínkách. Třetím kritériem byla efektivita výuky: kromě demonstrace zneužití měly úkoly sloužit jako nástroj k pochopení podstaty zranitelnosti, metod jejího využití a vlivu na celkovou bezpečnost systému ze strany studentů.

Zvláštní pozornost byla věnována tomu, aby výsledný soubor úkolů co nejúplněji odrážel rozmanitost kategorií OWASP Top 10:2025. Cílem nebylo pouze klasifikovat jednotlivé

scénáře, ale vybrat takové zranitelnosti, které jsou nejen reprezentativní pro své kategorie, ale také umožňují hloubkově prozkoumat jejich podstatu: od příčin vzniku a metod zneužití až po dopad na důvěrnost, integritu a dostupnost.

Čas potřebný k vyřešení každého úkolu se stal podstatným faktorem při rozhodování. Vzhledem k tomu, že laboratorní scénáře byly určeny pro kurz „Počítačové viry a bezpečnost“, musel jsem je přizpůsobit reálným podmínkám výuky. Úkoly byly proto strukturovány tak, aby jejich hlavní část mohla být dokončena v rámci jednoho výukového bloku, tedy za 40–50 minut. Toto omezení se ukázalo jako velmi důležité, protože některé technicky zajímavé scénáře by byly pro standardní výukový proces příliš rozsáhlé nebo by vyžadovaly nadměrné přípravné činnosti.

Kromě časového rámce bylo klíčové stanovit přiměřenou úroveň obtížnosti úkolů. Příliš jednoduché úkoly by studentům neposkytly hluboké odborné porozumění testovaným problémům. Na druhou stranu by příliš složité scénáře mohly vést k tomu, že by se studenti ztratili v technických detailech a přehlédli samotnou podstatu zranitelnosti. Proto jsem při tvorbě úkolů hledal zlatou střední cestu mezi realističností a srozumitelností. Každý úkol měl mít jasně formulovaný cíl, srozumitelný podrobný návod a možnost změřit výsledek, přičemž měl zachovávat odbornou úroveň odpovídající požadavkům diplomové práce.

V rámci této práce bylo zjištěno, že ne každá objevená zranitelnost je vhodná pro využití ve výuce v laboratorních podmínkách. Pro úspěšnou výuku je nutné, aby scénář nejen obsahoval zranitelnost, ale také vykazoval takové vlastnosti, jako je stabilita, spolehlivost spuštění, reprodukovatelnost a pedagogická přístupnost. Praktické potíže, které v průběhu práce vznikly v souvislosti s těmito požadavky, měly podstatný vliv na sestavení finálního souboru laboratorních prací.

Volba projektu Vulhub jako hlavního zdroje scénářů se zranitelnostmi byla podmíněna jeho schopností poskytovat předem nakonfigurovaná prostředí Docker obsahující aktuální a obecně známé zranitelnosti. Z metodologického hlediska se toto řešení ukázalo jako velmi účinné, protože umožnilo pracovat se skutečnými scénáři bez nutnosti jejich úplného ručního modelování. To zase přispělo k udržení vysoké úrovně technické autenticity jednotlivých úkolů. Nicméně v průběhu práce bylo zjištěno, že ne všechny scénáře prezentované v rámci projektu Vulhub mají stejnou vhodnost pro praktické použití v laboratorních podmínkách.

Při používání různých kontejnerů jsem narazil na řadu problémů. Některé z nich se odmítaly spustit, jiné trpěly nestabilitou, nedostatkem dokumentace nebo nepředvídatelným chováním během provozu. Část scénářů byla funkční pouze částečně nebo vyžadovala natolik specifické podmínky, že by jejich zavedení do výuky bylo neúčelné. Vyskytly se také případy, kdy byla zranitelnost, ačkoli spadala do určité kategorie OWASP, buď příliš obtížná na demonstraci, nebo příliš vázaná na konkrétní verze systémů, nebo naopak příliš vzdálená úrovni přípravy studentů. To mě stavělo před volbu: zachovat maximální technickou věrohodnost, nebo scénář upravit (či zcela přepracovat) za účelem dosažení maximální vzdělávací účinnosti.

Konečný výběr laboratorních úkolů vycházel z jejich praktické demonstrační hodnoty. Kromě existence zranitelnosti a jejího teoretického zařazení do dané kategorie se klíčovým

faktorem stala možnost provést jasnou, stabilní a reprodukovatelnou demonstraci. Zvláštní pozornost jsem věnoval ověření použitelnosti scénářů v mém prostředí, jejich bezpečné integraci s Dockerem a zaručení jednoznačného výsledku při provozu. Tato praktická selekce měla rozhodující vliv na sestavení finálního souboru úkolů.

Důležitým úkolem bylo zajistit, aby metodika práce s úkoly od samého počátku odpovídala zvolenému didaktickému přístupu. Cílem bylo, aby úkoly byly pro studenty zvládnutelné, aniž by je přetěžovaly zbytečnými technickými obtížemi, a zároveň zachovávaly realistický kontext bezpečnosti. To znamenalo nutnost zjednodušit některé technické aspekty, aniž by se přitom ztratila podstata útoku. Jednoduše řečeno, úkoly měly být dostatečně jednoduché na provedení, ale ne natolik, aby přestaly být relevantní pro reálné scénáře. Tento požadavek se stal jednou z nejzávažnějších výzev v průběhu celého procesu vývoje laboratorních úkolů.

V některých případech bylo například nutné změnit způsob využití zranitelnosti nebo zkrátit rozsah úkolu, aby jej student mohl splnit v přijatelném čase a zároveň pochopit princip fungování exploitu. V jiných úkolech naopak bylo nutné zachovat vysokou úroveň technické realističnosti, aby bylo možné adekvátně ukázat důsledky dané zranitelnosti. Konečná podoba úkolů tak vždy představuje rovnováhu mezi odbornou přesností, technickou proveditelností a přístupností pro výuku.

Na základě stanovených kritérií byl sestaven finální seznam deseti laboratorních úkolů. U většiny z nich byly využity hotové kontejnery z projektu Vulhub. Tyto kontejnery zajistily praktické scénáře co nejvíce se blížící reálným podmínkám pro demonstraci vybraných zranitelností. Tímto způsobem se mi podařilo pokrýt podstatnou část kategorií OWASP Top 10:2025, přičemž jsem pracoval se skutečnými aplikacemi a známými metodami zneužití. Zároveň se podařilo zachovat jednotnou technologickou platformu, zejména díky použití Dockeru a Kali Linuxu.

Tab. 1: Přehled navržených laboratorních úloh

Číslo úlohy	Název úlohy	Kategorie OWASP	Zdroj kontejneru
1	Drupal	A05	Vulhub
2	Joomla	A05	Vulhub
3	Redis	A02	Vulhub
4	Apache Struts2	A05 / A08	Vulhub
5	Spring4Shell	A03 / A05	Vulhub
6	Superset	A07 / A01	Vulhub
7	PHP IMAP	A10	Vulhub
8	PHP XXE Injection	A06	Vulhub
9	SecureNote	A04	Autorské řešení
10	Employee Portal	A09	Autorské řešení

(zdroj: vlastní)

V průběhu práce byly zjištěny závažné problémy v kategoriích A04 Kryptografické selhání a A09 Selhání v logování a oznamování bezpečnostních incidentů. Analýza projektu Vulhub odhalila nedostatek vhodných a plně funkčních scénářů, které by splňovaly požadavky této práce. Stávající podobné scénáře vykazovaly nedostatečnou stabilitu, neumožňovaly jednoznačně ověřit zranitelnosti nebo neodpovídaly stanoveným pedagogickým cílům. V důsledku toho bylo rozhodnuto o samostatném vývoji posledních dvou laboratorních úkolů ve formě samostatných webových aplikací.

První z představených aplikací, SecureNote, byla speciálně vyvinuta za účelem ilustrace kryptografických zranitelností. Jeho cílem je ukázat, jak kombinace několika běžných problémů z kategorie A04 – konkrétně použití hashové funkce MD5 bez přidání „soli“, použití režimu ECB pro šifrování AES-128 a přítomnost hardcoded kryptografického klíče – představuje značná rizika. Vývoj SecureNote umožnil vytvořit scénář, který je nejen technicky správný, ale má také vysokou pedagogickou hodnotu, protože názorně ukazuje, že kryptografické neúspěchy jsou často výsledkem nejen jedné chyby, ale komplexu nesprávných rozhodnutí. Tímto způsobem se daný autorský úkol stal zdrojem jak technického pokroku, tak tvůrčího přínosu k projektu.

Druhá aplikace vyvinutá autorem, Employee Portal, byla vytvořena s cílem pokrýt kategorii A09, která se vyznačuje nedostatečnou úrovní vedení protokolů a upozorňování. Hlavní důraz byl kladen na modelování situace, kde klíčovým problémem je absence nebo nedostatečnost detekčních mechanismů, nikoli jednotlivé zranitelnosti. Aplikace je navržena tak, aby umožňovala provedení řady typických útoků, včetně SQL injekce, Stored XSS a porušení přístupových práv, přičemž zajišťuje nedostatečné zaznamenávání událostí. Tento přístup umožňuje studentům ukázat, že bezpečnost zahrnuje nejen prevenci neoprávněného přístupu, ale také schopnost organizace zaznamenávat, analyzovat a včas reagovat na incidenty. Tato úloha tak představuje významný autorský přínos k práci.

Konečná sada laboratorních úkolů byla sestavena na základě integrace externích a vlastních scénářů. Prvních osm úkolů vychází z materiálů projektu Vulhub, zatímco poslední dva byly vypracovány autorem s ohledem na specifika práce a požadavky na pokrytí kategorií OWASP Top 10:2025. Tato strategie se ukázala jako nejvhodnější, protože zajistila zachování technické autenticity v rámci dostupných externích zdrojů a umožnila doplnit chybějící aspekty prostřednictvím vlastních vývoje.

Z metodologického hlediska lze tuto kombinaci považovat za významnou výhodu. Použití hotových kontejnerů zajistilo realističnost a umožnilo efektivně pracovat se známými zranitelnostmi, stejně jako s již existujícími aplikacemi. Autorské scénáře pak rozšířily možnosti řešení tím, že do něj zahrnuly oblasti, které nebylo možné pokrýt standardními metodami. Výsledkem byl komplex úkolů, který není pouhým souborem hotových prostředků, ale uceleným, profesionálně podloženým systémem, plně odpovídajícím úkolům diplomové práce.

Ukázalo se, že právě tato část práce je z hlediska autorské iniciativy a tvůrčího přístupu nejvýraznější. Příprava laboratorních úkolů nebyla pouhým mechanickým kopírováním, ale komplexním procesem zahrnujícím výběr, ověření, přizpůsobení, metodické zpracování

a v případě potřeby i vytvoření vlastních jedinečných úkolů. Právě proto lze vypracování laboratorních úkolů považovat za základní kámen celé práce, kde se teoretické znalosti o bezpečnosti spojují s praktickými zkušenostmi a schopností předat je formou přístupnou pro výuku.

5.2 Struktura laboratorních úloh

Po identifikaci konkrétních scénářů vznikla potřeba vypracovat jednotnou strukturu laboratorních úkolů. Tato struktura měla zajistit jak technickou reprezentativnost, tak didaktickou vhodnost. Jednotná forma prezentace se ukázala jako kriticky důležitá, protože tvoří společný základ pro všechna cvičení a umožňuje studentům soustředit se na podstatu zranitelností, nikoli na variabilitu formulací nebo organizaci každého cvičení. Úkolem tedy nebylo pouhé sestavení pokynů, ale vytvoření souboru úkolů, který by studentům poskytl nezbytný kontext, jasně definované cíle, praktická doporučení a zároveň možnosti pro samostatnou reflexivní analýzu výsledků.

Každé laboratorní cvičení bylo strukturováno jako samostatný protokol sestávající z několika vzájemně propojených částí. Navzdory rozmanitosti technického obsahu, dané specifikou zkoumaných zranitelností, zůstala základní vnitřní struktura protokolů neměnná. Toto rozhodnutí bylo přijato záměrně. Z pohledu studentů spočívá výhoda v tom, že opakující se logika práce minimalizuje kognitivní zátěž spojenou s osvojováním formátu. Z pohledu vyučujících zajišťuje jednotná struktura celistvost systému úkolů, což usnadňuje jejich integraci a systematické využití ve vzdělávacím procesu.

Struktura každého úkolu zahrnuje úvodní teoretickou část. Jejím cílem je stručně a výstižně objasnit podstatu konkrétní zranitelnosti. Tato část si neklade za cíl poskytnout vyčerpávající teoretický výklad, ale spíše slouží jako praktický úvod pro studenta a připravuje ho na řešení následujících úkolů. Proto teoretická analýza vždy vysvětluje původ zranitelnosti, použité vektory útoků, její zařazení do určitých typů aplikací nebo komponent, jakož i možné negativní dopady. V některých případech je také nutné osvětlit specifické technologické nuance, jako je interakce s HTTP požadavky, zpracování XML dat, logika autentizace nebo význam konkrétních parametrů v kontextu aplikace.

Teoretická analýza hraje klíčovou roli, protože pomáhá studentovi přejít od role pouhého vykonavatele k roli uvědomělého účastníka procesu. Místo toho, aby slepě plnil jednotlivé kroky, student začíná chápat jejich odůvodnění. To je obzvláště důležité v případech, kdy samotná posloupnost příkazů bez vysvětlení působí mechanicky a nepřispívá k hlubokému proniknutí do podstaty problému. Teoretická část tak funguje jako průvodce, který propojuje obecné bezpečnostní koncepty s konkrétními praktickými úkoly laboratorních prací. Zároveň je nutné dbát na to, aby se nerozrostla do samostatné, nadbytečné kapitoly, ale sloužila jako jasný a cílený úvod do kontextu konkrétního scénáře.

Po teoretické části laboratorní práce následuje praktický úkol, který určuje, co přesně má student udělat. Je koncipován tak, aby se neomezoval pouze na výčet jednotlivých úkonů, ale jasně vyjadřoval smysl celé práce. Úkol proto začíná stručným popisem cíle a poté je

rozčleněn do řady konkrétních bodů, které představují hlavní etapy, jimiž musí student projít.

V této práci bylo znění zadání klíčové pro dosažení optimální přesnosti. Příliš obecné zadání by mohlo u studentů vyvolat zmatek a nejistotu ohledně požadovaného výsledku. Příliš podrobný úkol by zase riskoval, že se bude překrývat s částí o postupu, čímž by ztratil svou roli stručného vodítka. Úkoly proto byly prezentovány jako seznam klíčových cílů cvičení, přičemž technické detaily byly ponechány pro následující část.

Samostatná a podstatná část je věnována organizaci laboratorního prostředí, což bylo dáno praktickými důvody. Vzhledem k tomu, že všechny scénáře vycházejí z jednotného laboratorního prostředí, bylo nutné studentům podrobně předvést postup spuštění zadaného úkolu a způsoby ověření jeho připravenosti k řešení. Tato část zpravidla zahrnuje navigaci do příslušného pracovního adresáře, inicializaci potřebného kontejneru nebo webové služby a kontrolu dostupnosti aplikace prostřednictvím prohlížeče nebo příkazového řádku.

Tato část má zásadní metodologický význam, protože simuluje reálný přístup k testování bezpečnosti. Na rozdíl od přímého zahájení útokem zahrnuje reálná praxe předběžnou přípravu prostředí, ověření funkčnosti aplikace a prozkoumání její struktury. Studenti si tak od samého začátku osvojují důležitou dovednost: jakékoli testování musí vycházet z předem známého a kontrolovaného výchozího stavu. Pro autorské aplikace SecureNote a Employee Portal je tato část obzvláště důležitá, protože dokazuje, že testovací scénáře nebyly pouze převzaty zvenčí, ale byly zcela vyvinuty a jsou řízeny v rámci daného projektu.

Podrobný návod tvoří největší část každého laboratorního úkolu. Podrobně ukazuje, jak zneužít zranitelnosti nebo analyzovat bezpečnost. Návod je navržen tak, aby studentům poskytoval přesné pokyny a zároveň napodoboval logiku skutečného penetračního testování. To znamená, že jednotlivé kroky sledují přirozený pracovní postup: od počáteční kontroly funkčnosti a odhalení zranitelného místa až po jeho využití a vyhodnocení následků.

Aby byly pokyny co nejefektivnější, vědomě jsem se vyhnul dvěma potenciálním problémům. Prvním z nich je nedostatečná podrobnost, která by mohla studenty připravit o nezbytnou podporu, což by vedlo k chybám a nepochopení úkolu. Druhým problémem je přílišná automatizace, kdy studenti pouze slepě následují pokyny, aniž by chápali jejich smysl. Proto byl každý krok v pokynech navržen s ohledem na tato rizika: obsahuje jak konkrétní technické pokyny, tak stručné vysvětlení cíle a očekávaného výsledku.

Ačkoli se některé úkoly zaměřují na HTTP interakci, jako je tomu například při práci s Burp Suite, a jiné na příkazový řádek, webové shelly, databáze nebo systémovou konfiguraci, základní logika zůstává stejná. Studenti musí postupně provádět kontroly, provádět konkrétní akce a analyzovat výsledky z hlediska souladu se stanovenými cíli. To přispívá k rozvoji nejen technických dovedností, ale také kritického myšlení a schopnosti interpretovat získaná data.

V rámci tohoto přístupu se klade zvláštní důraz na potvrzení účinnosti provedených opatření. To znamená, že pouhé napodobení útoku nestačí – je třeba přesvědčivě prokázat, že byl úspěšný a vedl ke konkrétním, měřitelným důsledkům. Příkladem takových důkazů může být: vytvoření souboru, odhalení důvěrných údajů, úspěšné přihlášení pod účtem

jiného uživatele, získání systémových informací nebo narušení dostupnosti služby. Dodržování tohoto principu je zásadně důležité jak pro zajištění profesionální preciznosti, tak pro kvalitní dokumentaci procesu, protože poskytuje studentovi i lektorovi jasné důkazy o úspěšném splnění zadaného úkolu.

Z pedagogického hlediska spočívá hodnota tohoto přístupu v tom, že se neomezuje na soubor nesouvislých technických pokynů, ale je začleněn do celkové logiky řešení úkolu. Postup kroků, kde každý následující krok vyplývá z předchozího a směřuje k jasně formulovanému cíli, umožňuje studentovi vnímat jednotlivé příkazy nikoli jako chaotický soubor akcí, ale jako smysluplný řetězec vedoucí od formulace problému k jeho empirickému potvrzení.

V rámci struktury laboratorních prací hrají kontrolní otázky klíčovou roli. Byly cíleně začleněny do protokolů jako prvek podněcující analytické myšlení a reflexi. Hlavním úkolem těchto otázek není opakování mechanických úkonů, které student již provedl, ale prohloubení porozumění laboratornímu úkolu prostřednictvím interpretace, srovnání a pochopení získaných výsledků. Kontrolní otázky tak představují významný doplněk, který přesahuje rámec čistě technického provedení.

V této části se student začíná zabývat nejen samotným procesem, ale i jeho významem. Kontrolní otázky jsou například zaměřeny na to, aby student vysvětlil princip zranitelnosti, rozlišil mezi ručním a automatizovaným provozem, analyzoval dopady na důvěrnost a dostupnost dat, navrhl možnosti ochrany a určil jejich místo v rámci OWASP Top 10. V některých případech mohou tyto otázky také podnítit srovnání laboratorních zkušeností se skutečnými hrozbami nebo úvahy o tom, jak by se situace vyvíjela v produkčním prostředí.

Zavedení kontrolních otázek výrazně zvýšilo efektivitu laboratorních prací. Nutí studenty, aby se po úspěšném splnění úkolu nezastavili na dosaženém výsledku, ale vrátili se k provedené práci a zhodnotili ji s ohledem na širší aspekty bezpečnosti. Laboratorní cvičení tak přestávají být pouhou technickou dovedností a stávají se nástrojem pro rozvoj kritického myšlení. Zároveň to dává vyučujícímu možnost přesně určit, do jaké míry student pochopil podstatu úkolu, a ne jen mechanicky splnil pokyny.

Standardizovaná struktura laboratorních prací otevírá možnosti pro hlubší analýzu a srovnání. Jelikož mají všechny úkoly společný základ, je snazší je hodnotit nejen z hlediska technické náročnosti, ale i podle parametrů, jako je didaktická hodnota, čas potřebný k jejich splnění a povaha požadovaných znalostí. To má velký význam pro následné využití ve vzdělávacím procesu, protože umožňuje učitelům přijímat uváženější rozhodnutí o tom, která zadání se nejlépe hodí pro začátečníky a která pro studenty s vyšší úrovní přípravy.

Komplexní přístup k organizaci práce přispívá ke zvýšení celkové úrovně formalizace procesů. Vypracované protokoly již nejsou vnímány jako izolované dokumenty, ale nabývají charakteru systematizovaného souboru vzájemně propojených laboratorních cvičení. Přestože je každý úkol zaměřen na odhalení specifických kategorií zranitelností, všechny se drží jednotného formátu prezentace. Takový přístup posiluje celistvost celého souboru materiálů a zároveň vytváří základ pro jejich následnou adaptaci a škálování.

Laboratorní úkoly tedy nepředstavují pouhé rozdělení látky na jednotlivé části. Slouží jako základ metodiky, která vede studenta při jeho práci na problému bezpečnosti. Tento proces zahrnuje všechny fáze: od pochopení podstaty problému a jeho praktického zkoumání až po hloubkovou analýzu získaných výsledků. Právě tato komplexní interakce teorie, praktických úkolů, algoritmů provádění a kontrolních otázek prokázala svou účinnost při dosahování jak profesních, tak vzdělávacích cílů tohoto kurzu.

5.3 Realizace laboratorních úloh

Se zavedením jednotné struktury laboratorních protokolů bylo možné zahájit praktickou práci na jednotlivých úkolech. Právě tyto úkoly představují hlavní praktickou hodnotu celé provedené práce, neboť slouží jako přímé ztvárnění teoretických znalostí a metodických přístupů v konkrétních laboratorních scénářích. Úkolem přitom nebylo pouze technické ztvárnění zranitelných aplikací, ale také vytvoření souboru úkolů, který by byl profesně relevantní, snadno pochopitelný v rámci výuky a dostatečně variabilní, aby se studenti seznámili s širokým spektrem webových zranitelností a jejich důsledků.

Uvedená sada obsahuje deset laboratorních úkolů, které byly vypracovány s cílem co nejúplněji pokrýt kategorie OWASP Top 10:2025. Tabulka 1 obsahuje podrobný přehled každého z navrhovaných scénářů, včetně jejich zařazení do příslušných kategorií OWASP a popisu původu použitých prostředí. Již na základě tohoto přehledu je zřejmé, že výběr úkolů nebyl určen pouze technickou dostupností konkrétních kontejnerů. Důležitým faktorem bylo také úsilí o vytvoření uceleného systému scénářů, který by kombinoval jak známé a snadno reprodukovatelné zranitelnosti, tak i autorská řešení, která jsem vyvinul sám v situacích, kdy chyběla vhodná externí prostředí.

Prvních osm úkolů bylo postaveno na kontejnerech Vulhub, které poskytly spolehlivý technologický základ pro demonstraci řady dobře známých zranitelností. Samy o sobě však tyto kontejnery nepředstavovaly hotové výukové materiály. Každý z nich vyžadoval metodickou úpravu: nejen spuštění prostředí, ale také vypracování jasného postupu, určení potřebné hloubky ponoření, výběr adekvátního rozsahu demonstrace a doplnění vysvětlení o důsledcích a ochranných opatřeních. Praktická hodnota práce tedy spočívala nejen v technickém nasazení, ale také v pečlivém didaktickém zpracování.

V rámci prvního laboratorního úkolu se provádí analýza systému Drupal a zranitelnosti CVE-2018-7602. Volba tohoto scénáře je dána jeho reprezentativností jako příkladu útoku typu „vzdálené spuštění kódu“ a také jeho schopností názorně demonstrovat kritičnost kategorie A05: „Injekce“ v kontextu moderních hrozeb. Studenti budou muset projít kompletním cyklem práce se zranitelností, počínaje určením verze cílové aplikace, výběrem vhodných nástrojů pro její zneužití a konče přímým provedením útoku. Podstatná část úkolu je věnována demonstraci důsledků úspěšného zneužití, včetně možnosti provádění libovolných příkazů na serveru, úpravy obsahu aplikace a narušení její dostupnosti. Tento scénář se tak mění z čistě technické demonstrace exploitu na komplexní simulaci skutečné bezpečnostní události.

Druhé zadání se týká práce se systémem Joomla a zranitelnosti CVE-2017-8917, která představuje klasický případ SQL injekce. Toto zadání má velký pedagogický význam, protože skvěle ilustruje rozdíl mezi ručním a automatizovaným přístupem k testování. Nejprve studenti ručně analyzují HTTP požadavky, identifikují zranitelný parametr a používají techniky založené na chybách k získání informací z databáze. Teprve poté je předvedena práce nástroje sqlmap, která ukazuje, jak lze celý proces automatizovat. Úloha tak rozvíjí jak technické dovednosti, tak porozumění procesu ručního ověření zranitelnosti a následného automatizovaného získávání dat.

Třetí laboratorní úkol je věnován prozkoumání Redisu a představuje úkol, který se liší od předchozích scénářů. Hlavní zranitelnost v tomto případě není způsobena logickými chybami aplikace, ale nesprávnou konfigurací služby, což odpovídá klasifikaci A02: Security Misconfiguration. Zařazení tohoto úkolu má za cíl rozšířit záběr studovaných problémů a vyhnout se jednostrannému zaměření na scénáře injekcí nebo vzdáleného spuštění kódu. Studenti budou muset provést analýzu dostupnosti služby bez autentizace, prostudovat konfigurační parametry a otestovat možnosti zápisu dat na serverový disk. Význam tohoto úkolu spočívá v demonstraci toho, jak i zdánlivě nevýznamné chyby v konfiguraci mohou vést k závažným bezpečnostním rizikům. Zdůrazňuje také, že bezpečnostní hrozby mohou vzniknout nejen na úrovni zdrojového kódu aplikace, ale i v jejích provozních nastaveních.

Čtvrté laboratorní cvičení je zaměřeno na studium zranitelnosti S2-066 CVE-2023-50164, která byla objevena ve frameworku Apache Struts2. Tento scénář je obzvláště zajímavý, protože přesahuje rámec tradiční klasifikace OWASP a týká se současně dvou kategorií: „Nezabezpečené zpracování vstupních dat“ (A05) a „Porušení integrity“ (A08). Konkrétně se tato zranitelnost projevuje nesprávným zpracováním uživatelského vstupu a manipulací s požadavky během procesu načítání souborů. Studenti postupně provádějí následující kroky: ověření funkčnosti nahrávání, použití technik obcházení omezení manipulací s přístupovými cestami k souborům a nakonec nahrání škodlivého JSP webového shellu. Zvláštní pozornost je věnována příčinné souvislosti mezi počáteční manipulací s názvem souboru a následným získáním možnosti provádět libovolné příkazy na serveru. Tento úkol tak názorně ilustruje, jak i zdánlivě triviální funkčnost webové aplikace může při nedostatečné pozornosti věnované bezpečnosti ve fázi návrhu vést k úplnému kompromitování systému.

Páté laboratorní cvičení je věnováno analýze chyby Spring4Shell CVE-2022-22965 a představuje komplexní scénář, který zahrnuje zranitelnosti z kategorií A03 a A05. Na jedné straně se zaměřuje na zranitelnost vlastní konkrétnímu frameworku a jeho mechanismům zpracování dat. Na druhé straně je zneužití této zranitelnosti prováděno prostřednictvím cílené manipulace se vstupními parametry. Realizace tohoto úkolu patří k technicky nejnáročnějším, protože od studentů vyžaduje hluboké porozumění HTTP protokolu a schopnost modifikovat požadavky za účelem změny konfigurace Tomcatu, což v konečném důsledku umožňuje vytvořit JSP webový shell. Následující fáze zahrnují analýzu získaného přístupu prostřednictvím čtení konfiguračních souborů, systémových informací a dalších dat potvrzujících míru kompromitace. Tento úkol má značnou profesionální hodnotu, protože názorně

demonstruje, jak může zranitelnost v populárním frameworku ohrozit bezpečnost celé serverové infrastruktury.

Šesté laboratorní cvičení je zaměřeno na praktické uplatnění znalostí o zranitelnosti CVE-2023-27524 v kontextu Apache Superset. Podle přijaté klasifikace zahrnuje toto cvičení dvě kategorie hrozeb: A07 Obcházení autentizace a A01 Neoprávněný přístup k funkcím. Zařazení do kategorie A07 je dáno tím, že scénář útoku předpokládá možnost generování platného relačního cookie bez znalosti přihlašovacích údajů uživatele. Důsledky úspěšného zneužití této zranitelnosti plně odpovídají kritériím kategorie A01, protože útočník získá neoprávněný přístup k funkcím přesahujícím rámec jeho oprávnění, včetně úplné kontroly nad systémem. Dvojí klasifikace je v tomto případě přirozená, protože v praxi jsou obcházení autentizace a následné získání administrativních oprávnění často neoddělitelné. Z hlediska vzdělávacího procesu má tento úkol vysokou hodnotu, protože studentům názorně ukazuje, že bezpečnostní problémy se mohou projevovat nejen na úrovni zpracování dat, ale také ve fázích autentizace a správy relací. Následné akce, jako je vytváření nových uživatelů, konfigurace ovládacích panelů a přístup k důvěrným informacím, slouží jako přesvědčivá ilustrace potenciálních škod.

Sedmé laboratorní cvičení je věnováno studiu zranitelnosti CVE-2018-19518, která byla objevena v komponentu PHP IMAP. Tento praktický příklad má vysokou metodologickou hodnotu, protože názorně demonstruje, jak mohou zranitelnosti vzniknout v důsledku nesprávného zpracování výjimečných situací, a nikoli pouze v důsledku standardních chyb při zadávání. Právě proto patří tento úkol do kategorie A10: Mishandling of Exceptional Conditions. V průběhu řešení studenti nejprve analyzují fungování funkce imap_open() a její vstupní parametry. Poté uměle vytvoří výjimečnou situaci, ve které aplikace nesprávně zpracuje zadaná data. To povede k možnosti spuštění systémových příkazů a v důsledku toho k získání neoprávněného přístupu k serveru. Důležitost tohoto úkolu spočívá v tom, že rozšiřuje porozumění studentů o způsobech vzniku zranitelností a ukazuje alternativní cestu, odlišnou od běžných útoků typu „injekce“ do hlavního toku aplikace.

Osmé laboratorní cvičení je zaměřeno na zkoumání zranitelnosti XXE v kontextu PHP aplikací. Tato zranitelnost spadá do kategorie A06: Nebezpečný návrh, což zdůrazňuje, že její příčiny spočívají v architektonických nedostatcích, nikoli v ojedinělých programových chybách. V rámci úkolu budou studenti provádět modifikaci HTTP požadavků s cílem předat aplikaci speciálně vytvořené XML dokumenty obsahující externí entity. To jim umožní posoudit možnost neoprávněného čtení interních souborů systému. Podstatná hodnota tohoto úkolu spočívá v tom, že u studentů vytváří hluboké pochopení toho, že zranitelnosti mohou být způsobeny nejen nesprávným zpracováním vstupních dat v úzkém smyslu, ale také mylným předpokladem o bezpodmínečné bezpečnosti XML parseru. Kromě toho úkol přispívá k rozvoji dovedností interpretace chování aplikace při interakci s různými typy vstupních dat a různými zdroji dat.

Devátý laboratorní úkol představuje mé první zcela originální řešení, které bylo vytvořeno speciálně pro účely tohoto projektu. Aplikace SecureNote byla vyvinuta pro kategorii A04: Cryptographic Failures, protože v rámci projektu Vulhub nebyl nalezen vhodný

a zároveň stabilní scénář. Místo vytvoření uměle zjednodušeného příkladu jsem si dal za úkol vyvinout aplikaci, která studentům názorně předvede několik různých kryptografických problémů. Za tímto účelem byly do aplikace integrovány takové zranitelnosti, jako je slabé hashování hesel (MD5 bez solného klíče), nesprávné použití režimu AES-128-ECB a hard-coded kryptografický klíč. Z pedagogického hlediska je tento úkol cenný tím, že ilustruje kryptografické selhání jako komplex vzájemně propojených zranitelností, nikoli jako izolované problémy. Kromě toho je právě v tomto úkolu nejlépe naplněn požadavek na vlastní tvůrčí přínos, protože aplikaci jsem zcela vyvinul a realizoval sám.

Desátá laboratorní práce, která je výsledkem samostatného vývoje, se věnuje kategorii A09 OWASP: „Selhání v oblasti vedení bezpečnostních protokolů a upozornění“. Aby se demonstrovalo, že bezpečnostní problém nemusí spočívat pouze v existenci zranitelností, ale také v neschopnosti systému zaznamenávat a analyzovat útoky, byla vyvinuta aplikace Employee Portal. V rámci tohoto úkolu student provede řadu typických útoků, včetně SQL injekce, Stored XSS a porušení přístupových práv, s následnou kontrolou zaznamenání těchto akcí v systému protokolování. Význam tohoto úkolu spočívá v přechodu od zneužití zranitelností k hodnocení úrovně zabezpečení a analýze detekčních mechanismů. To předpokládá podstatný vlastní přínos, protože scénář byl vyvinut od nuly s ohledem na specifika cílové kategorie OWASP.

V rámci celého souboru laboratorních úkolů jsem dbál na to, aby každý jednotlivý úkol představoval určitý krok ve výuce a nabízel různou úroveň obtížnosti. Úlohy se liší nejen typem zranitelnosti, ale také použitými nástroji, technikami a hloubkou analýzy. Některé z nich vyžadují přímou práci s HTTP komunikací, jiné použití specializovaných nástrojů pro exploitaci a třetí analýzu konfigurací, kryptografie nebo logů. To bylo záměrné, aby nevznikla série technicky homogenních úkolů, ale aby byl vytvořen komplex, který studentům názorně předvede široké spektrum problémů webové bezpečnosti a rozmanitost metod jejich zkoumání.

Současně s tím bylo klíčovým požadavkem zachování jednotnosti metodického přístupu. Ačkoli měl každý scénář své technická specifika, jejich zpracování bylo sjednoceno s cílem zajistit plynulý výukový proces: od uvědomění si zranitelnosti přes její praktické prozkoumání až po následnou analýzu důsledků. Tento přístup zaručoval, že soubor úkolů nebyl vnímán jako soubor nesourodých technik hackování, ale jako ucelený, strukturovaný vzdělávací systém.

Vytváření laboratorních úkolů přesahuje rámec pouhé technické realizace deseti scénářů. Jedná se o vytvoření komplexního výukového systému, který integruje skutečné zranitelnosti, jednotné laboratorní prostředí, logicky sestavené protokoly a autorský obsah vyvinutý za účelem vyplnění mezer ve stávajících scénářích. Právě tato synergie zajišťuje odbornou a pedagogickou hodnotu celé sady úkolů a vyzdvihuje ji jako ústřední složku veškeré odvedené práce.

5.4 Testování úloh

Po dokončení jednotlivých laboratorních prací byla provedena kontrola jejich praktické použitelnosti a souladu s požadavky stanovenými při jejich vývoji. Testování bylo zaměřeno na posouzení nejen technické funkčnosti zranitelných aplikací, ale také stability prostředí, srozumitelnosti postupů a efektivního využití času v rámci vzdělávacího procesu. Cílem této fáze bylo potvrdit reprodukovatelnost laboratorních úkolů, dosažení očekávaných výsledků a soulad jejich rozsahu s předpokládaným využitím v rámci výukového bloku.

V úvodní fázi bylo provedeno posouzení funkční způsobilosti jednotlivých scénářů. U každého laboratorního úkolu byla provedena kontrola správného spuštění prostředí, dostupnosti aplikace na očekávaném síťovém portu a možnosti provedení všech kriticky důležitých kroků vedoucích k demonstraci zranitelnosti. V tomto kontextu bylo nutné ověřit nejen samotnou existenci zranitelnosti v aplikaci, ale také determinovanost a ověřitelnost výsledku jejího zneužití. Testování proto zahrnovalo zejména kontrolu dostupnosti webového rozhraní, ověření funkčnosti HTTP připojení, potvrzení vytvoření webového obalu, extrakci důvěrných dat, získání neoprávněného přístupu nebo ověření změn provedených v aplikaci.

Druhým podstatným kritériem byla stabilita a reprodukovatelnost provádění laboratorních úkolů. Vzhledem k předpokládanému opakovanému použití scénářů bylo nutné ověřit možnost jejich opakovaného spuštění v srovnatelném stavu. V tomto ohledu se výhoda kontejnerového prostředí stala zřejmou, protože většina úkolů mohla být po zastavení a následném spuštění vrácena do původního stavu bez nutnosti provádět složité přenastavení. Současně bylo v průběhu testování zjištěno, že ne všechny scénáře vykazují stejnou stabilitu chování. U některých úloh bylo nutné počítat s delším obdobím inicializace aplikace, zatímco v jiných případech bylo nutné do postupu zahrnout kontrolní fázi pro ověření připravenosti služby k provedení testování.

Časová složka byla třetím parametrem, který bylo třeba sledovat. Každý úkol byl posuzován z hlediska jeho proveditelnosti v rámci jednoho výukového bloku. Výsledky testování ukázaly, že většina scénářů, za předpokladu adekvátní přípravy prostředí, odpovídá stanoveným časovým rámcům. Některé úkoly však, zejména v počátečních fázích osvojování funkcí aplikace nebo při nastavování požadovaných nástrojů, vyžadovaly více času. Typickými příklady jsou scénáře předpokládající zachycení a modifikaci HTTP provozu nebo práci s exploity v izolovaném virtuálním prostředí. V souvislosti s tím bylo ve fázi finální úpravy protokolů rozhodnuto o vyloučení nadbytečných nebo příliš složitých fází s cílem soustředit se na prvky, které mají podstatný didaktický význam.

V rámci testování bylo provedeno hodnocení praktických aspektů plnění úkolů, včetně identifikace vznikajících problémů. Byly zaznamenány technické potíže, jako například dočasná nedostupnost služby po inicializaci kontejneru, závislost na specifických nastaveních aplikace nebo nutnost ruční správy prostředí. Kromě toho bylo zjištěno, že některé původně vyvinuté scénáře, navzdory své technické funkčnosti, představovaly pro studenty nadměrnou složitost nebo zmatenost. Tyto okolnosti posloužily jako základ pro provedení změn

v postupech, zjednodušení jednotlivých komponent nebo přeformulování úkolů s cílem zajistit jejich odbornou správnost a soulad s podmínkami vzdělávacího procesu.

Vyskytl se zásadní problém: ne všechny scénáře získané z vnějších zdrojů byly připraveny k použití „tak, jak jsou“. U řady kontejnerů bylo nutné opakovaně ověřovat jejich správnou funkci. Navíc některé scénáře, které se zdály být vhodné, musely být zcela vyloučeny, protože buď nebyly stabilní, nebo neumožňovaly názorně demonstrovat zranitelnosti. Tento proces jasně ukázal, že testování úkolů není pouhou závěrečnou fází, ale kriticky důležitou součástí vývoje. Bez něj by nebylo možné s jistotou prohlásit, že vytvořený soubor laboratorních prací je prakticky použitelný ve vzdělávacím procesu.

Výsledky testování svědčí o jeho úspěchu. Byla úspěšně ověřena základní funkčnost všech úkolů finální verze, prokázána možnost praktické demonstrace zranitelností a jejich použitelnost v navrhovaném laboratorním prostředí. Souběžně s tím testování odhalilo řadu nuancí, což nám umožnilo upravit jednotlivé protokoly a upřesnit metodiku zpracování. Výsledkem je, že tato fáze podstatně přispěla k vytvoření souboru laboratorních úkolů, který se vyznačuje nejen technickou funkčností, ale také názorností, reprodukovatelností a skutečnou pedagogickou hodnotou.

5.5 Využití ve výuce

Předložené laboratorní úkoly byly vypracovány s ohledem na jejich dvojí funkci: jednak jako technická demonstrace zranitelností, jednak jako komplexní výukový materiál pro kurz věnovaný bezpečnosti webových aplikací. Jejich hlavní přínos spočívá v tom, že umožňují propojit teoretické znalosti s praktickým vyzkoušením útoků v kontrolovaném prostředí. Studenti tak mají možnost nejen se seznámit s popisem jednotlivých kategorií OWASP Top 10, ale také samostatně prozkoumat konkrétní mechanismy zranitelností, sledovat jejich provedení a vyhodnotit jejich důsledky.

Z pedagogického hlediska má jednotná struktura úkolů velký význam. Každý scénář je koncipován tak, aby obsahoval teoretické zdůvodnění, jasně formulovaný úkol, doporučení k přípravě pracovního prostředí, podrobné pokyny a kontrolní otázky. To nejen pomáhá studentům rychleji se zorientovat v látce, ale také dává učitelům možnost flexibilně využívat jednotlivé úkoly ve výuce a minimalizovat tak časovou náročnost na jejich přizpůsobení. Důležitou výhodou je také to, že studenti při práci s různými typy zranitelností postupně osvojují jednotný postup, což přispívá k vytvoření uceleného pohledu na bezpečnostní problémy.

Z hlediska vzdělávání představuje samotné laboratorní prostředí významnou výhodu. Díky integraci Kali Linuxu, Dockeru a předem připravených scénářů mohou studenti opakovaně a bez problémů plnit výukové úkoly. To je obzvláště cenné, protože výukový proces by měl být z technického hlediska co nejvíce předvídatelný a snadno ovladatelný. Kdyby scénáře vyžadovaly složité předběžné nastavení nebo neustálý ruční zásah, bylo by jejich použití ve výuce obtížné. V současné konfiguraci však prostředí umožňuje rychle spouštět určité úkoly a poté je snadno ukončit nebo vrátit do původního stavu.

Existuje několik účinných způsobů, jak začlenit laboratorní úkoly do výuky. Jedním z nich je strukturovaná praktická výuka, při které lektor provází studenty celým procesem krok za krokem. Tento formát je obzvláště užitečný při práci s technicky náročnými úkoly nebo při výuce začínajících specialistů na penetrační testování. Jiný přístup předpokládá samostatné osvojení látky studenty, kde úkol slouží jako jasný plán činnosti a role lektora se omezuje na konzultace. Možný je také hybridní přístup, který kombinuje vysvětlení teoretických základů zranitelnosti s následným samostatným praktickým procvičením.

Pedagogická hodnota spočívá v tom, že úkoly se neomezují pouze na mechanické použití exploitů. Prostřednictvím kontrolních otázek a důrazu na analýzu důsledků jsou studenti podněcováni k úvahám, které přesahují rámec technické realizace útoku. Učí se posuzovat jeho význam pro bezpečnost informačního systému jako celku. Tento přístup je obzvláště důležitý ve vysokoškolském vzdělávání, kde úkolem není pouze osvojení konkrétních nástrojů, ale také formování schopnosti studentů analyzovat problémy, interpretovat získaná data a rozumět širšímu kontextu bezpečnosti.

Další významnou výhodou je rozmanitost praktických scénářů. Úlohy jsou sestaveny s ohledem na všechny kategorie OWASP Top 10:2025 a pokrývají široké spektrum útočných technik, nástrojů a cílů. To studentům umožňuje nejen studovat jednotlivé zranitelnosti, ale také si vytvořit ucelenou představu o tom, jak rozmanité mohou být chyby ve webových aplikacích a jaké jsou jejich skutečné důsledky. Takový přístup nejen zlepšuje porozumění celé oblasti webové bezpečnosti, ale také činí proces učení zajímavějším a zabraňuje nudě z jednotvárných úkolů.

Vhodnost praktického využití těchto úkolů ve vzdělávacím procesu je nejzřetelnější při studiu předmětů souvisejících s počítačovou bezpečností, penetračním testováním a bezpečností webových aplikací. V rámci kurzu „Počítačové viry a bezpečnost“ mohou sloužit jako praktická složka, která doplňuje teoretický materiál a poskytuje studentům možnost přímého pozorování mechanismů útoků a jejich následků. Kromě toho mohou úkoly sloužit jako základ pro seminární cvičení, samostatnou přípravu nebo práci mimo výuku.

Hlavní výhodou tohoto řešení je jeho škálovatelnost. Díky jednotnému prostředí a protokolové základně, na nichž jsou scénáře postaveny, je v budoucnu možné jak přidávat nové laboratorní moduly, tak upravovat stávající tak, aby odpovídaly aktuálním vzdělávacím cílům. Tento přístup zajišťuje dlouhodobou hodnotu celého systému a umožňuje rychle se přizpůsobovat vývoji hrozeb v oblasti webové bezpečnosti a aktualizacím doporučení OWASP.

Lze s jistotou říci, že navrhované laboratorní úkoly významně přispívají k výuce. Poskytují studentům cennou příležitost setkat se s reálnými scénáři útoků v bezpečném prostředí, což napomáhá rozvoji jejich technických dovedností a analytického myšlení. Kromě toho slouží tato cvičení jako hotový a dobře organizovaný zdroj pro lektory. Právě díky své profesní relevanci, technické dostupnosti a didaktické účinnosti je tato sada cvičení vynikajícím nástrojem pro praktickou výuku v oblasti bezpečnosti webových aplikací.

Závěr

Cílem diplomové práce bylo navrhnout a vytvořit soubor laboratorních úloh zaměřených na výuku penetračního testování webových aplikací se zaměřením na zranitelnosti dle OWASP Top 10:2025. Tento cíl byl splněn propojením teoretické analýzy problematiky webové bezpečnosti s návrhem a realizací praktických scénářů, které umožňují bezpečně demonstrovat vybrané typy útoků v kontrolovaném laboratorním prostředí.

V teoretické části práce byly zpracovány hlavní kategorie zranitelností webových aplikací podle OWASP Top 10:2025. Důraz byl kladen nejen na jejich definici a mechanismus vzniku, ale také na možnosti jejich zneužití a dopady na informační systémy. Tato část vytvořila odborný základ pro navazující řešení a ukázala, že bezpečnost webových aplikací nelze chápat pouze jako otázku jednotlivých chyb v kódu, ale jako širší problém zahrnující návrh aplikace, konfiguraci prostředí, správu identit, integritu dat i schopnost detekce incidentů.

Na teoretická východiska navázal návrh laboratorního prostředí založeného na technologiích Kali Linux, Docker a Vulhub. Zvolené řešení umožnilo vytvořit stabilní, opakovatelné a bezpečné prostředí pro praktické ověřování zranitelností. V práci byla popsána i struktura prostředí a metodika testování, která vychází z logiky reálného penetračního testování, tedy z průzkumu, exploitace a následné analýzy dopadů.

Hlavním výstupem práce je sada deseti laboratorních úloh. Většina z nich byla realizována na základě scénářů z projektu Vulhub, které byly upraveny pro potřeby výuky. U kategorií A04: Cryptographic Failures a A09: Security Logging and Alerting Failures však nebylo možné využít vhodné externí scénáře, a proto byly vytvořeny dvě vlastní webové aplikace – SecureNote a Employee Portal. Právě tento autorský přínos představuje důležitou součást práce, protože umožnil pokrýt i ty oblasti OWASP Top 10, které by jinak zůstaly bez odpovídající laboratorní demonstrace.

Vytvořené laboratorní úlohy byly navrženy v jednotné struktuře, která zahrnuje teoretický rozbor, zadání úkolu, přípravu prostředí, podrobný postup a kontrolní otázky. Tento přístup se ukázal jako vhodný jak z technického, tak z didaktického hlediska. Výsledkem tak není pouze soubor technických cvičení, ale ucelený výukový materiál využitelný ve vysokoškolské výuce.

Přínos práce spočívá především ve vytvoření prakticky využitelného systému laboratorních scénářů, který propojuje teorii s reálnou demonstrací útoků v bezpečném prostředí. Práce tak poskytuje základ pro výuku problematiky webové bezpečnosti v předmětu Počítačové viry a bezpečnost a současně nabízí možnost dalšího rozšíření o nové scénáře podle budoucího vývoje v oblasti webových zranitelností.

Na základě dosažených výsledků lze konstatovat, že práce splnila stanovený cíl a vytvořila odborně i pedagogicky využitelný výstup. Současně potvrdila, že praktická demonstrace zranitelností představuje důležitý nástroj pro lepší pochopení bezpečnosti webových aplikací a pro přípravu studentů na reálné problémy v oblasti kybernetické bezpečnosti.

Seznam použité literatury

[1] BOUTEMEUR, Jamila; LELLA, Ifigeneia; BAKATSIS, Ilias; CHATZICHRISTOS, Georgios; FOLEY, Kevin et al. European Union Agency for Cybersecurity. *ENISA THREAT LANDSCAPE 2025.* 2025, s. 89. ISSN 2363-3050.

[2] OWASP. *OWASP Foundation – Open Worldwide Application Security Project.* [online]. Dostupné z: https://owasp.org/

[3] VERIZON. *2025 Data Breach Investigations Report.* Verizon Business, 2025. Dostupné z: https://www.verizon.com/business/resources/T16f/reports/2025-dbir-data-breach-investigations-report.pdf

[4] SINGER, P.W a FRIEDMAN, Allan. *Cybersecurity and Cyberwar: What Everyone Needs to Know®*. Oxford University Press, 2014. ISBN 978-0-19-991809-6.

[5] KOLOUCH, Jan; KROPÁČOVÁ, Andrea; KUNC, Martin a BAŠTA, Pavel. *CyberSecurity*. Praha: CZ.NIC, z.s.p.o., 2019. ISBN 978-80-88168-34-8.

[6] EUROPOL. *Internet Organised Crime Threat Assessment (IOCTA) 2024.* The Hague: Europol, 2024. [online]. Dostupné z: https://www.europol.europa.eu/publication-events/main-reports/internet-organised-crime-threat-assessment-iocta-2024

[7] CISA. *Supply Chain Compromise.* [online]. Dostupné z: https://www.cisa.gov/news-events/alerts/2021/01/07/supply-chain-compromise

[8] NIST. *Cloud Computing Security Reference Architecture.* Gaithersburg: NIST. [online]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication500-292.pdf

[9] European Union. *Nařízení Evropského parlamentu a Rady (EU) 2016/679 ze dne 27. dubna 2016 o ochraně fyzických osob v souvislosti se zpracováním osobních údajů a o volném pohybu těchto údajů (General Data Protection Regulation – GDPR).* Úřední věstník Evropské unie, 2016. [online]. Dostupné z: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[10] STUTTARD, Dafydd a PINTO, Marcus. *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws.* Indianapolis: Wiley Publishing, 2008. ISBN 978-0-470-17077-9.

[11] WEIDMAN, Georgia. *Penetration testing: a hands-on introduction to hacking.* San Francisco: No Starch Press, 2014. ISBN 978-1-59327-564-8.

[12] XYGENI. *OWASP Top 10 2025 Explained for Developers.* [online]. Xygeni, 2025. Dostupné z: https://xygeni.io/blog/owasp-top-10-2025-explained-for-developers/

[13] NATIONAL INSTITUTE OF STANDARDS AND TECH. *Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms*. Independently Published, 2019. ISBN 1078206384.

[14] SCAMBRAY, Joel; LIU, Vincent a SIMA, Caleb. *Hacking exposed: web applications : web application security secrets and solutions.* 3rd ed. New York: McGraw-Hill, c2011. ISBN 978-0-07-174064-7.

[15] KOLOUCH, Jan. *CyberCrime*. Praha: CZ.NIC, z.s.p.o., 2016. ISBN 978-80-88168-18-8.

[16] U. S. DEPARTMENT COMMERCE a NATIONAL INSTITUTE AND TECHNOLOGY. *Systems Security Engineering.* CreateSpace Independent Publishing Platform, 2017. ISBN 1548558141.
[17] NATIONAL INSTITUTE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Digital Identity Guidelines.* CreateSpace Independent Publishing Platform, 2017. ISBN 1548399132.
[18] NIST. Computer Security Incident Handling Guide. CreateSpace Independent Publishing Platform, 2013. ISBN 1494726378.
[19] KASPERSKI, Kris. Komputernye virusy iznutri i snaruzhi. Piter, 2013. ISBN 5-469-00982-3.
[20] NIST. Secure Software Development Framework (SSDF) Version 1.1*: Recommendations for Mitigating the Risk of Software Vulnerabilities.* Gaithersburg: NIST, 2022. [online]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-218.pdf
[21] HADNAGY, Christopher. *Social engineering: the science of human hacking.* Second edition. Indianapolis, IN: Wiley, [2018]. ISBN 978-1-119-43338-5.
[22] PTES. *Technical Guidelines*. Online. 2012. Dostupné z: http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines
[23] NIST. *Technical Guide to Information Security Testing and Assessment.* CreateSpace Independent Publishing Platform, 2014. ISBN 1495215539.
[24] HERZOG, Pete. *The Open Source Security Testing Methodology Manual*. Manual. New York, USA: ISECOM, 2010. Dostupné z: https://www.isecom.org/OSSTMM.3.pdf
[25] KHAWAJA, Gus. Kali Linux penetration testing bible. Indianapolis, Indiana: Wiley, [2021]. ISBN 978-1-119-71908-3.
[26] LOZANO, Carlos A.; SHAH, Dhruv a WALIKAR, Riyaz Ahemed. *Hands-on appplication penetration testing with burp suite: use burp suite and its features to inspect, detect, and exploit security vulnerabilities in your web applications.* Birmingham: Packt, [2019]. ISBN 978-1-78899-406-4.
[27] LYON, Gordon. *Nmap network scanning: official Nmap project guide to network discovery and security scanning.* Sunnyvale: Insecure, 2008. ISBN 978-0-9799587-1-7.
[28] OREBAUGH, Angela. *Wireshark a Ethereal: kompletní průvodce analýzou a diagnostikou sítí.* Brno: Computer Press, 2008. ISBN 978-80-251-2048-4.
[29] sqlmap: *Automatic SQL injection and database takeover tool* [online]. Dostupné z: https://sqlmap.org/
[30] KENNEDY, David; AHARONI, Mati; KEARNS, Devon; O'GORMAN, Jim a GRAHAM, Daniel G. *Metasploit: the penetration tester's guide*. 2nd edition. San Francisco: No Starch Press, [2025]. ISBN 978-1-71850-298-7.
[31] Gobuster. *Gobuster – Directory/File, DNS and VHost busting tool written in Go* [online]. Dostupné z: https://gobuster.org/
[32] CIRT.NET. The Nikto Web Scanner. Online. 2025. Dostupné z: https://cirt.net/nikto/.
[33] Openwall. *John the Ripper password cracker* Online. Dostupné z: https://www.openwall.com/john/
[34] Docker. *Docker: Accelerated Container Application Development.* Online. Dostupné z: https://www.docker.com/
[35] Vulhub. *Vulhub: Open-source vulnerable Docker environments* [online]. Dostupné z: https://vulhub.org/

Příloha A: Seznam přiložených souborů

Umístění a název souboru	Popis
\Protokoly\ZaykovDPpr1.docx	Laboratorní úloha č. 1 – Drupal
\Protokoly\ZaykovDPpr2.docx	Laboratorní úloha č. 2 – Joomla
\Protokoly\ZaykovDPpr3.docx	Laboratorní úloha č. 3 – Redis
\Protokoly\ZaykovDPpr4.docx	Laboratorní úloha č. 4 – Apache
\Protokoly\ZaykovDPpr5.docx	Laboratorní úloha č. 5 – Spring4Shell
\Protokoly\ZaykovDPpr6.docx	Laboratorní úloha č. 6 – Superset
\Protokoly\ZaykovDPpr7.docx	Laboratorní úloha č. 7 – PHP IMAP
\Protokoly\ZaykovDPpr8.docx	Laboratorní úloha č. 8 – PHP XXE Injection
\Protokoly\ZaykovDPpr9.docx	Laboratorní úloha č. 9 – SecureNote
\Protokoly\ZaykovDPpr10.docx	Laboratorní úloha č. 10 – Employee Portal
\Vzorove_Protokoly\ZaykovDPpr1_VZOR.docx	Vzorová laboratorní úloha č. 1 – Drupal
\Vzorove_Protokoly\ZaykovDPpr2_VZOR.docx	Vzorová laboratorní úloha č. 2 – Joomla
\Vzorove_Protokoly\ZaykovDPpr3_VZOR.docx	Vzorová laboratorní úloha č. 3 – Redis
\Vzorove_Protokoly\ZaykovDPpr4_VZOR.docx	Vzorová laboratorní úloha č. 4 – Apache
\Vzorove_Protokoly\ZaykovDPpr5_VZOR.docx	Vzorová laboratorní úloha č. 5 – Spring4Shell
\Vzorove_Protokoly\ZaykovDPpr6_VZOR.docx	Vzorová laboratorní úloha č. 6 – Superset
\Vzorove_Protokoly\ZaykovDPpr7_VZOR.docx	Vzorová laboratorní úloha č. 7 – PHP IMAP
\Vzorove_Protokoly\ZaykovDPpr8_VZOR.docx	Vzorová laboratorní úloha č. 8 – PHP XXE Injection
\Vzorove_Protokoly\ZaykovDPpr9_VZOR.docx	Vzorová laboratorní úloha č. 9 – SecureNote
\Vzorove_Protokoly\ZaykovDPpr10_VZOR.docx	Vzorová laboratorní úloha č. 10 – Employee Portal
\Zaykov_DP_LP.zip	Laboratorní prostředí